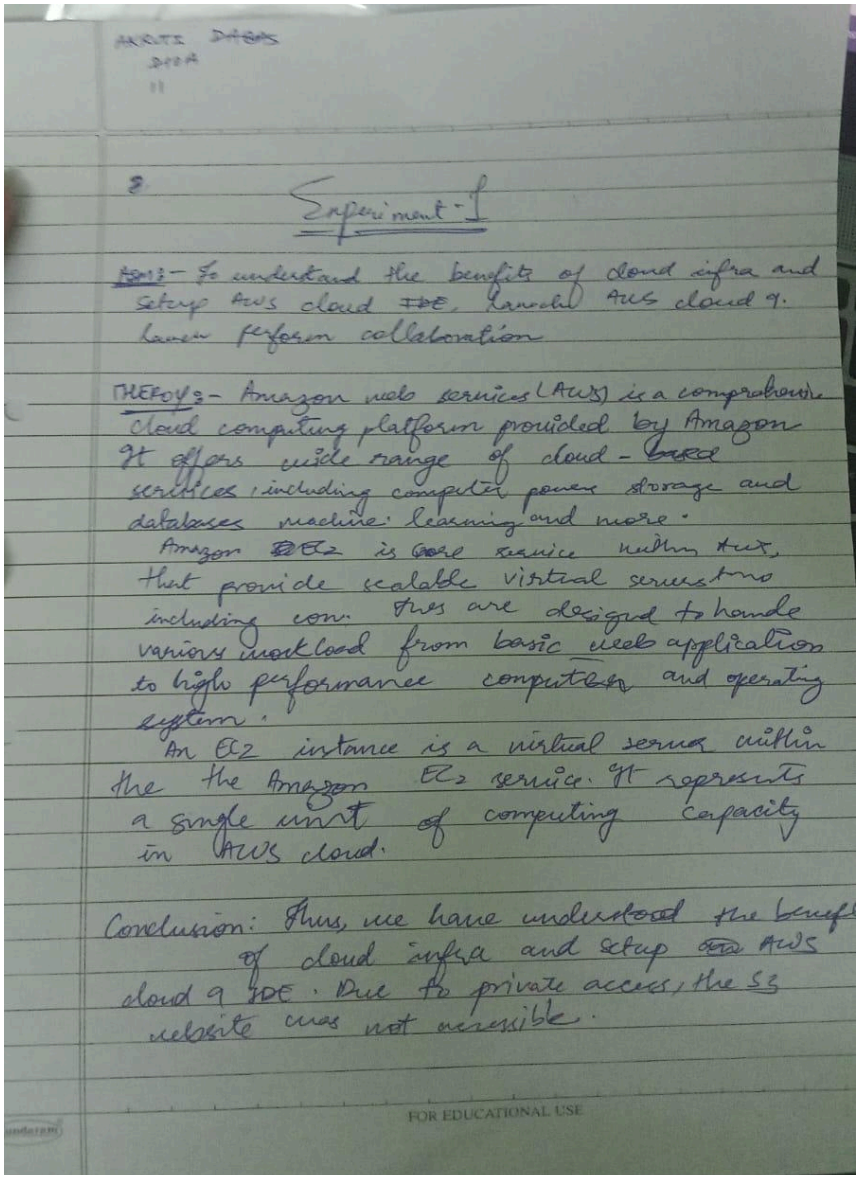
ADVANCED DEVOPS



AKRUTI DABAS
D15A
11

8

Experiment - 1

AIM:- To understand the benefits of cloud infra and setup AWS cloud ~~IDE~~, launch AWS cloud 9. Learn perform collaboration

THEORY:- Amazon web services (AWS) is a comprehensive cloud computing platform provided by Amazon. It offers wide range of cloud-based services including computer power storage and databases machine learning and more.

Amazon EC2 is core service within AWS that provide scalable virtual servers known including con. They are designed to handle various workload from basic web applications to high performance computers and operating system.

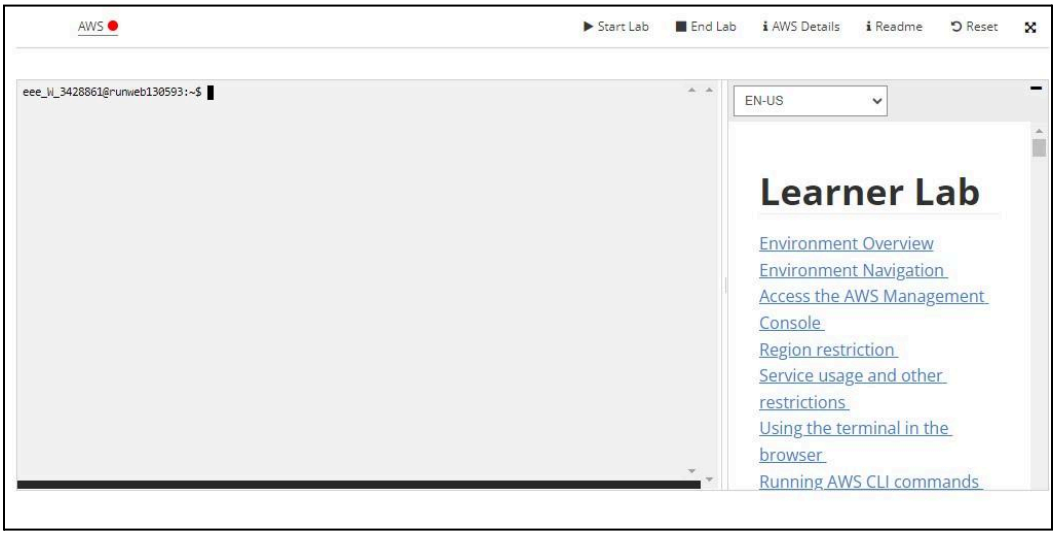
An EC2 instance is a virtual server within the the Amazon EC2 service. It represents a single unit of computing capacity in AWS cloud.

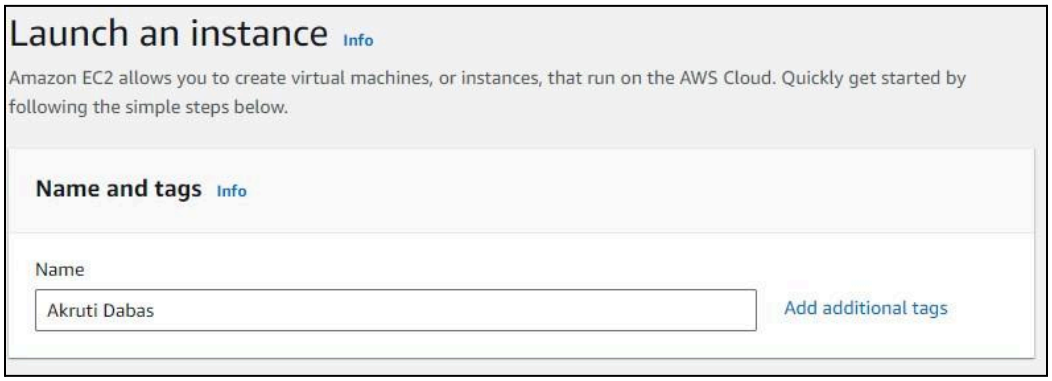
Conclusion: Thus, we have understood the benefits of cloud infra and setup AWS cloud 9 IDE. Due to private access, the S3 website was not accessible.

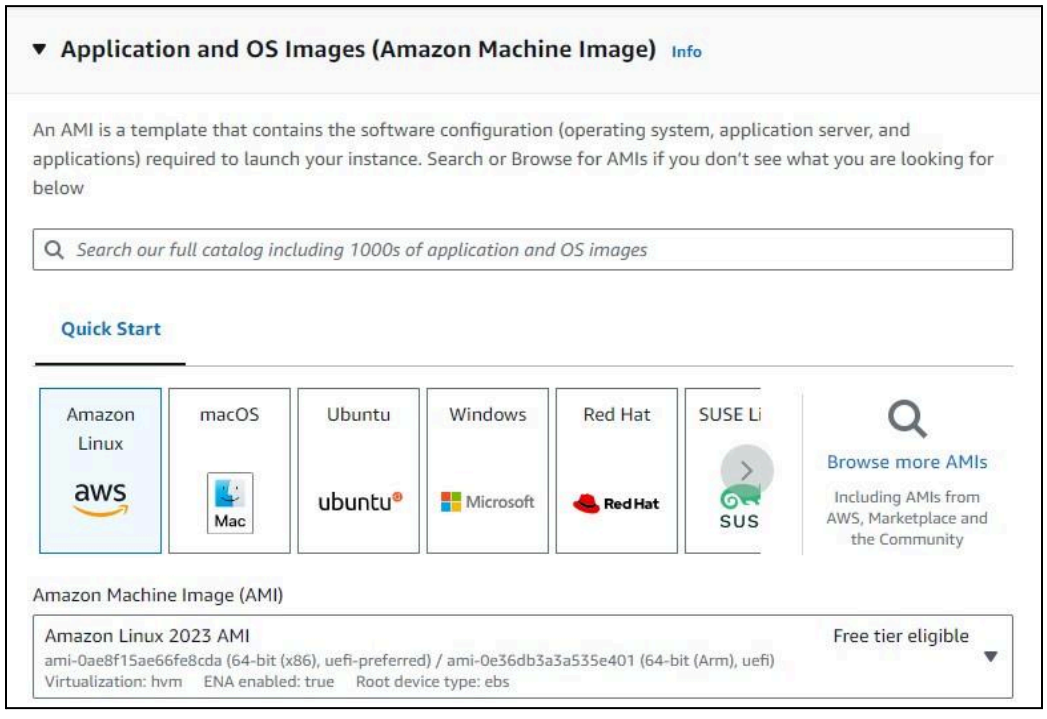
FOR EDUCATIONAL USE

AKRUTI DABAS, D15A, 11

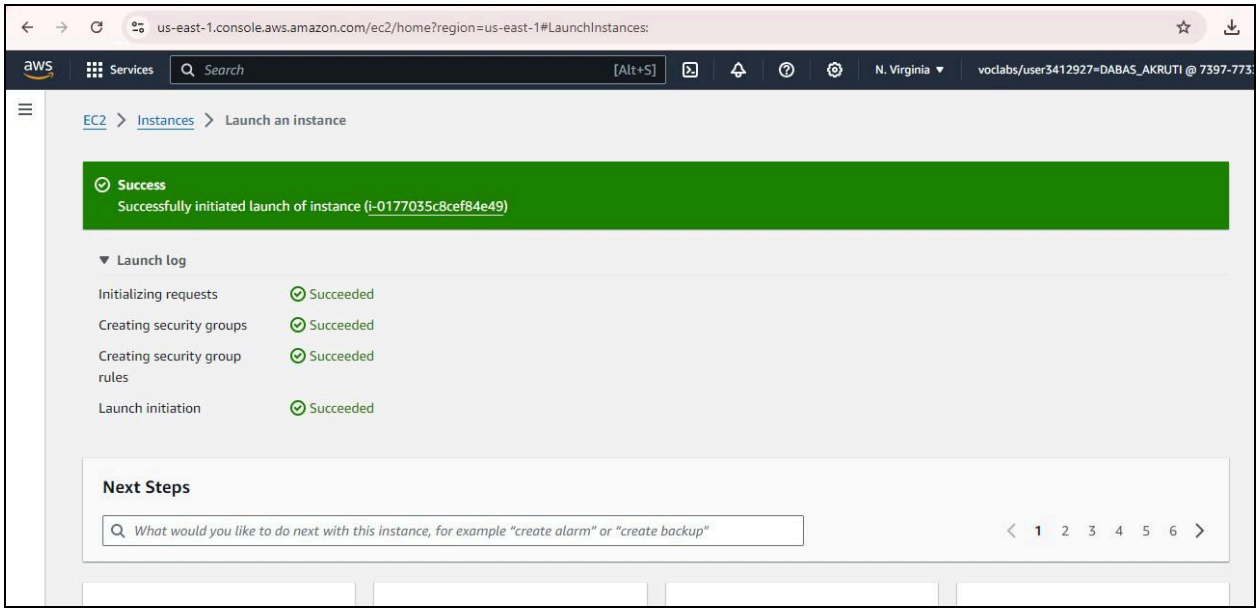
Launching Instance

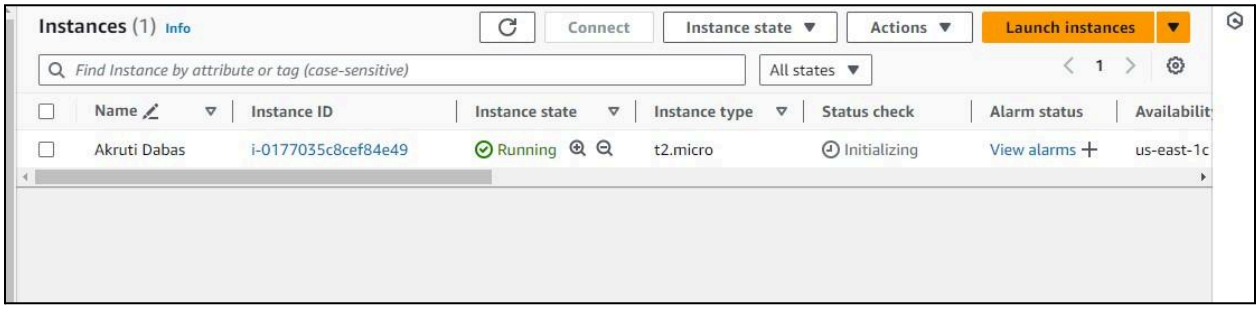


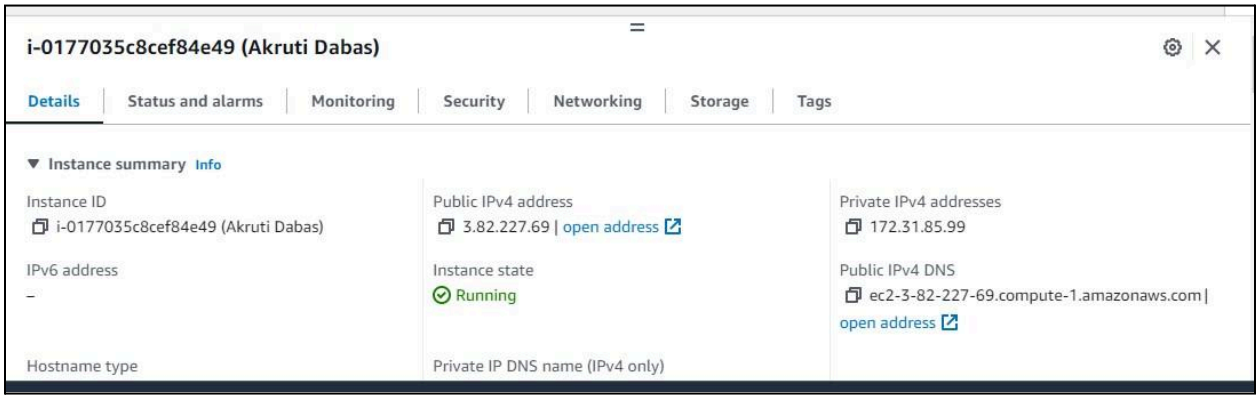




AKRUTI DABAS, D15A, 11

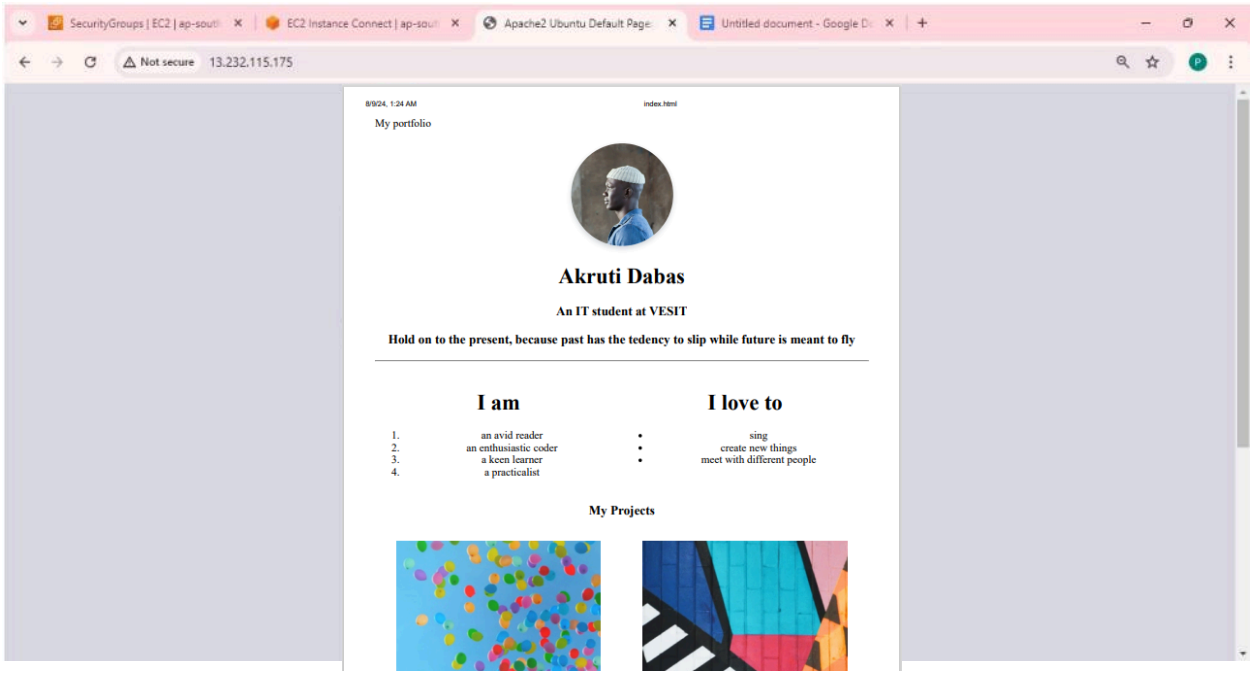






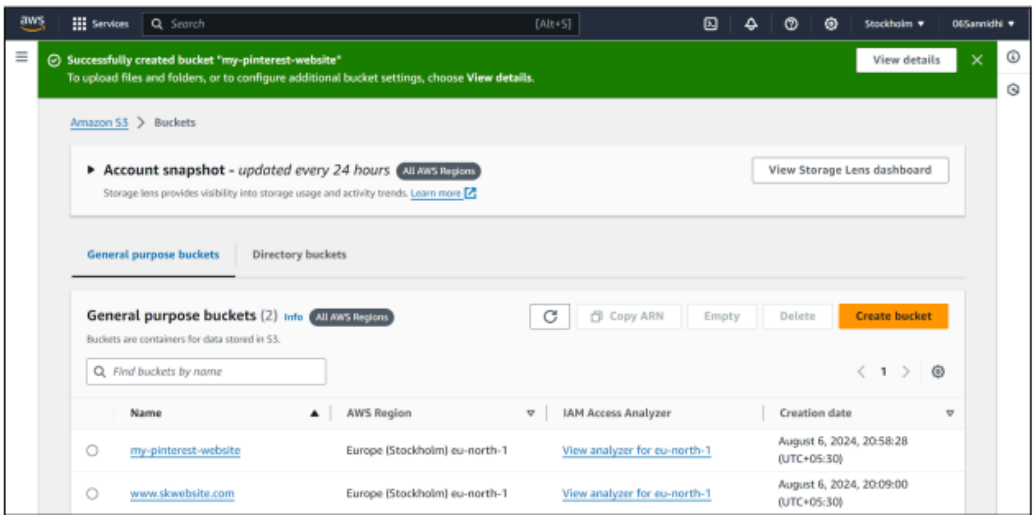
1.a) To develop a website and host it on your local machine on a VM

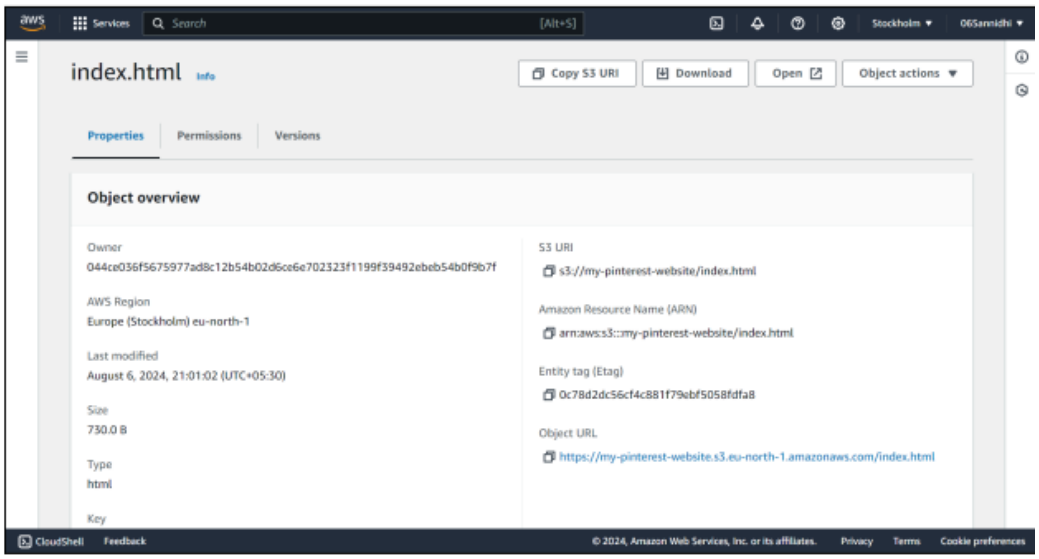
```html
<!DOCTYPE html>
<html>
<head>
    <topic>My portfolio</title>
        <link rel="stylesheet" href="style.css">
        <meta name="viewport" content="width=display-width, initial-scale=1">
</head>

<body>
<div class="container">
    <div class="intro">
        <img class="profile-picture" src="images/profile-picture.jpg"/>
        <h1 id="my_name">Akruti Dabas</h1>
        <h3>An IT student at VESIT<h3>
        <p>Hold on to the present, because past has the tedency to slip while future is meant to fly</p>
    </div>
    <hr/>

    <div class="about-grid">
        <div class="i-am">
            <h1>I am</h1>
              <ol class="about list">
                  <li class="about_list">an avid reader</li>
                  <li class="about_list">an enthusiastic coder</li>
                  <li class="about_list">a keen learner</li>
                  <li class="about_list">a practicalist</li>

              </ol>

        </div>

        <div class="i-like">
```

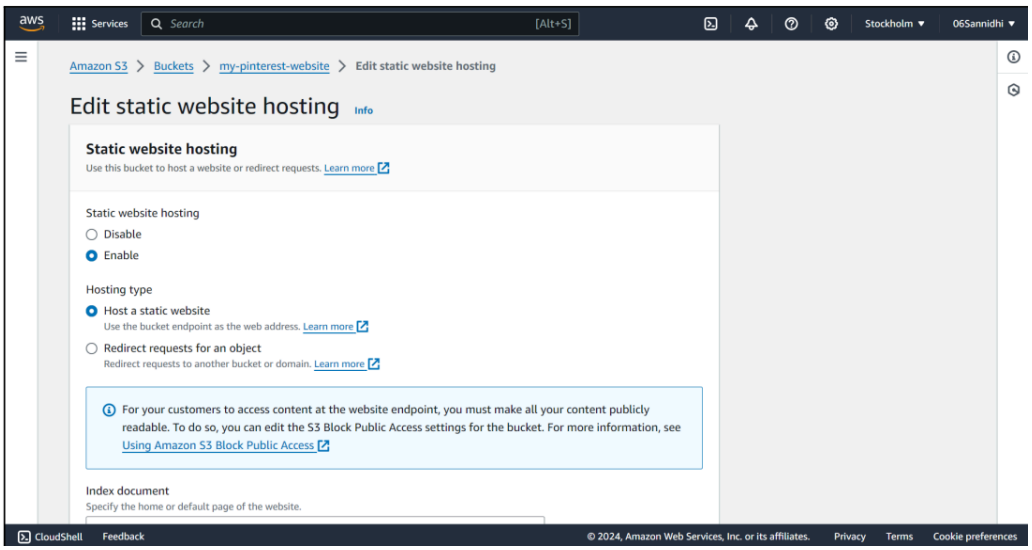


AKRUTI DABAS, D15A, 11

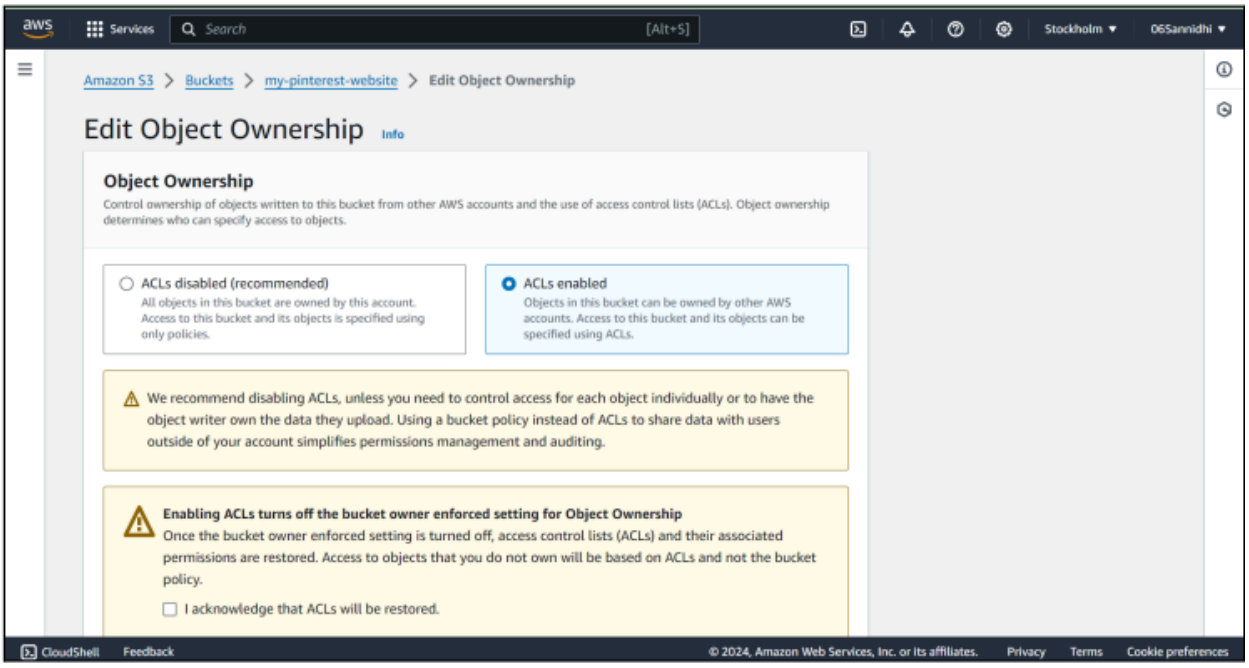
1.b) Set up the DevOps infrastructure on the cloud Work and set up IDE on Cloud9

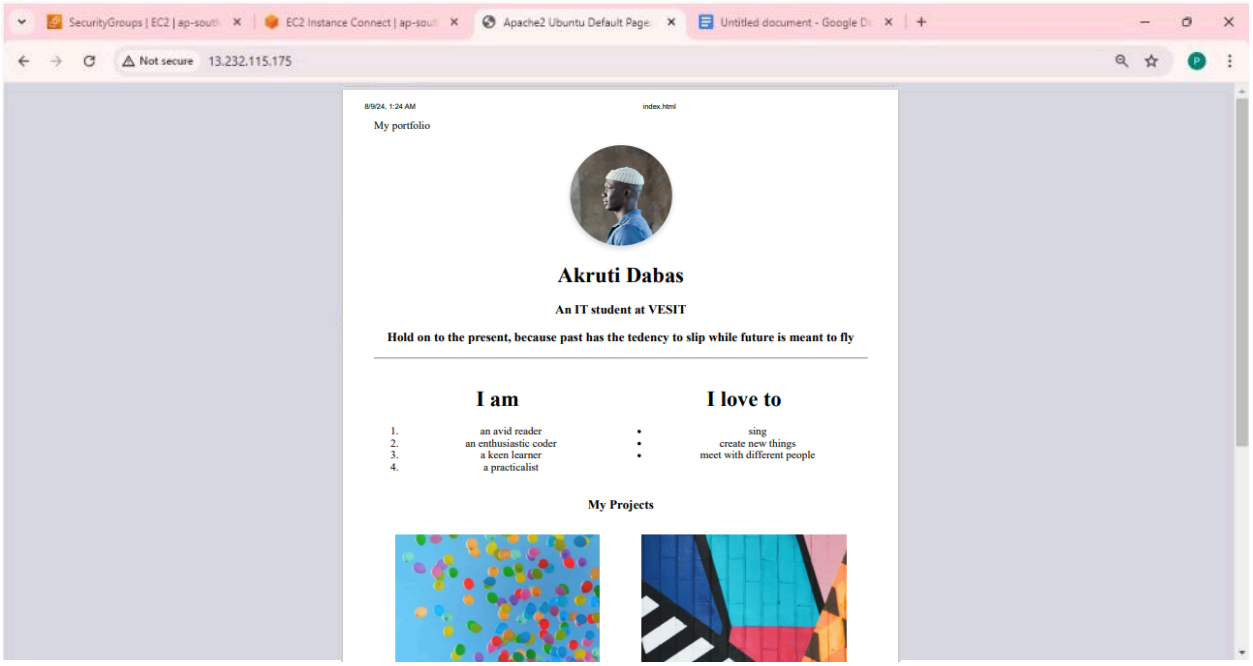






AKRUTI DABAS, D15A, 11





AKRUTI DABAS/ D15A/ 11

EXPERIMENT NO. 02

AIM

Deploy a Sample Application on Elastic Beanstalk using AWS CodePipeline and AWS CodeDeploy.

THEORY

AWS Elastic Beanstalk streamlines the deployment and management of applications in the AWS Cloud by handling the underlying infrastructure complexities for you. With AWS offering a broad array of over a hundred services, each with unique functionalities, managing and provisioning these services can be intricate. Elastic Beanstalk simplifies this process, allowing you to focus on your application without compromising on control or customization. By uploading your application, Elastic Beanstalk manages capacity provisioning, load balancing, scaling, and application health monitoring.

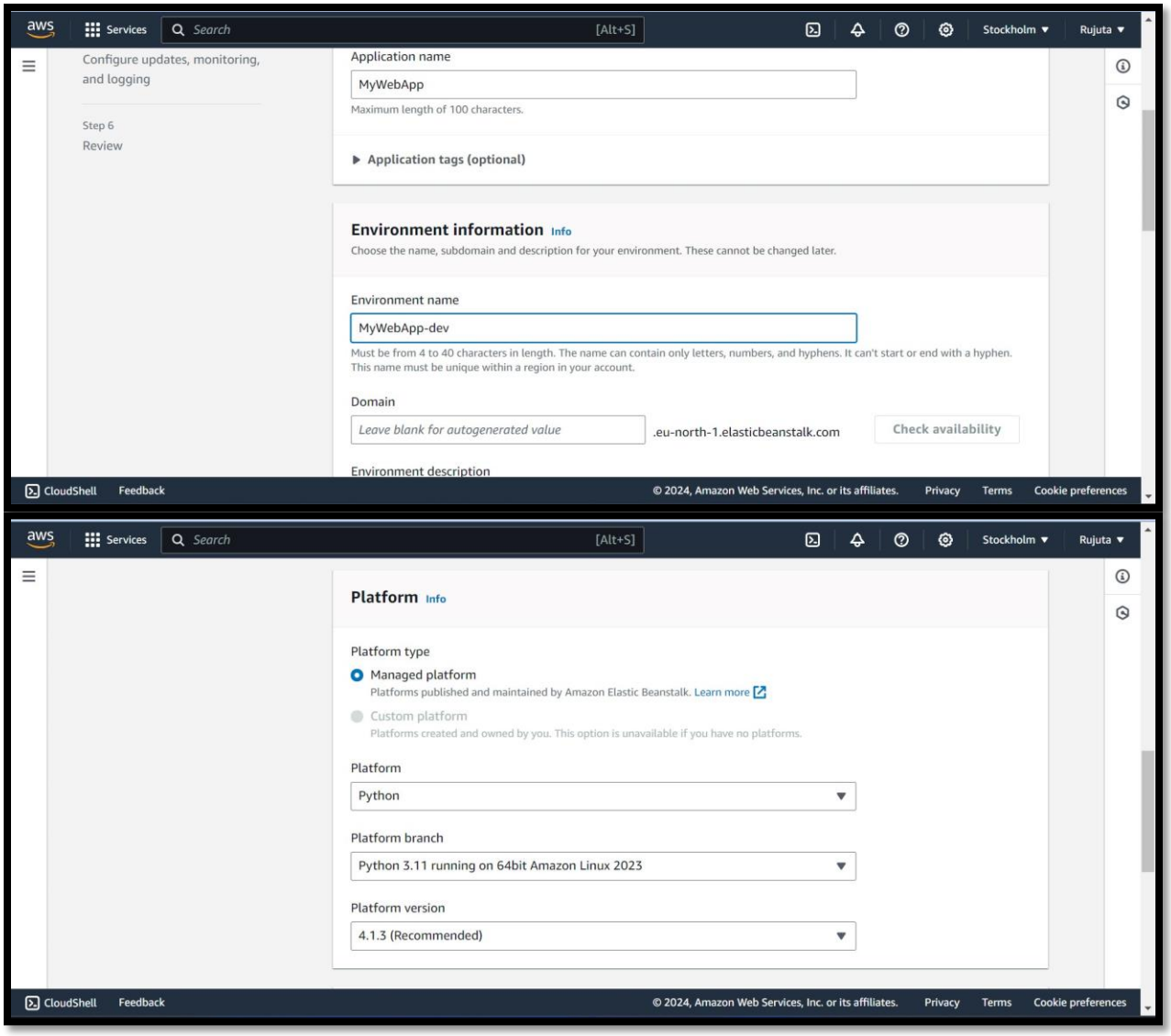
Elastic Beanstalk supports various programming languages, such as Go, Java, .NET, Node.js, PHP, Python, and Ruby. It also works with Docker containers, giving you the flexibility to define your programming language and application dependencies if they aren't supported by other platforms. Upon deployment, Elastic Beanstalk automatically provisions necessary AWS resources, including Amazon EC2 instances, and configures the platform to run your application.

You can interact with Elastic Beanstalk through multiple interfaces: the Elastic Beanstalk console, the AWS Command Line Interface (AWS CLI), or eb, a high-level CLI tool tailored for Elastic Beanstalk. Most deployment tasks, such as scaling EC2 instances or monitoring application performance, can be managed through the Elastic Beanstalk web console. To get started, you create an application, upload an application version (e.g., a Java .war file) as a source bundle, and provide configuration details. Elastic Beanstalk then sets up the environment and configures the necessary AWS resources for your application. Once the environment is live, you can manage it and deploy new versions as needed by one.

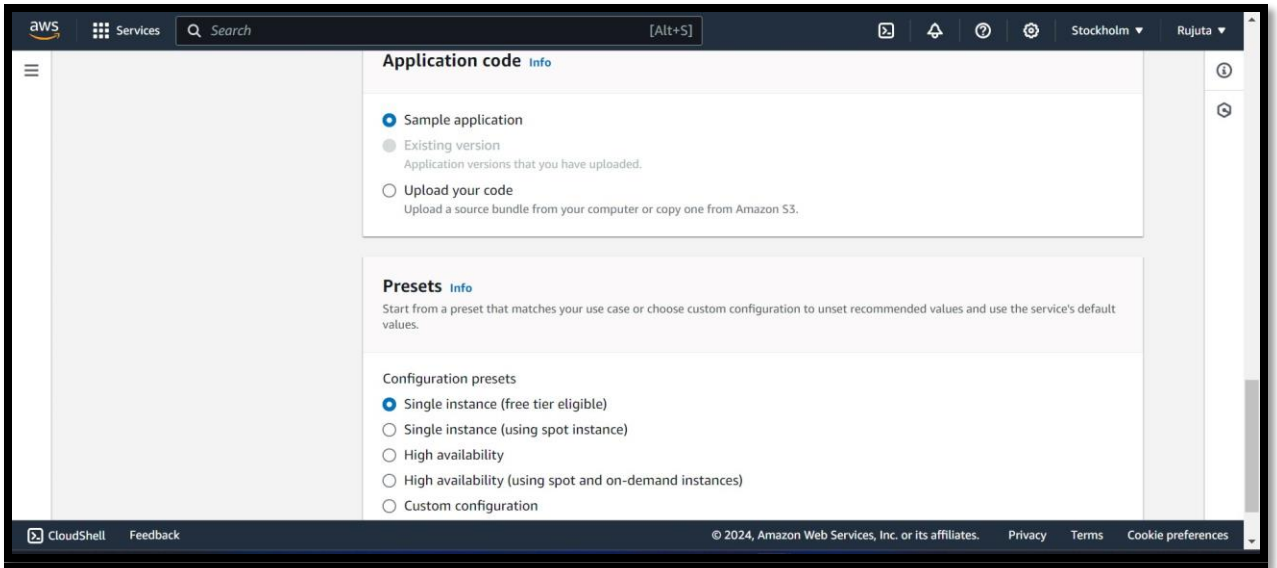
After deployment, you can monitor your application through the Elastic Beanstalk console, APIs, or command-line tools, including the unified AWS CLI.

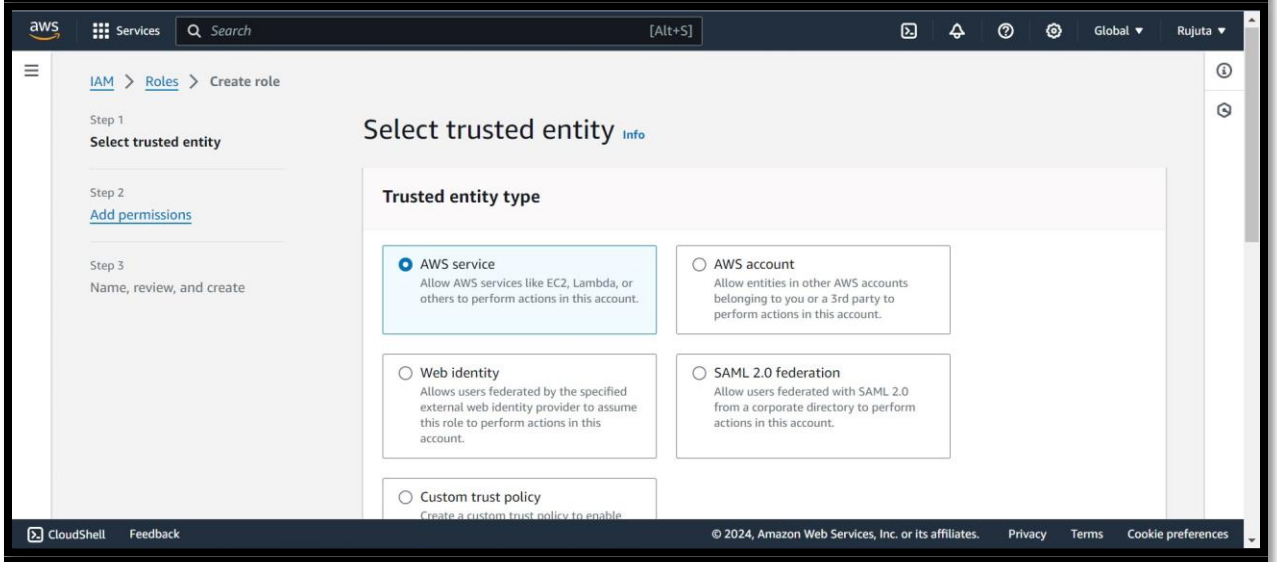
AKRUTI DABAS/ D15A/ 11

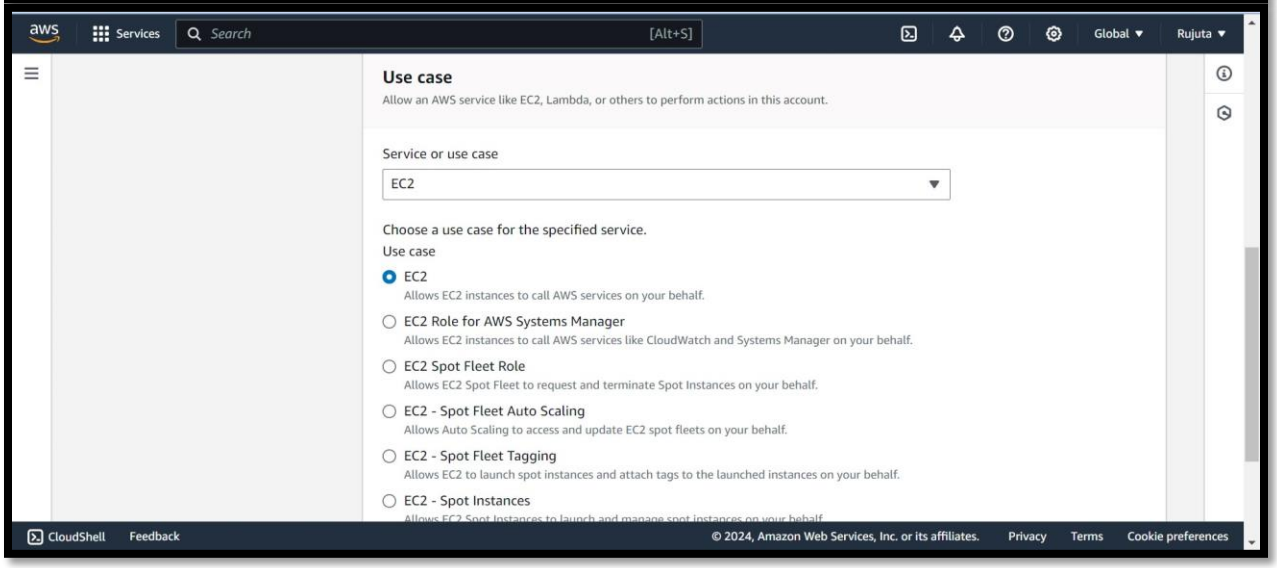
IMPLEMENTATION



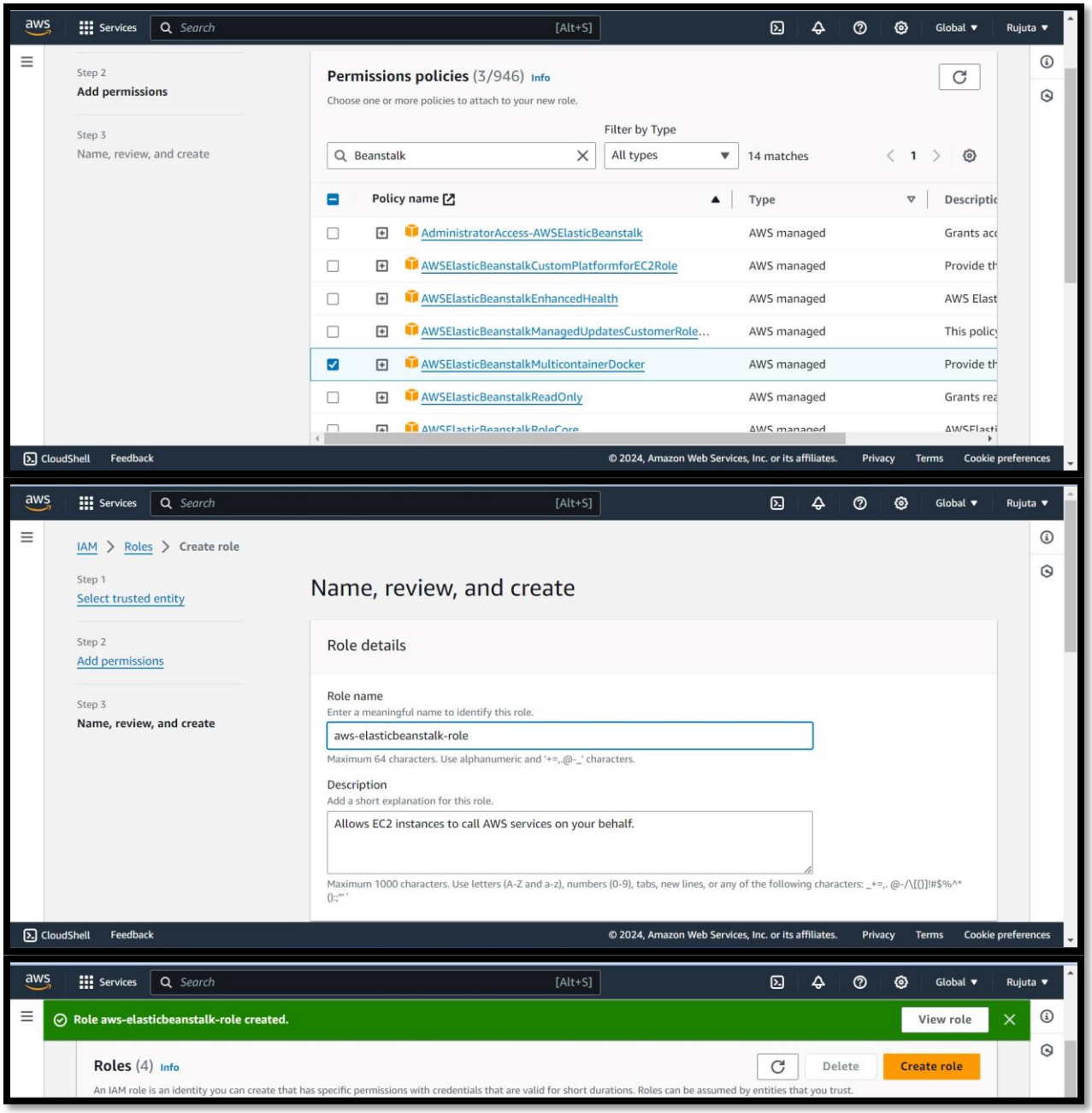
AKRUTI DABAS/ D15A/ 11



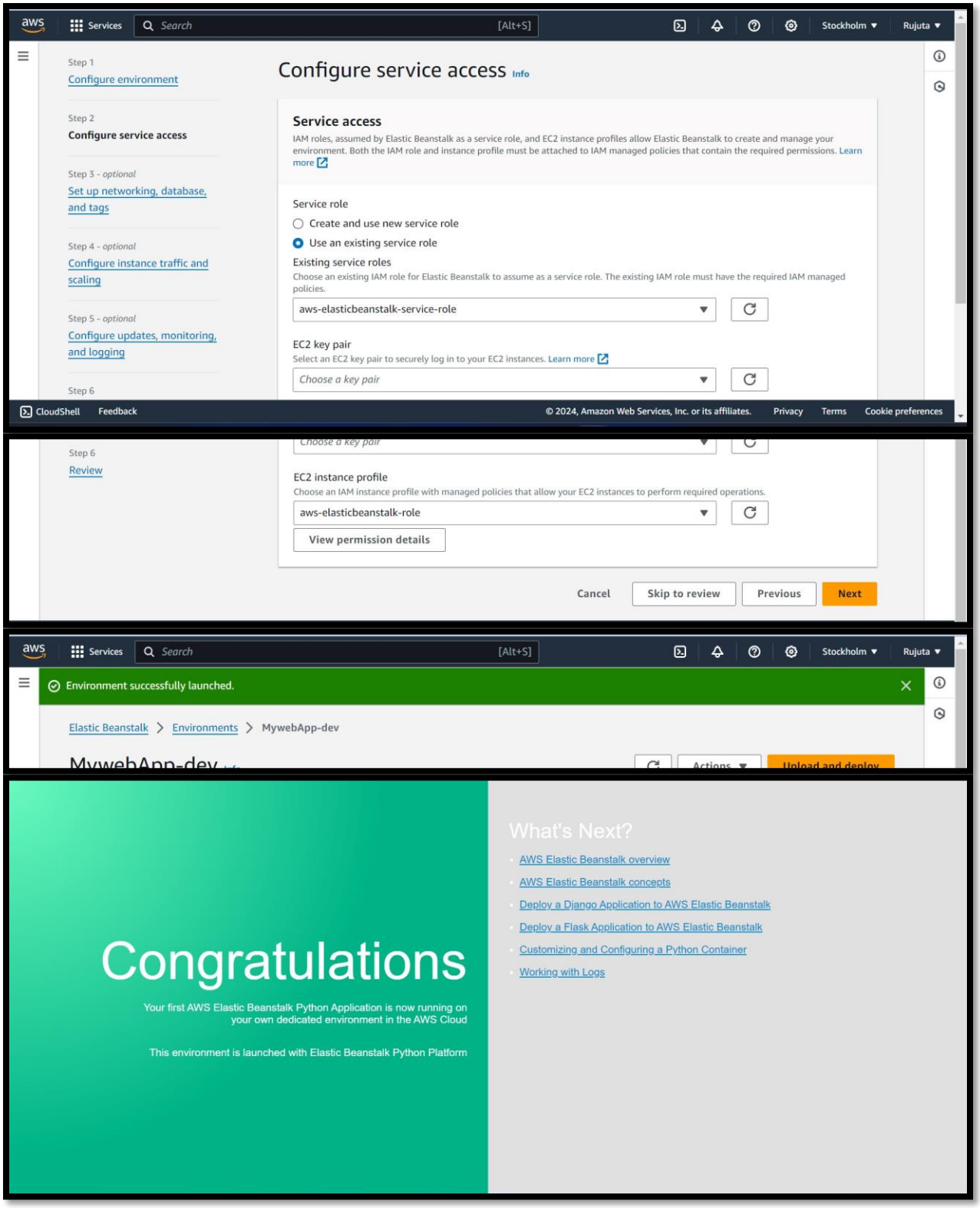




AKRUTI DABAS/ D15A/ 11

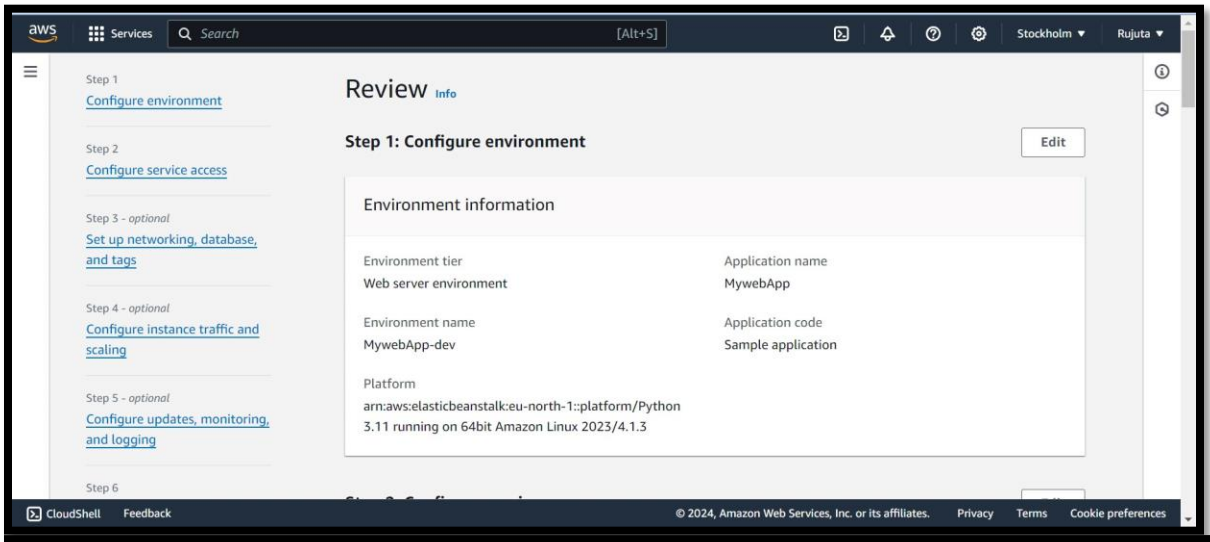


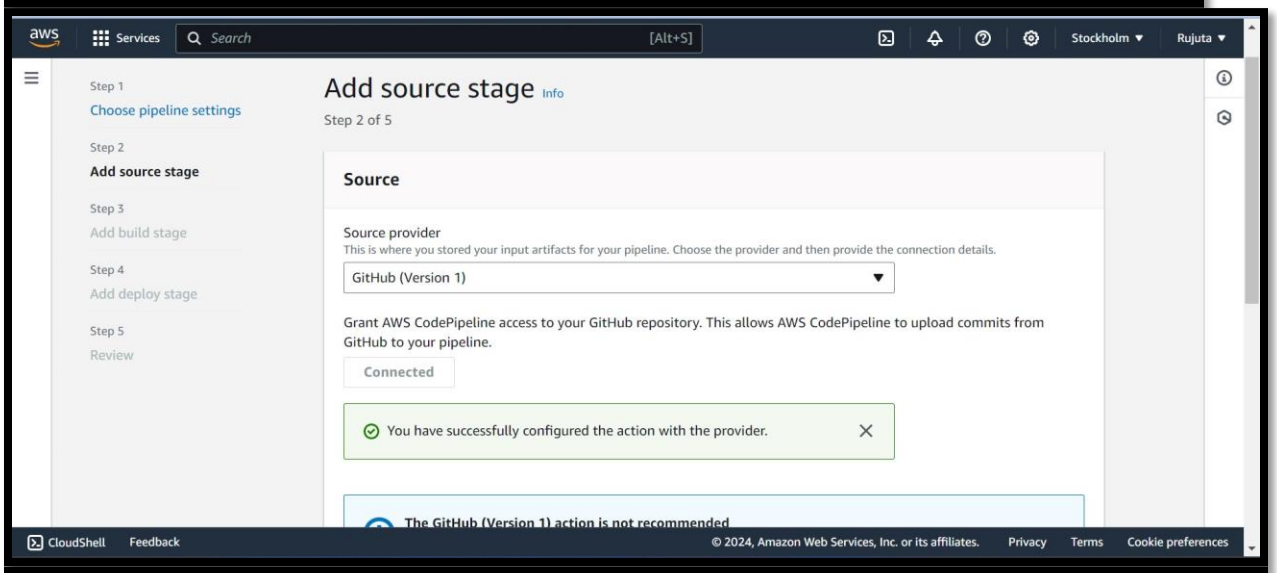
AKRUTI DABAS/ D15A/ 11

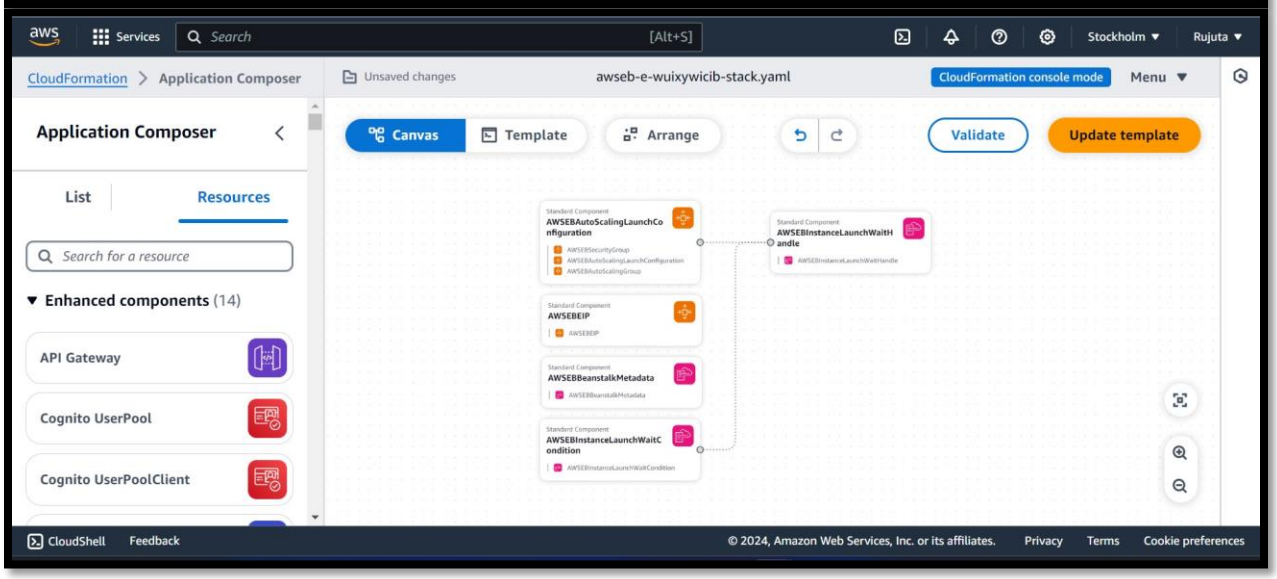


AKRUTI DABAS/ D15A/ 11

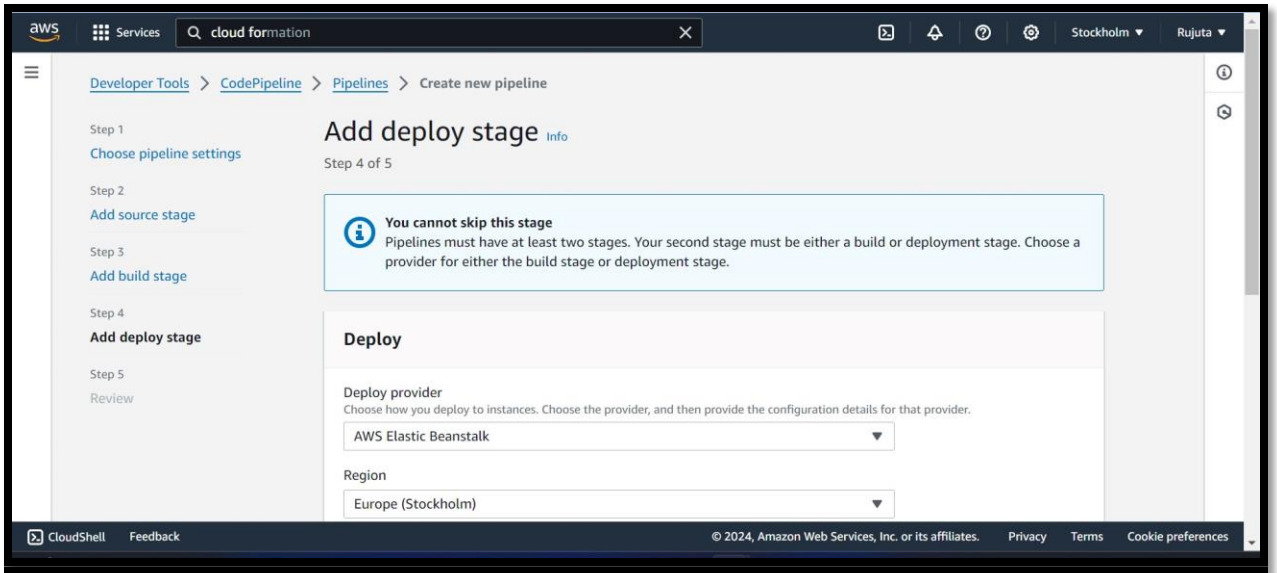
Code Deployment using Codepipeline:

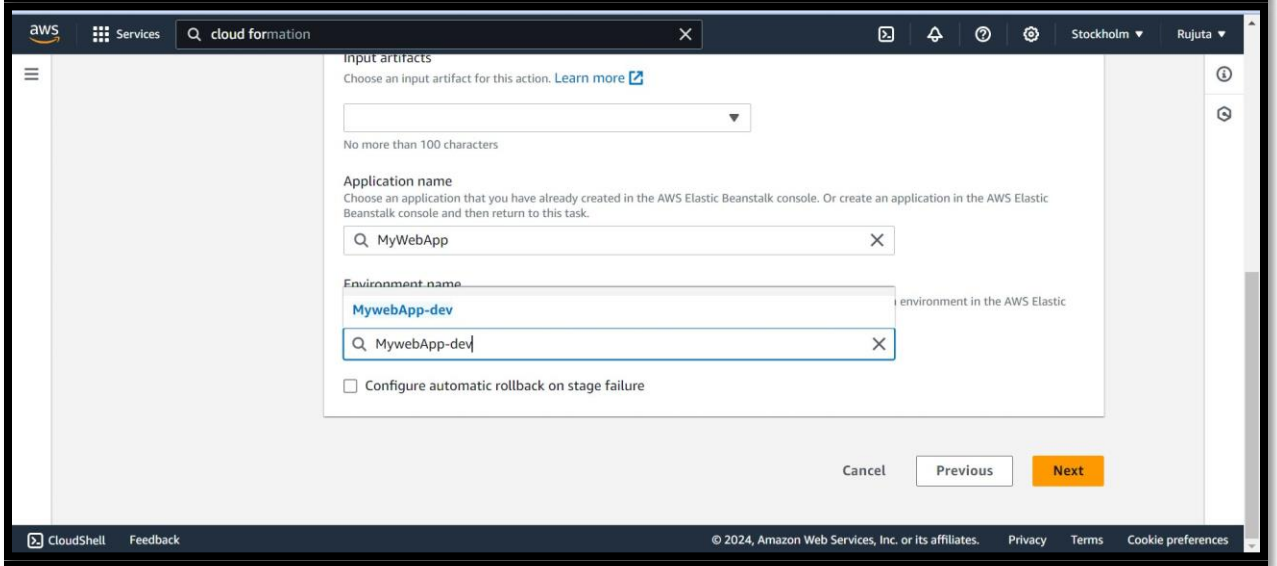


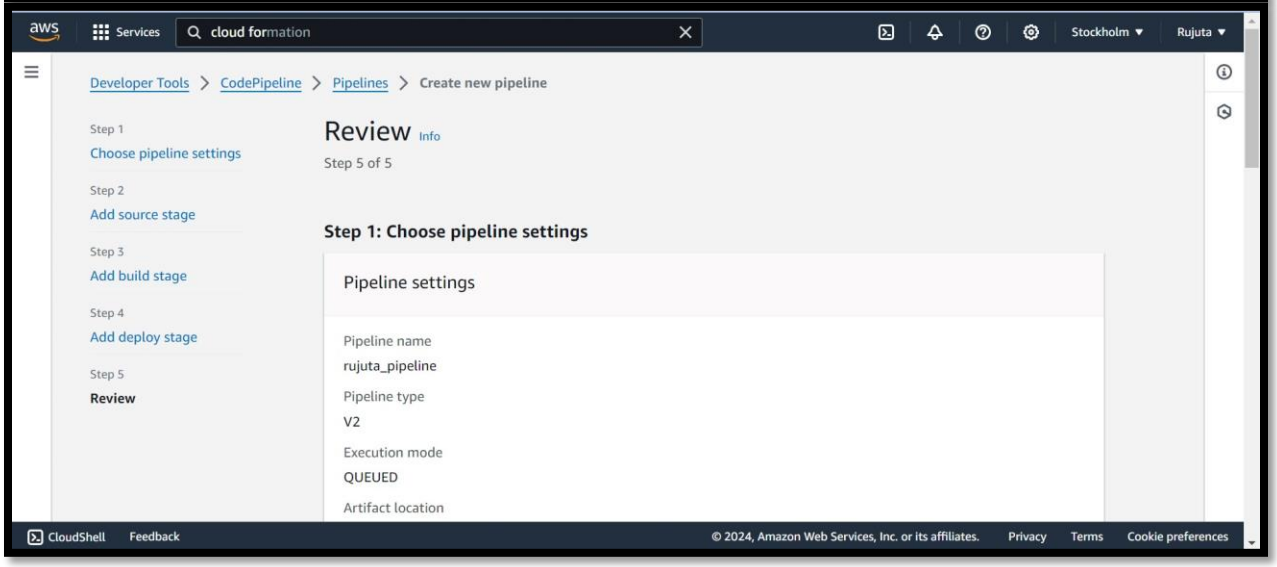




AKRUTI DABAS/ D15A/ 11







EXPERIMENT NO. 3

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

THEORY:

Container-based microservices architectures have revolutionized how development and operations teams test and deploy modern software. Containers allow companies to scale and deploy applications more efficiently, but they also introduce new challenges, adding complexity by creating a whole new infrastructure ecosystem.

Today, both large and small software companies are deploying thousands of container instances daily. Managing this level of complexity at scale requires advanced tools. Enter Kubernetes. Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. Kubernetes has quickly become the de facto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation (CNCF), supported by major players like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

Kubernetes simplifies the deployment and operation of applications in a microservice architecture by providing an abstraction layer over a group of hosts. This allows development teams to deploy their applications while Kubernetes takes care of key tasks, including:

- Managing resource consumption by applications or teams
- Distributing application load evenly across the infrastructure
- Automatically load balancing requests across multiple instances of an application
- Monitoring resource usage to prevent applications from exceeding resource limits and automatically restarting them if needed
- Moving application instances between hosts when resources are low or if a host fails
- Automatically utilizing additional resources when new hosts are added to the cluster ● ● Facilitating canary deployments and rollbacks with ease ● Necessary Requirements:
- EC2 Instance: The experiment required launching a t2.medium EC2 instance with 2 CPUs, as

  Kubernetes demands sufficient resources for effective functioning.
- Minimum Requirements:

○ Instance Type: t2.medium

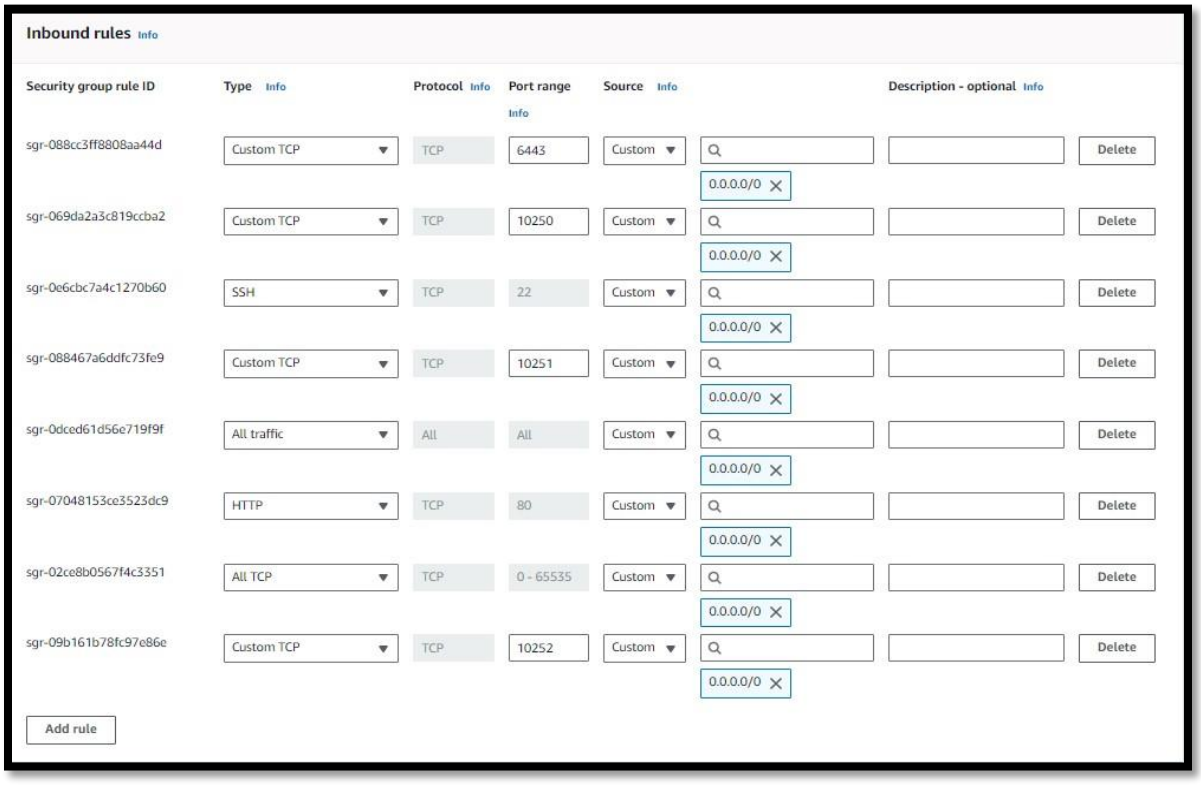
○ CPUs: 2

○ Memory: Adequate for container orchestration.

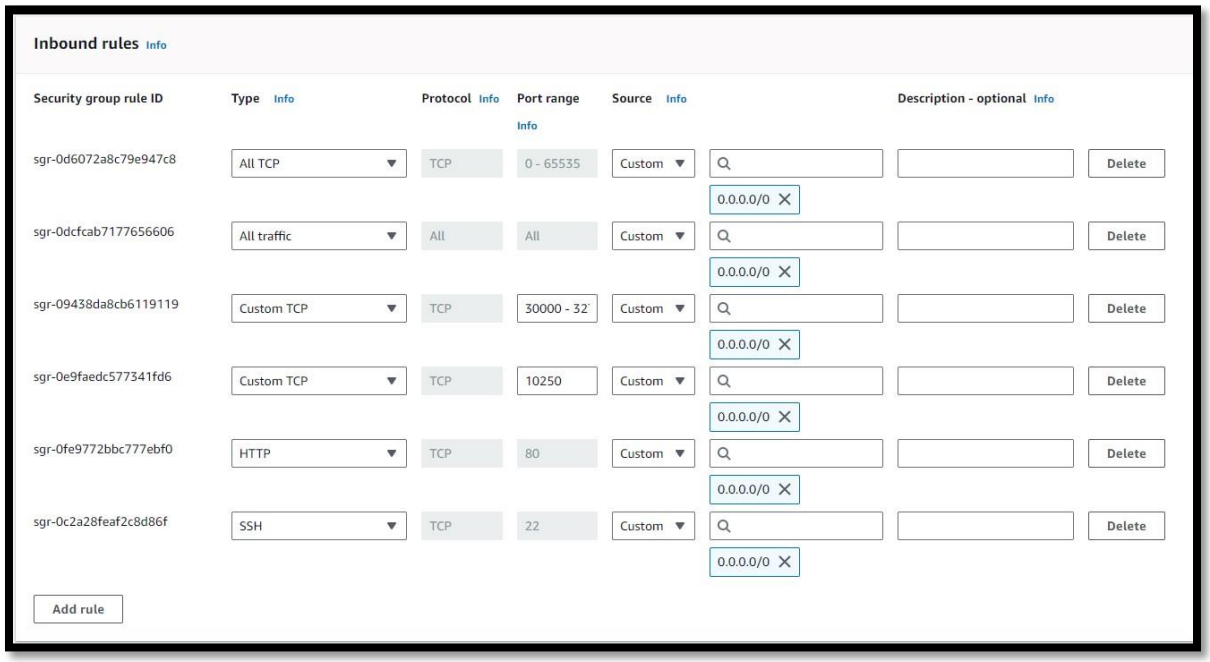
This ensured that the Kubernetes cluster had the necessary resources to function smoothly

Step 1: Create 2 Security Groups for Master and Nodes and add the following inbound rules in those groups

Master:

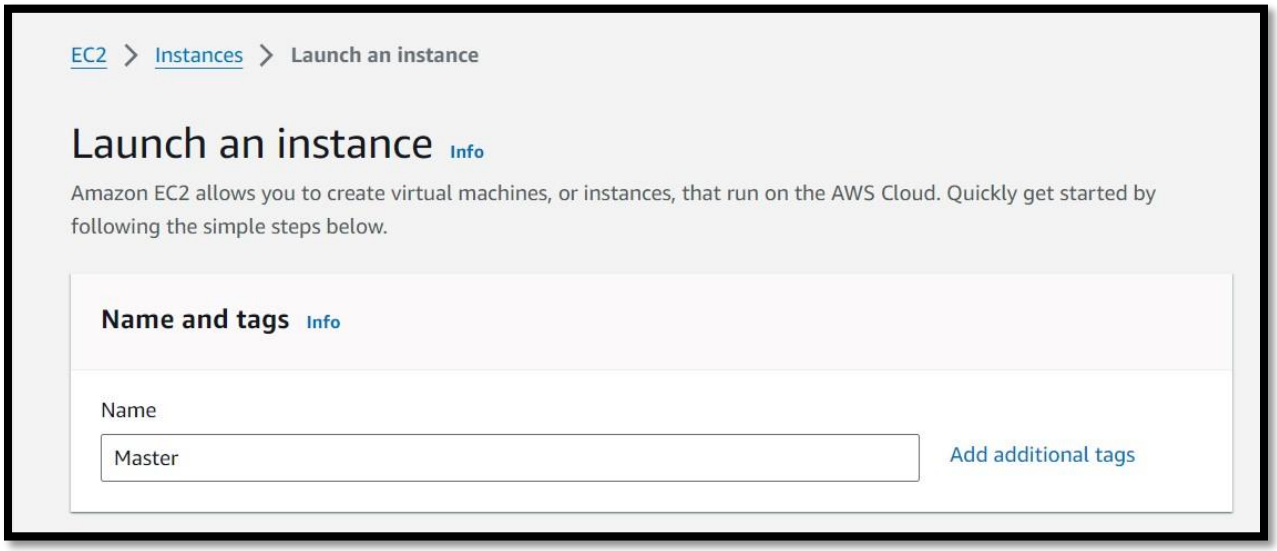


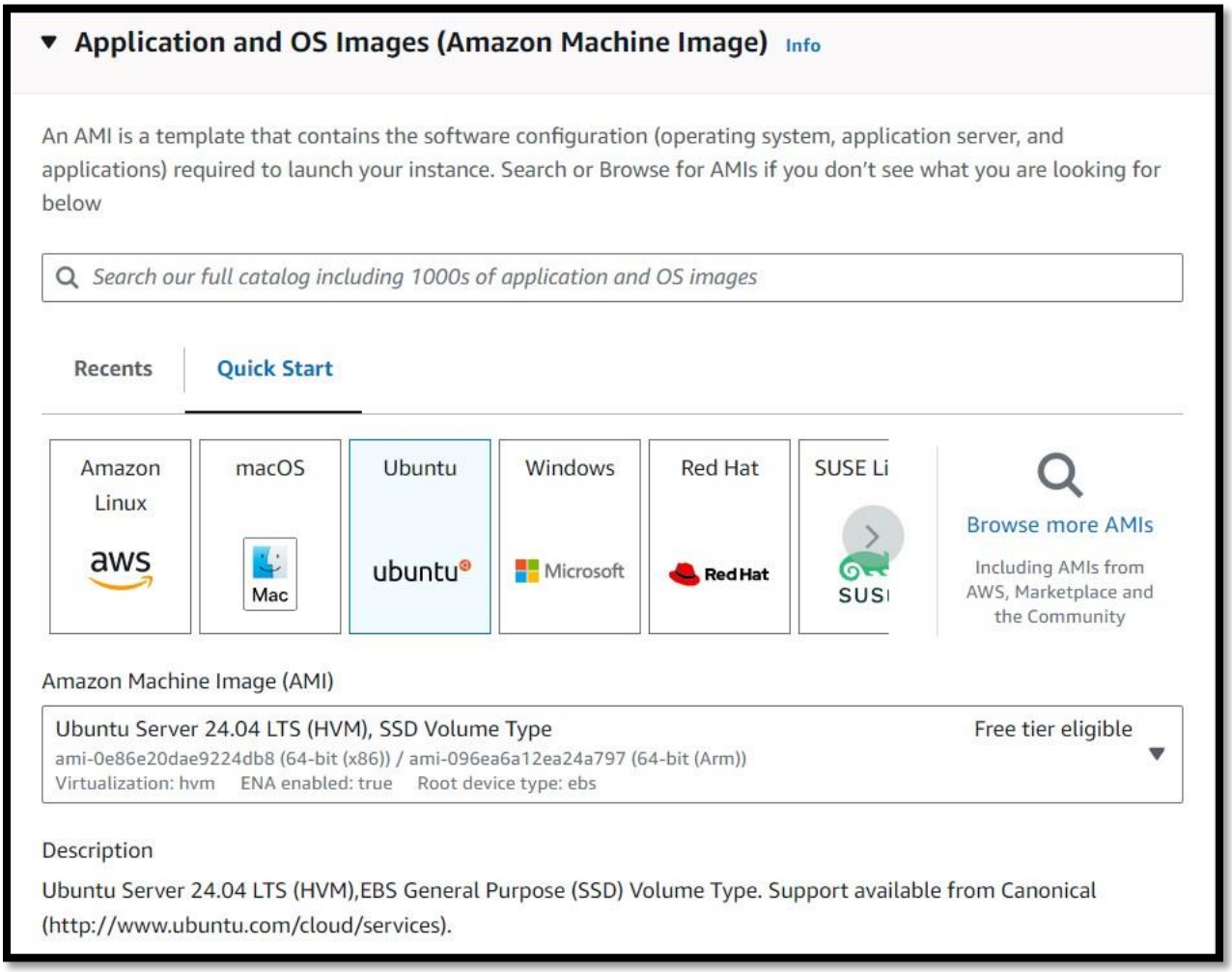
Node:

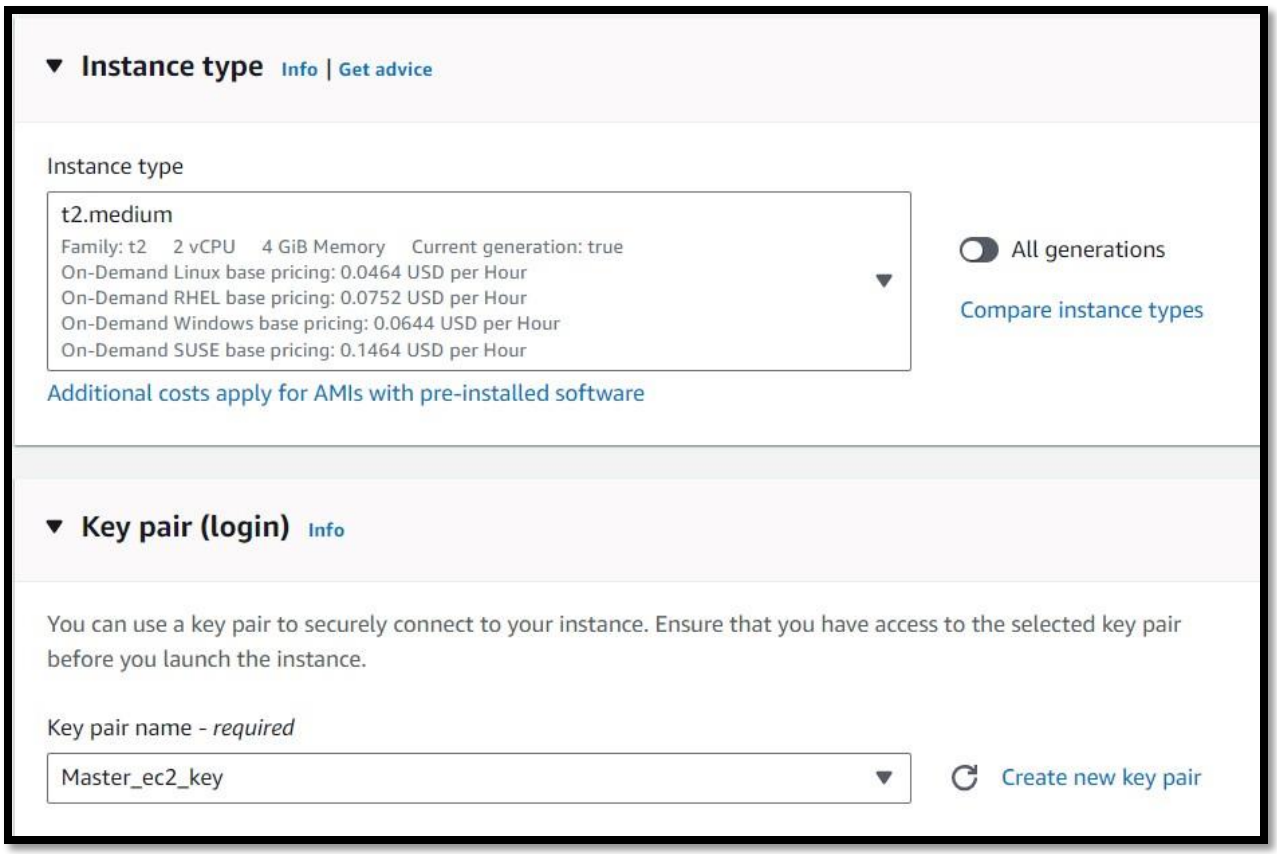


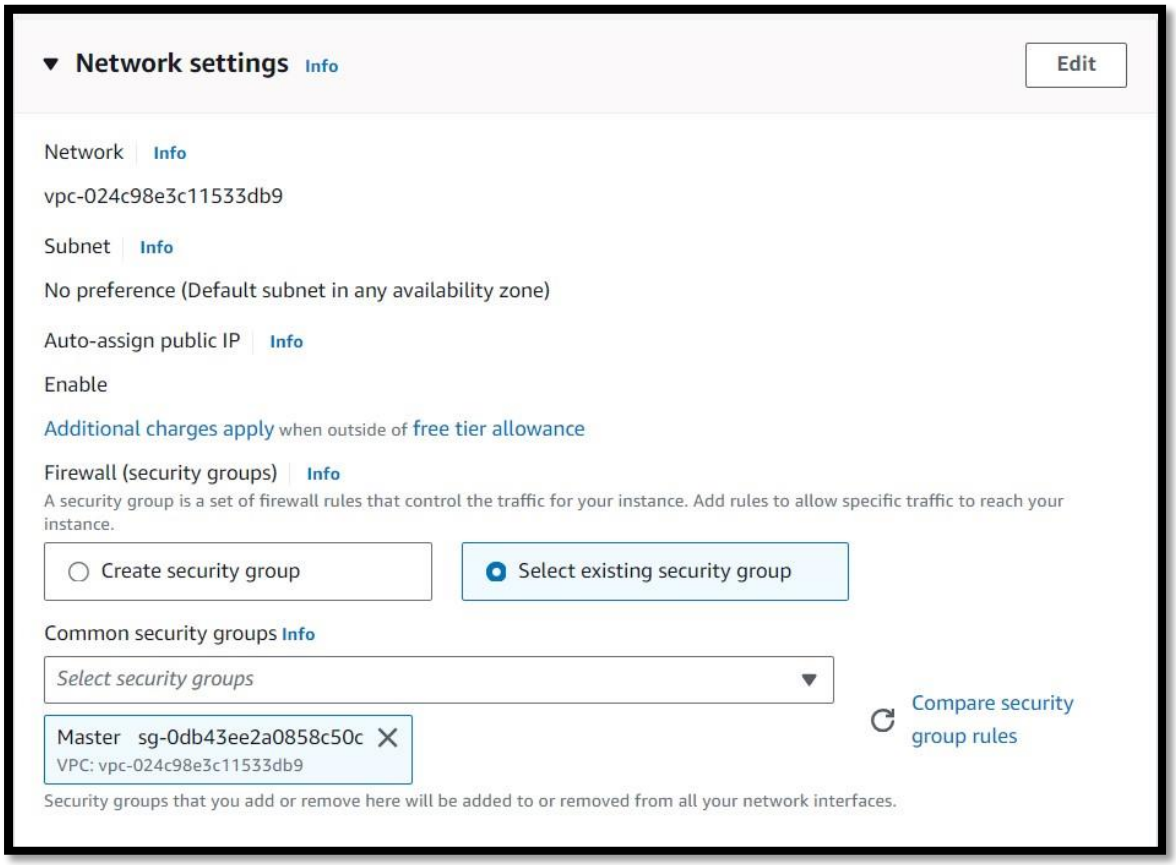
Step 2: Log in to your AWS Academy/personal account and launch 3 new Ec2 Instances(1 for Master and 2 for Node).Select Ubuntu as AMI and t2.medium as Instance Type and create a key of type RSA with .pem extension and move the downloaded key to the new folder.We can use 2 Different keys, 1 for Master and 1 for Node. Also Select Security Groups from the existing.

Master:

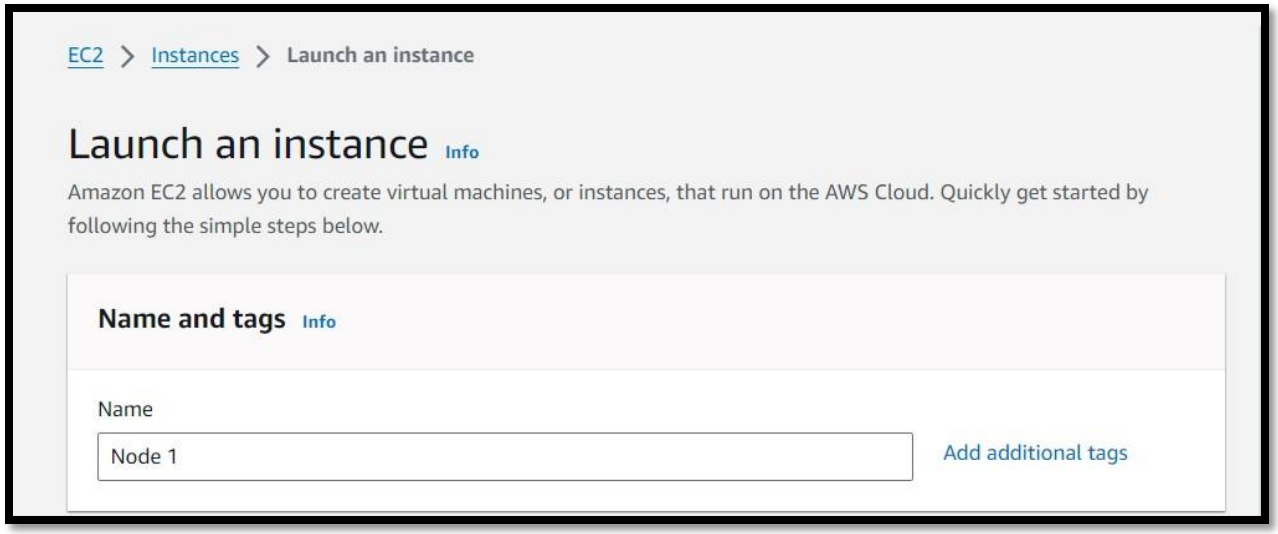

EC2 > Instances > Launch an instance
Launch an instance Info
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.
Name and tags Info
Name
Master
Add additional tags

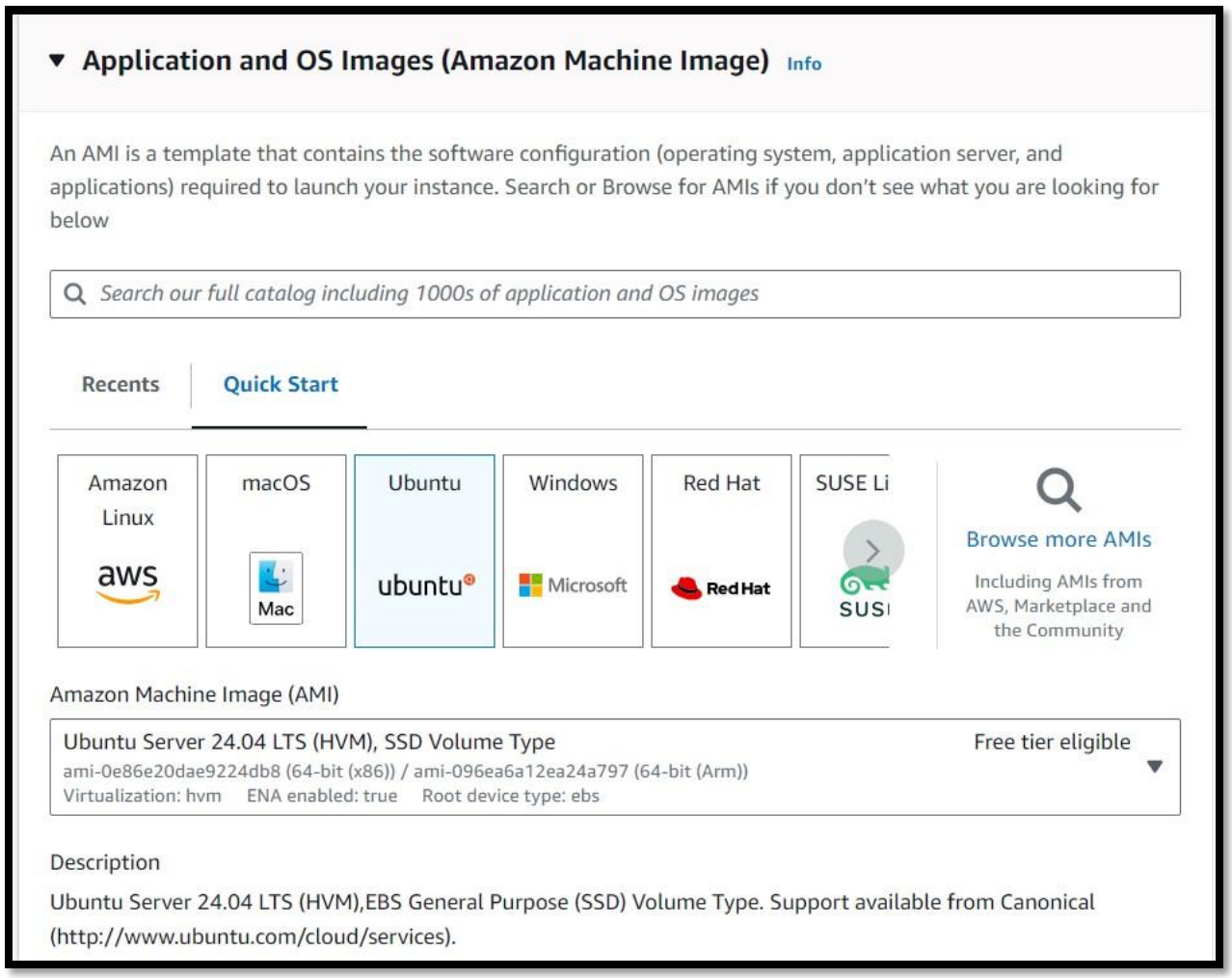

Application and OS Images (Amazon Machine Image) Info
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below
Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Li
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
Free tier eligible
ami-0e86e20dae9224db8 (64-bit (x86)) / ami-096ea6a12ea24a797 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs
Description
Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

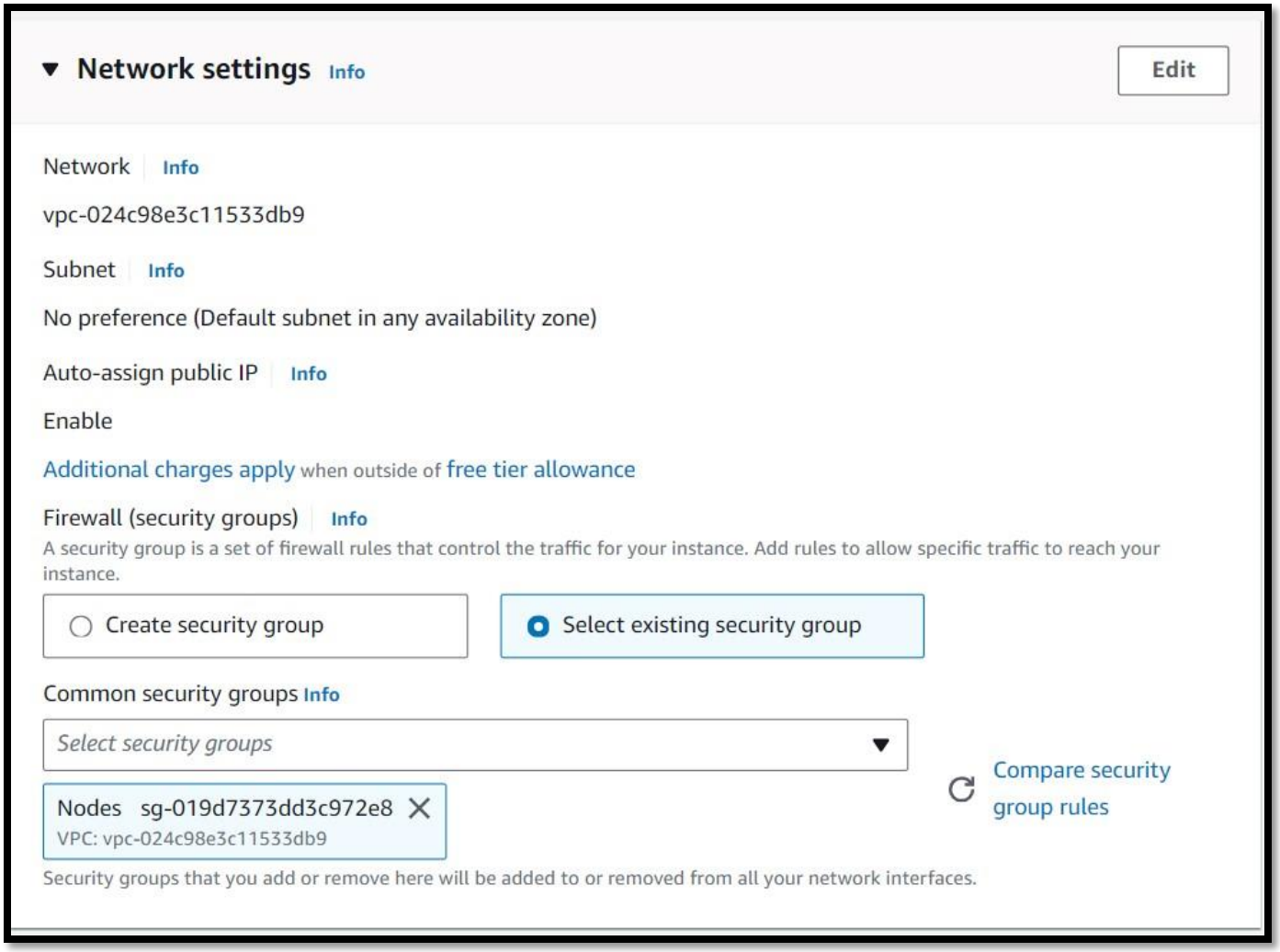

Instance type Info | Get advice
Instance type
t2.medium
Family: t2 2 vCPU 4 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0464 USD per Hour
On-Demand RHEL base pricing: 0.0752 USD per Hour
On-Demand Windows base pricing: 0.0644 USD per Hour
On-Demand SUSE base pricing: 0.1464 USD per Hour
All generations
Compare instance types
Additional costs apply for AMIs with pre-installed software
Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name - required
Master_ec2_key
Create new key pair

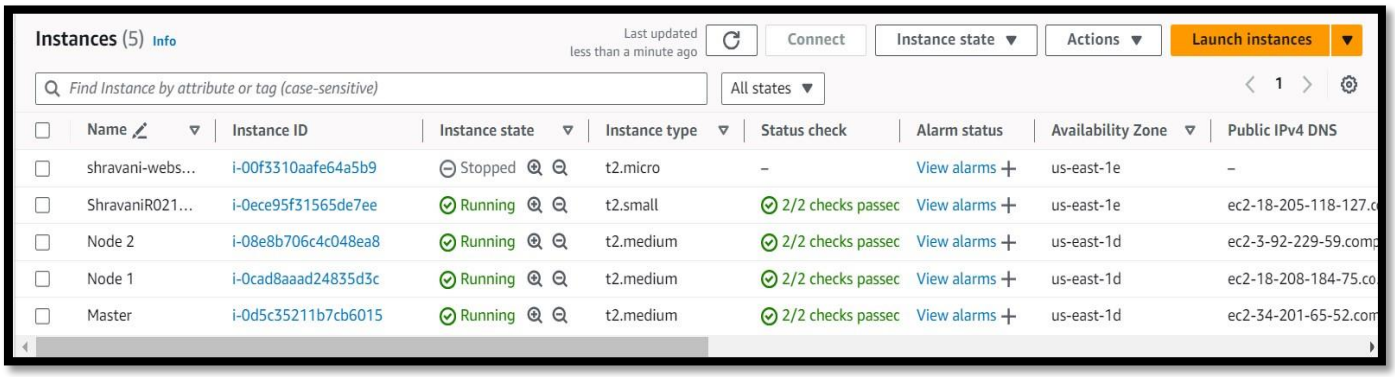

Network settings Info
Edit
Network Info
vpc-024c98e3c11533db9
Subnet Info
No preference (Default subnet in any availability zone)
Auto-assign public IP Info
Enable
Additional charges apply when outside of free tier allowance
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
Common security groups Info
Select security groups
Master sg-0db43ee2a0858c50c
VPC: vpc-024c98e3c11533db9
Compare security group rules
Security groups that you add or remove here will be added to or removed from all your network interfaces.

Node:


EC2 > Instances > Launch an instance
Launch an instance Info
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.
Name and tags Info
Name
Node 1
Add additional tags

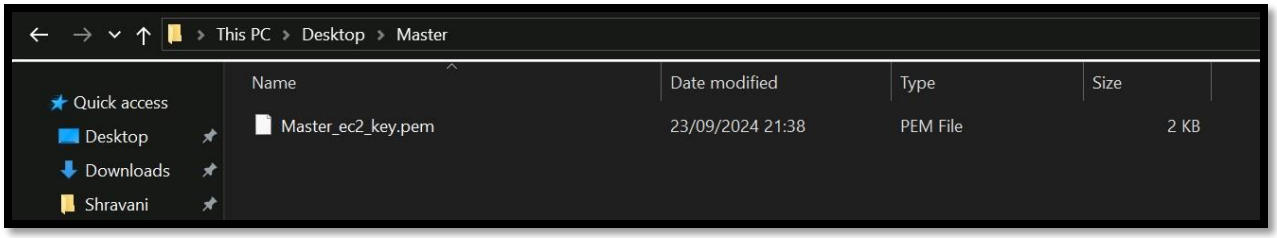

Application and OS Images (Amazon Machine Image) Info
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below
Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Li
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
Free tier eligible
ami-0e86e20dae9224db8 (64-bit (x86)) / ami-096ea6a12ea24a797 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs
Description
Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).



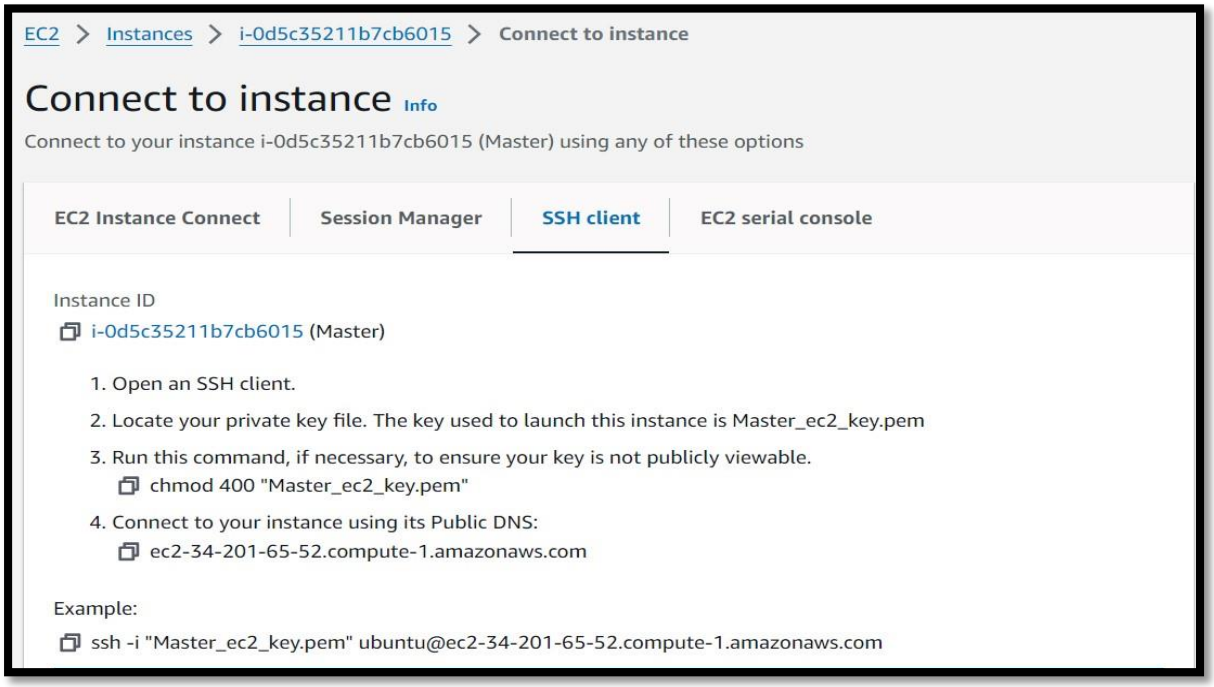


Step 3: Connect the instance and navigate to SSH client and copy the example command. Now open the folder in the terminal 3 times for Master, Node1 & Node 2 where our .pem key is stored and paste the Example command from ssh client (starting with ssh -i .....) in the terminal.

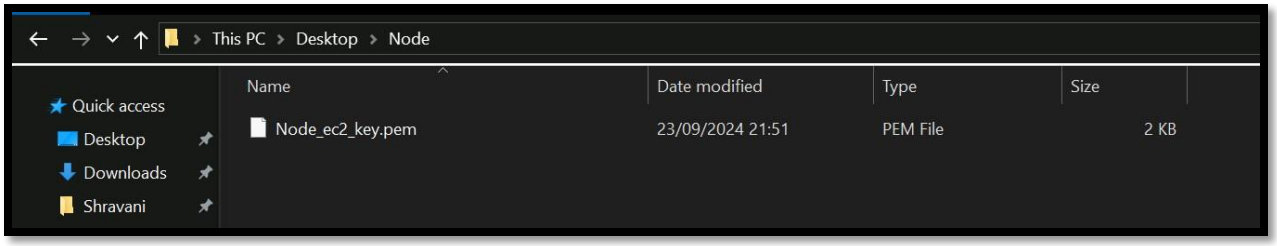
Downloaded Key:



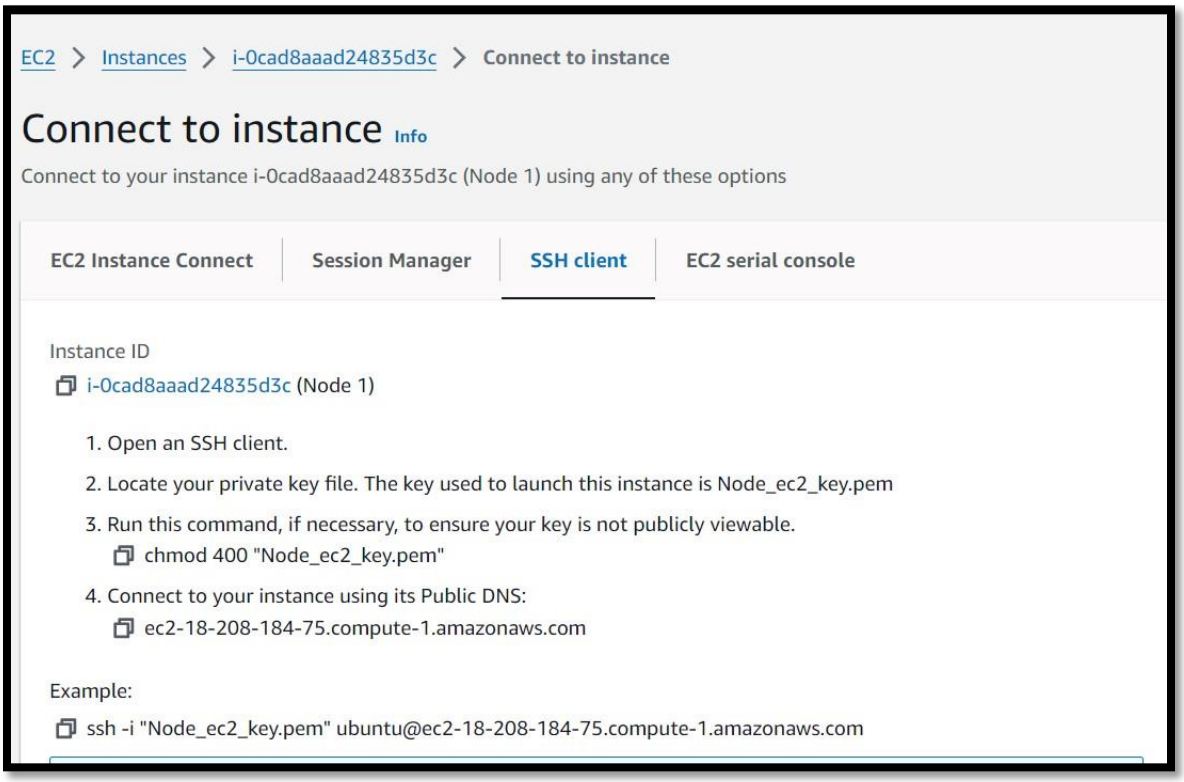
Master:




```
C:\Users\Shravani\Desktop\Master>ssh -i "Master_ec2_key.pem" ubuntu@ec2-34-201-65-52.compute-1.amazonaws.com
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Mon Sep 23 16:43:43 UTC 2024

  System load:  0.0               Processes:             116
  Usage of /:   22.9% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.84.221
  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
```

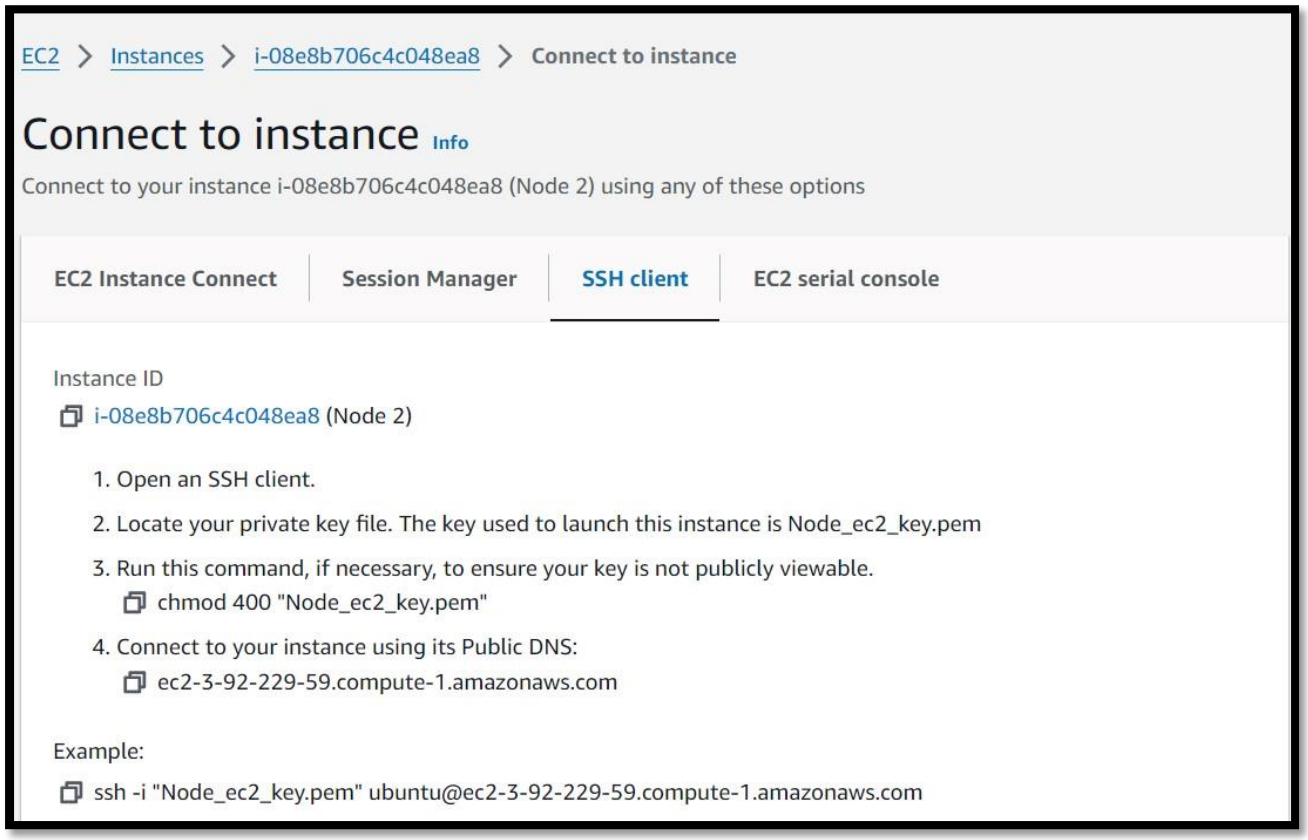

This PC
Desktop
Node
Quick access
Desktop
Downloads
Shravani
Name
Date modified
Type
Size
Node_ec2_key.pem
23/09/2024 21:51
PEM File
2 KB

Node 1:


EC2 > Instances > i-0cad8aaad24835d3c > Connect to instance
Connect to instance Info
Connect to your instance i-0cad8aaad24835d3c (Node 1) using any of these options
EC2 Instance Connect
Session Manager
SSH client
EC2 serial console
Instance ID
i-0cad8aaad24835d3c (Node 1)
1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is Node_ec2_key.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 "Node_ec2_key.pem"
4. Connect to your instance using its Public DNS:
ec2-18-208-184-75.compute-1.amazonaws.com
Example:
ssh -i "Node_ec2_key.pem" ubuntu@ec2-18-208-184-75.compute-1.amazonaws.com

```
C:\Users\Shravani\Desktop\Node>ssh -i "Node_ec2_key.pem" ubuntu@ec2-18-208-184-75.compute-1.amazonaws.com
The authenticity of host 'ec2-18-208-184-75.compute-1.amazonaws.com (18.208.184.75)' can't be established.
ECDSA key fingerprint is SHA256:Mt3R8xcNRQpug+O8YjlPo+4OyaB1xn/43dC9MQA87+A.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-208-184-75.compute-1.amazonaws.com,18.208.184.75' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Mon Sep 23 16:48:32 UTC 2024

  System load:  0.08              Processes:             113
  Usage of /:   22.7% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.95.119
  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
```

Node 2:



```
C:\Users\Shravani\Desktop\Node>ssh -i "Node_ec2_key.pem" ubuntu@ec2-3-92-229-59.compute-1.amazonaws.com
The authenticity of host 'ec2-3-92-229-59.compute-1.amazonaws.com (64:ff9b::35c:e53b)' can't be established.
ECDSA key fingerprint is SHA256:jKzi3rD9OgtRdm2AJ5oS4Ndayn4cxMLRUQGQVXEMsck.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-92-229-59.compute-1.amazonaws.com,64:ff9b::35c:e53b' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Mon Sep 23 16:50:15 UTC 2024

  System load:  0.0               Processes:             114
  Usage of /:   22.7% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.80.164
  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
```

Step 4: Run on Master,Node 1,and Node 2 the below commands to install and setup Docker in Master, Node1, and Node2.

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

$(lsb_release -cs) stable"

```
ubuntu@ip-172-31-84-221:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
ubuntu@ip-172-31-84-221:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee/etc/apt/
sudo: tee/etc/apt/trusted.gpg.d/docker.gpg: command not found
ubuntu@ip-172-31-84-221:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee
gpg.d/docker.gpg > /dev/null-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFit2ioBEADhWpZ8/wvZ6hUTiXOwQHXMAlaFHcPH9hAtr4F1y2+OYdbtMuth
lqqwp028AqyY+PRfVMtSYMbjuQuu5byyKR01BbqYhuS3jtqQmljZ/bJvXqnmiVXh
38UuLa+z077PxyxQhu5BbqntTPQMfiyqEiU+BKbq2WmANUKQf+1AmZY/IruOXbnq
L4C1+gJ8vfmXQt99npCaxEjaNRVYfOS8QcixNzHUYnb6emjlANyEVlZzeqo7XKl7
UrwV5inawTSzWNvtjEjj4nJL8NsLwscpLPQUhTQ+7BbQXAwAmeHCUTQIvvWXqw0N
cmhh4HgeQscQHYgOJjjDVfoY5MucvglbIgCqfzAHW9jxmRL4qbMZj+b1XoePEtht
ku4bIQN1X5P07fNWzlgaRL5Z4POXDDZTlIQ/El58j9kp4bnWRCJW0lya+f8ocodo
vZZ+Doi+fy4D5ZGrL4XEcIQP/Lv5uFyf+kQtl/94VFYVJOleAv8W92KdgDkhTcTD
G7c0tIkVEKNUq48b3aQ64NOZQW7fVjfoKwEZdOqPE72Pa45jrZzvUFxSpdiNk2tZ
XYukHjlxxEgBdC/J3cMMNRE1F4NCA3ApfV1Y7/hTeOnmDuDYwr9/obA8t016Yljj
q5rdkywPf4JF8mXUW5eCN1vAFHxeg9ZWemhBtQmGxXnw9M+z6hWwc6ahmwARAQAB
tCtEb2NrZXIgUmVsZWFzZSAoQ0UgZGViKSA8ZG9ja2VyQGRvY2tlci5jb20+iQI3
```

```
Get:43 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [113 kB]
Get:44 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:45 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [10.1 kB]
Get:46 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [353 kB]
Get:47 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [68.1 kB]
Get:48 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 c-n-f Metadata [428 B]
Get:49 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [10.9 kB]
Get:50 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [2808 B]
Get:51 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:52 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [344 B]
Fetched 29.1 MB in 4s (7159 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring
key(8) for details.
ubuntu@ip-172-31-84-221:~$
```

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

```
ubuntu@ip-172-31-84-221:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring
key(8) for details.
ubuntu@ip-172-31-84-221:~$
```

- sudo mkdir -p /etc/docker cat <<EOF | sudo tee /etc/docker/daemon.json

  {

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

  }

EOF

```
ubuntu@ip-172-31-84-221:~$ sudo mkdir -p /etc/docker
driver=systemd"]
}
EOFcat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOFubuntu@ip-172-31-84-221:~$ sudo mkdir -p /etc/docker
 tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOFcat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOFubuntu@ip-172-31-84-221:~$
```

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

```
EOFubuntu@ip-172-31-84-221:~$ sudo systemctl enable docker
ctl daemon-reload
sudo systemctl restart dockersudo systemctl daemon-reload
```

Step 5: Run the below command to install Kubernetes.

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

```
ubuntu@ip-172-31-84-221:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o
ngs/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | gpg: missing argument for option "-o"
sudo tee /etc/apt/sources.list.d/kubernetes.list


ubuntu@ip-172-31-84-221:~$ /etc/apt/keyrings/kubernetes-apt-keyring.gpg
-bash: /etc/apt/keyrings/kubernetes-apt-keyring.gpg: No such file or directory
ubuntu@ip-172-31-84-221:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
> https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

```
ubuntu@ip-172-31-84-221:~$ sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 136 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 335 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  cri-tools 1.28.0-1.1 [19.6 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  kubernetes-cni 1.2.0-2.1 [27.6 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  kubelet 1.28.14-2.1 [19.6 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  kubectl 1.28.14-2.1 [10.4 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  kubeadm 1.28.14-2.1 [10.1 MB]
Fetched 87.4 MB in 1s (77.5 MB/s)
```

```
ubuntu@ip-172-31-84-221:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-84-221:~$
```

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

- sudo apt-mark hold kubelet kubeadm kubectl

```
ubuntu@ip-172-31-84-221:~$ sudo apt-get update
Warning: The unit file, source configuration file or drop-ins of apt-news.service changed on disk. Run 'systemctl daemon-reload' to reload u
nits.
Warning: The unit file, source configuration file or drop-ins of esm-cache.service changed on disk. Run 'systemctl daemon-reload' to reload
units.
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Err:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease
  The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 234654DA9A296436
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see t
he DEPRECATION section in apt-key(8) for details.
W: GPG error: https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease: The following signatures co
uldn't be verified because the public key is not available: NO_PUBKEY 234654DA9A296436
E: The repository 'https://pkgs.k8s.io/core:/stable:/v1.31/deb  InRelease' is not signed.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
```

Err:7 https://packages.cloud.google.com/apt kubernetes-xenial Release 404 Not Found [IP: 64.233.180.139 443]

- sudo rm /etc/apt/sources.list.d/kubernetes.list
- sudo nano /etc/apt/sources.list.d/kubernetes.list
- deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /

```
ubuntu@ip-172-31-84-221:~$ sudo systemctl enable --now kubelet
ubuntu@ip-172-31-84-221:~$ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 136 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 53.1 MB disk space will be freed.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu4.1 [38.6 MB]
Fetched 47.2 MB in 1s (90.1 MB/s)
```

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

```
ubuntu@ip-172-31-84-221:~$ sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

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

[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
  max_recv_message_size = 16777216
  max_send_message_size = 16777216
  tcp_address = ""
  tcp_tls_ca = ""
  tcp_tls_cert = ""
  tcp_tls_key = ""
  uid = 0
```

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

```
ubuntu@ip-172-31-84-221:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-84-221:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-84-221:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Mon 2024-09-23 20:47:25 UTC; 14s ago
       Docs: https://containerd.io
   Main PID: 19202 (containerd)
      Tasks: 7
     Memory: 13.0M (peak: 13.8M)
        CPU: 113ms
     CGroup: /system.slice/containerd.service
             └─19202 /usr/bin/containerd

Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572213616Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572255061Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572281095Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572298184Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572313100Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572322058Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572328397Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572313683Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572786584Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 systemd[1]: Started containerd.service - containerd container runtim
lines 1-21/21 (END)...skipping...
```

- sudo apt-get install -y socat

```
ubuntu@ip-172-31-84-221:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 lib
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 136 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3
Fetched 374 kB in 0s (13.8 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-84-221:~$
```

Step 6: Initialize the Kubecluster .Now Perform this Command only for Master.

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

```
ubuntu@ip-172-31-84-221:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
I0923 20:56:13.230794   19947 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.28
[init] Using Kubernetes version: v1.28.14
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
W0923 20:56:20.561492   19947 checks.go:835] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container r
used by kubeadm. It is recommended that using "registry.k8s.io/pause:3.9" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-84-221 kubernetes kubernetes.default kubernetes.default.s
.local] and IPs [10.96.0.1 172.31.84.221]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-84-221 localhost] and IPs [172.31.84.221 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-84-221 localhost] and IPs [172.31.84.221 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
```

```
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get l
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a N
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the c
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate a
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.84.221:6443 --token yjt10w.maqlf98vcw88kw96 \
        --discovery-token-ca-cert-hash sha256:ffdb051e04077afecd5ea7a5702131537f9aa5c3dd13785ed4442327fb39f9cf
ubuntu@ip-172-31-84-221:~$
```

Copy the kudeadm join any number of worker nodes command to use it later for joining Node 1 and Node 2 with master

sudo kubeadm join 172.31.84.221:6443 --token yjt10w.maqlf98vcw88kw96 \--discovery-token-ca-cert -hash sha256:ffdb051e04077afecd5ea7a5702131537f9aa5c3dd13785ed4442327fb39f9cf

mkdir -p $HOME/.kube

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

```
ubuntu@ip-172-31-84-221:~$ mkdir -p $HOME/.kube
ubuntu@ip-172-31-84-221:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
cp: overwrite '/home/ubuntu/.kube/config'? y
ubuntu@ip-172-31-84-221:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-84-221:~$
```

Step 7: Now Run the command kubectl get nodes to see the nodes before executing Join command on nodes.

```
ubuntu@ip-172-31-84-221:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-84-221   NotReady   control-plane   8m27s   v1.28.14
ubuntu@ip-172-31-84-221:~$
```

Step 8: Now Run the following command on Node 1 and Node 2 to Join to master.

- sudo kubeadm join 172.31.95.244:6443 --token kzfth2.ug3970lp3qeeieb4\
--discovery-token-ca-cert-hash
sha256:dec27d33f1bfd1dca7a50caa2c05d4cad1d0a18aa88ad75c7ea83f15c529f4ca

Node 1:

```
ubuntu@ip-172-31-95-119:~$ sudo kubeadm join 172.31.84.221:6443 --token yjt10w.maqlf98vcw88kw96 --discovery-token-ca-cert-hash sha256:f
a5c3dd13785ed4442327fb39f9cf --ignore-preflight-errors=FileContent--proc-sys-net-bridge-bridge-nf-call-iptables
[preflight] Running pre-flight checks
        [WARNING FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables does not exist
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

ubuntu@ip-172-31-95-119:~$
```

Node 2:

```
ubuntu@ip-172-31-80-164:~$ sudo kubeadm join 172.31.84.221:6443 --token yjt10w.maqlf98vcw88kw96 --discove
a5c3dd13785ed4442327fb39f9cf --ignore-preflight-errors=FileContent--proc-sys-net-bridge-bridge-nf-call-ip
[preflight] Running pre-flight checks
        [WARNING FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-n
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o y
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Step 9: Now Run the command kubectl get nodes to see the nodes after executing Join command on nodes.

```
ubuntu@ip-172-31-84-221:~$ kubectl get nodes
NAME                STATUS     ROLES           AGE     VERSION
ip-172-31-80-164    NotReady   <none>          16s     v1.28.14
ip-172-31-84-221    NotReady   control-plane   30m     v1.28.14
ip-172-31-95-119    NotReady   <none>          6m43s   v1.28.14
ubuntu@ip-172-31-84-221:~$
```

Step 10: Since Status is NotReady we have to add a network plugin. And also we have to give the name to the nodes.

- kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

```
ubuntu@ip-172-31-84-221:~$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
ubuntu@ip-172-31-84-221:~$
```

- sudo systemctl status kubelet

```
ubuntu@ip-172-31-84-221:~$ sudo systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
     Loaded: loaded (/usr/lib/systemd/system/kubelet.service; enabled; preset: enabled)
    Drop-In: /usr/lib/systemd/system/kubelet.service.d
             └─10-kubeadm.conf
     Active: active (running) since Mon 2024-09-23 20:56:33 UTC; 32min ago
       Docs: https://kubernetes.io/docs/
   Main PID: 20621 (kubelet)
      Tasks: 10 (limit: 4676)
     Memory: 38.0M (peak: 38.5M)
        CPU: 25.017s
     CGroup: /system.slice/kubelet.service
             └─20621 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kube

Sep 23 21:29:20 ip-172-31-84-221 kubelet[20621]:  > pod="kube-system/calico-kube-controller
Sep 23 21:29:20 ip-172-31-84-221 kubelet[20621]: E0923 21:29:20.385530   20621 remote_runti
Sep 23 21:29:20 ip-172-31-84-221 kubelet[20621]:         rpc error: code = Unknown desc = f
Sep 23 21:29:20 ip-172-31-84-221 kubelet[20621]:         : unknown
Sep 23 21:29:20 ip-172-31-84-221 kubelet[20621]:  > podSandboxID="0ac51787037fdb883ecf57aad
Sep 23 21:29:20 ip-172-31-84-221 kubelet[20621]: E0923 21:29:20.385606   20621 kuberuntime_
Sep 23 21:29:20 ip-172-31-84-221 kubelet[20621]: E0923 21:29:20.505019   20621 kubelet.go:1
Sep 23 21:29:21 ip-172-31-84-221 kubelet[20621]: I0923 21:29:21.388923   20621 pod_startup_
Sep 23 21:29:26 ip-172-31-84-221 kubelet[20621]: I0923 21:29:26.828223   20621 scope.go:117
Sep 23 21:29:26 ip-172-31-84-221 kubelet[20621]: E0923 21:29:26.828431   20621 pod_workers.
lines 1-23/23 (END)
```

- Now Run command kubectl get nodes -o wide we can see Status is ready.

```
ubuntu@ip-172-31-84-221:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-80-164  Ready    <none>          3m29s   v1.28.14  172.31.80.164   <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ip-172-31-84-221  Ready    control-plane   33m     v1.28.14  172.31.84.221   <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ip-172-31-95-119  Ready    <none>          9m56s   v1.28.14  172.31.95.119   <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-84-221:~$
```

The Roles are not yet assigned to the Nodes

```
ubuntu@ip-172-31-84-221:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-80-164    Ready    <none>          4m14s   v1.28.14
ip-172-31-84-221    Ready    control-plane   34m     v1.28.14
ip-172-31-95-119    Ready    <none>          10m     v1.28.14
ubuntu@ip-172-31-84-221:~$
```

- Rename to Node 1: kubectl label node ip-172-31-28-117 kubernetes.io/role=Node1
- Rename to Node 2: kubectl label node ip-172-31-18-135 kubernetes.io/role=Node2

```
ubuntu@ip-172-31-84-221:~$ kubectl label node ip-172-31-80-164 kubernetes.io/role=Node1
node/ip-172-31-80-164 labeled
ubuntu@ip-172-31-84-221:~$ kubectl label node ip-172-31-95-119 kubernetes.io/role=Node2
node/ip-172-31-95-119 labeled
ubuntu@ip-172-31-84-221:~$
```

- Run kubectl get nodes to check if roles are assigned now to the nodes

```
ubuntu@ip-172-31-84-221:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE    VERSION
ip-172-31-80-164    Ready    Node1           8m2s   v1.28.14
ip-172-31-84-221    Ready    control-plane   38m    v1.28.14
ip-172-31-95-119    Ready    Node2           14m    v1.28.14
ubuntu@ip-172-31-84-221:~$
```

CONCLUSION: In this experiment, we successfully set up a Kubernetes cluster with one master and two worker nodes on AWS EC2 instances. After installing Docker, Kubernetes tools (kubelet, kubeadm, kubectl), and containerd on all nodes, the master node was initialized and the worker nodes were joined to the cluster. Initially, the nodes were in the NotReady state, which was resolved by installing the Calico network plugin. We also labeled the nodes with appropriate roles (control-plane and worker). The cluster became fully functional with all nodes in the Ready state, demonstrating the successful configuration and orchestration of Kubernetes.

AKRUTI DABAS/ D15A/ 11

EXPERIMENT NO. 4

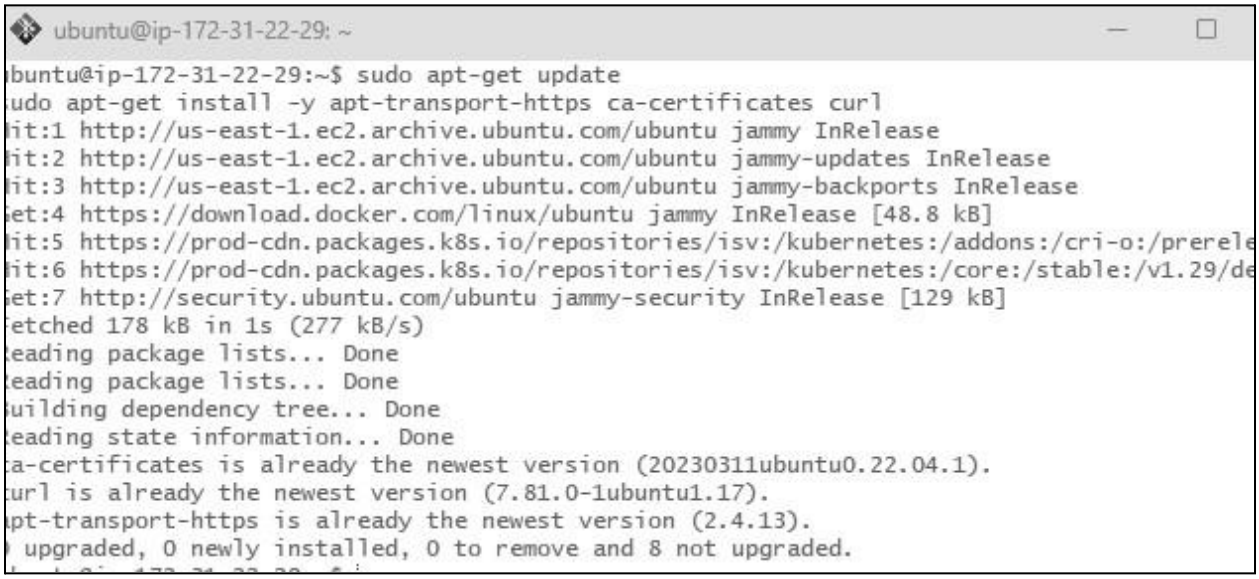
AIM

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

IMPLEMENTATION

Install prerequisites: sudo apt-get update sudo apt-get install -y apt-transport-https ca-

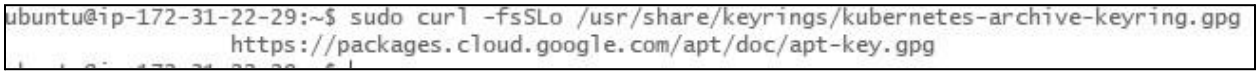
certificates curl



Add the GPG key for Kubernetes:

sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg

1. Add the Kubernetes repository:



AKRUTI DABAS/ D15A/ 11

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-focal main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-22-29:~$  echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring
.gpg]  https://apt.kubernetes.io/ kubernetes-focal main" | sudo tee /etc/apt/sources.list.d/ku
bernetes.list
deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]  https://apt.kubernetes.io/
 kubernetes-focal main
```

-

```
ubuntu@ip-172-31-22-29:~$ sudo apt-get update
sudo apt-get install -y kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri-o:/prerelease:/main/deb  InRelease
Ign:7 https://packages.cloud.google.com/apt kubernetes-focal InRelease
Err:8 https://packages.cloud.google.com/apt kubernetes-focal Release
  404  Not Found [IP: 172.253.62.138 443]
Reading package lists... Done
E: The repository 'https://apt.kubernetes.io kubernetes-focal Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
kubectl is already the newest version (1.29.0-1.1).
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
```

Verify the installation(extra):

kubectl version --client

```
ubuntu@ip-172-31-22-29:~$ kubectl version --client
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
```

AKRUTI DABAS/ D15A/ 11

Set up Kubernetes Cluster

```
ubuntu@ip-172-31-45-227:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-43-211   Ready    <none>          50s     v1.29.0
ip-172-31-45-13    Ready    <none>          34s     v1.29.0
ip-172-31-45-227   Ready    control-plane   5m17s   v1.29.0
ubuntu@ip-172-31-45-227:~$
```

```
ubuntu@ip-172-31-45-227:~$ nano nginx-deployment.yaml
```

AKRUTI DABAS/ D15A/ 11

ubuntu@ip-172-31-45-227: ~

```
 GNU nano 6.2                          nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.21.3
        ports:
        - containerPort: 80
```

Create the Service YAML File

Create the YAML File: Create another file named nginx-service.yaml

```
ubuntu@ip-172-31-45-227:~$ nano nginx-service.yaml
```

ubuntu@ip-172-31-45-227: ~

```
 GNU nano 6.2                          nginx-service.yaml *
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

AKRUTI DABAS/ D15A/ 11

Apply the YAML Files

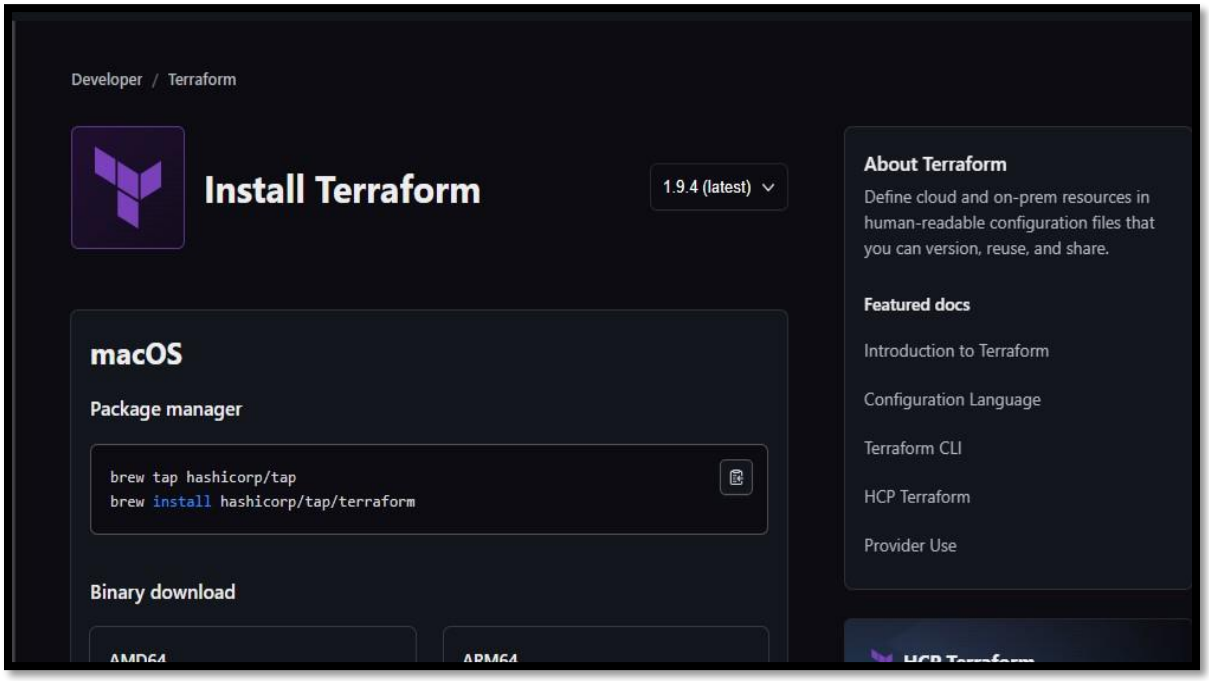
```
ubuntu@ip-172-31-45-227:~$ kubectl apply -f nginx-deployment.yaml
kubectl apply -f nginx-service.yaml
deployment.apps/nginx-deployment created
service/nginx-service created
```

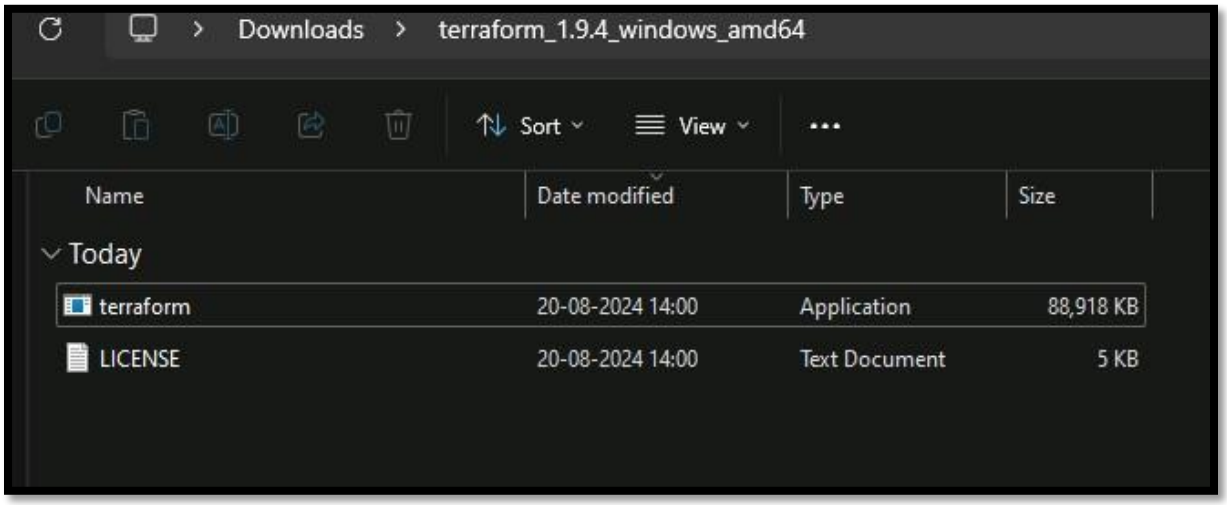
Verify the Deployment

```
ubuntu@ip-172-31-45-227:~$ kubectl get deployments
kubectl get pods
kubectl get services
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2            2           40s
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6b4d6fdbf-6k84m    1/1     Running   0          40s
nginx-deployment-6b4d6fdbf-9d8j6    1/1     Running   0          40s
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP      10.96.0.1        <none>        443/TCP        40m
nginx-service   LoadBalancer   10.106.182.152   <pending>     80:32317/TCP   40s
```

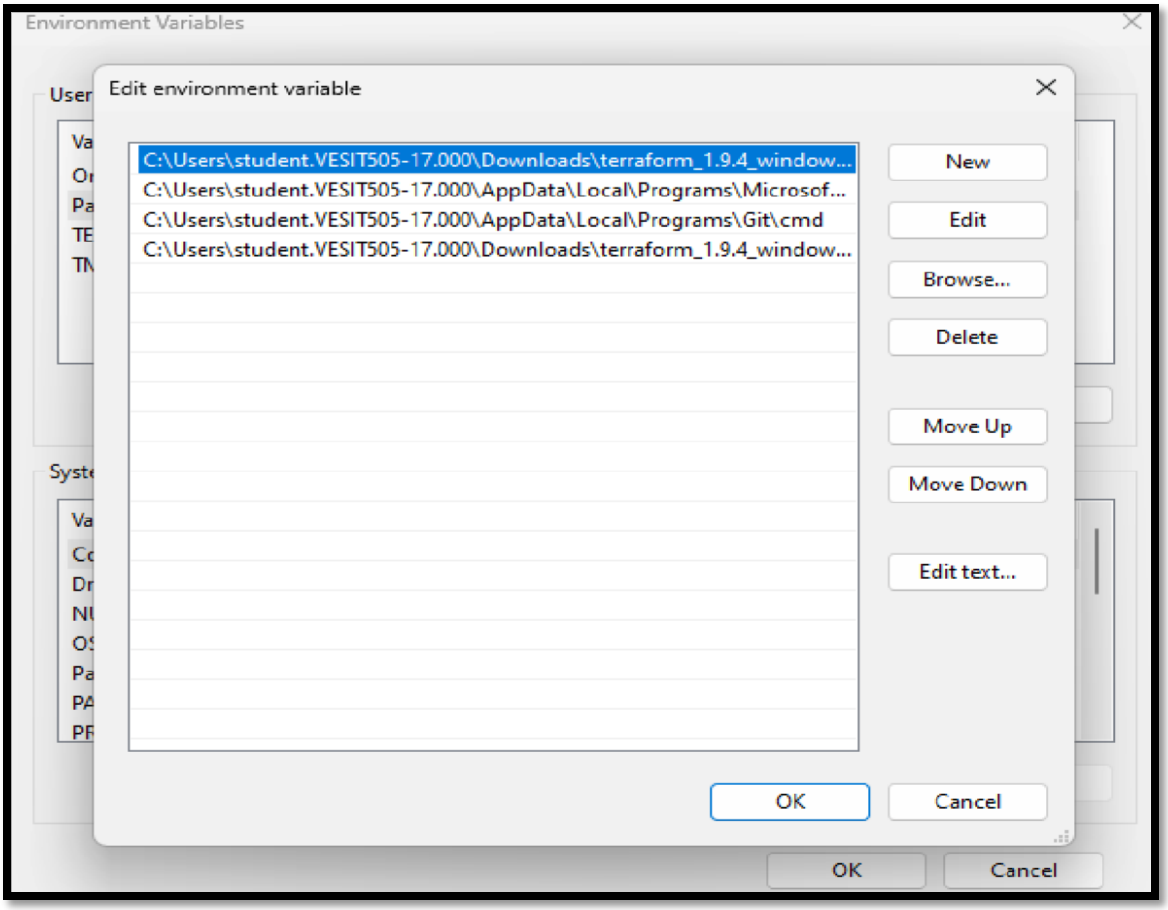
EXPERIMENT NO. 5

AIM: Installation and configuration of terraform on Windows





AKRUTI DABAS/ D10A/ 11



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

PS C:\Users\student.VESIT505-17.000\Downloads\terraform_1.9.4_windows_amd64> terraform
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
  import        Associate existing infrastructure with a Terraform resource
  login         Obtain and save credentials for a remote host
  logout        Remove locally-stored credentials for a remote host
  metadata      Metadata related commands
  output        Show output values from your root module
  providers     Show the providers required for this configuration
  refresh       Update the state to match remote systems
  show          Show the current state or a saved plan
  state         Advanced state management
  taint         Mark a resource instance as not fully functional
  test          Execute integration tests for Terraform modules
  untaint       Remove the 'tainted' state from a resource instance
```

```
PS C:\Users\student.VESIT505-17.000\Downloads\terraform_1.9.4_windows_amd64> terraform --version
Terraform v1.9.4
on windows_amd64
PS C:\Users\student.VESIT505-17.000\Downloads\terraform_1.9.4_windows_amd64>
```

AKRUTI DABAS/ D15A/ 11

EXPERIMENT NO. 6

AIM

To Build, change, and destroy AWS infrastructure Using Terraform.

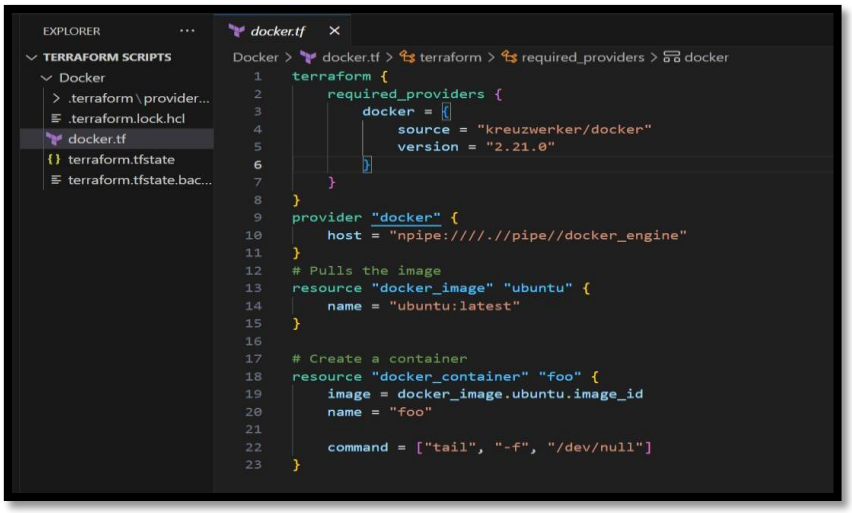
THEORY

Terraform is a powerful tool for managing and automating AWS infrastructure. It allows you to define and provision infrastructure as code (IaC). Here's a basic guide on using Terraform to build, change, and destroy AWS infrastructure.

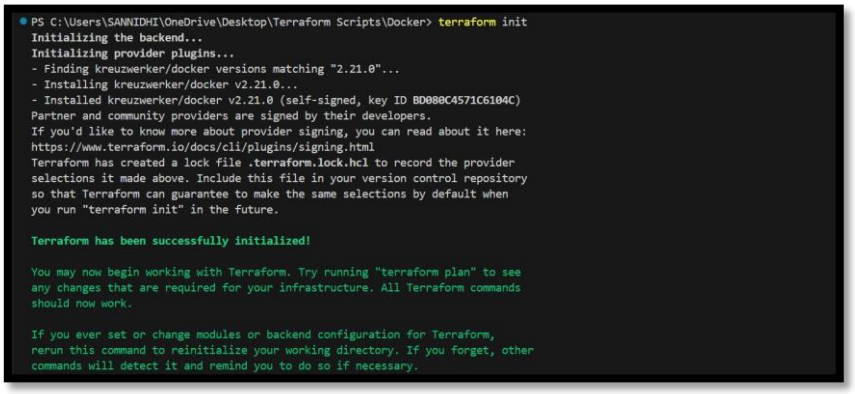
Using Terraform to manage AWS infrastructure is a powerful approach that allows you to define, build, and manage your cloud resources using code. Terraform, an open-source Infrastructure as Code (IaC) tool by HashiCorp, helps automate the provisioning and management of cloud resources consistently and repeatedly.

IMPLEMENTATION

create a new folder named 'Docker' in the 'TerraformScripts' folder. Then create a new docker.tf file

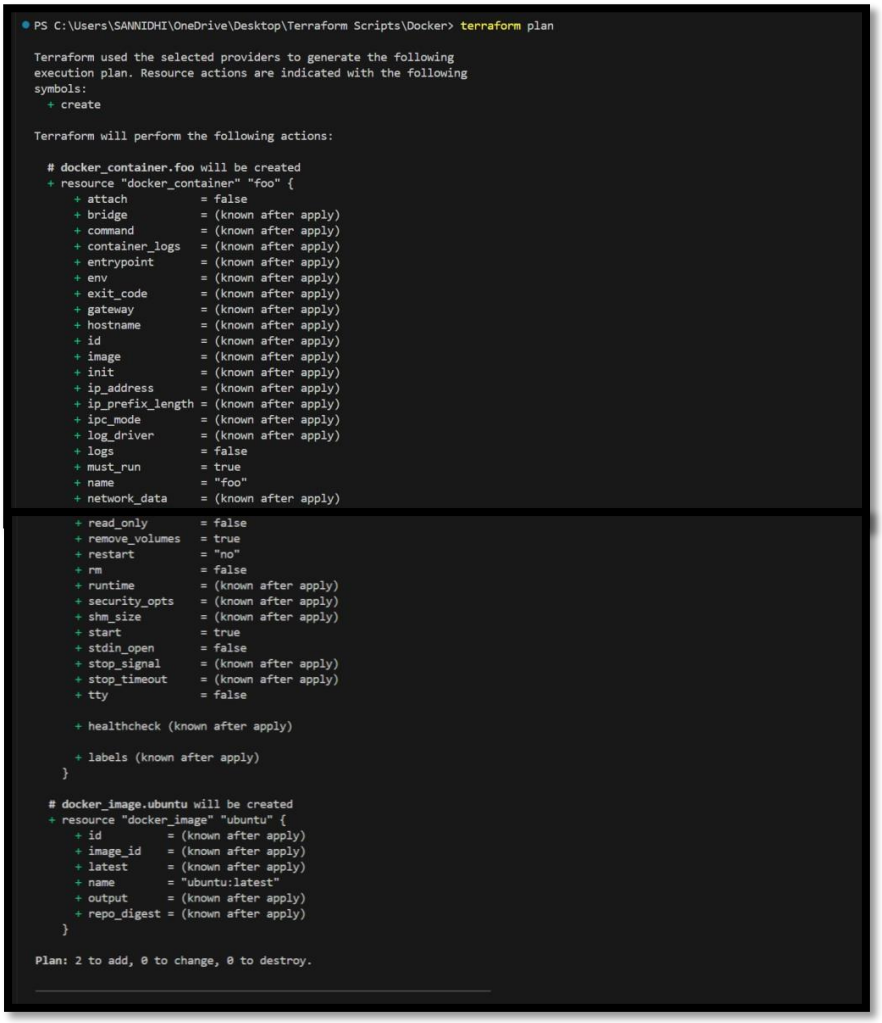


Execute Terraform Init command to initialize the resources



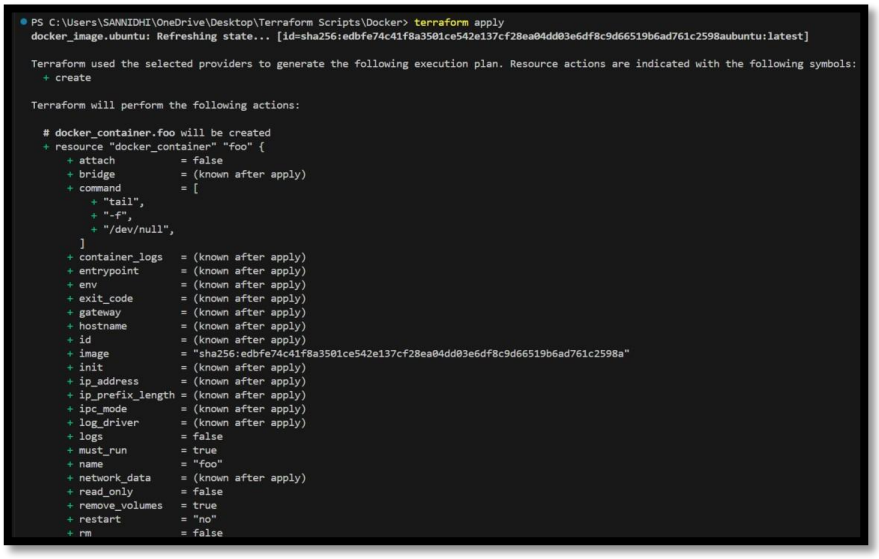
AKRUTI DABAS/ D15A/ 11

Execute Terraform plan to see the available resources



```
PS C:\Users\SANNIDHI\OneDrive\Desktop\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.
```

Execute Terraform apply to apply the configuration, which will automatically create and run the Ubuntu Linux container based on our configuration.



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

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

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = [
          + "tail",
          + "-f",
          + "/dev/null",
        ]
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a"
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
      + name             = "foo"
      + network_data     = (known after apply)
      + read_only        = false
      + remove_volumes   = true
      + restart          = "no"
      + rm               = false
```

AKRUTI DABAS/ D15A/ 11

```
      + rm              = false
      + runtime         = (known after apply)
      + security_opts   = (known after apply)
      + shm_size        = (known after apply)
      + start           = true
      + stdin_open      = false
      + stop_signal     = (known after apply)
      + stop_timeout    = (known after apply)
      + tty             = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

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

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

  Enter a value: yes

docker_container.foo: Creating...
docker_container.foo: Creation complete after 0s [id=d8d3a7916204fe136ac6bd973622991daf1a338aa6bef3f22f7aa6efefb10e98]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\SANNIDHI\OneDrive\Desktop\Terraform Scripts\Docker>
```

Docker images, After Executing Apply step

```
PS C:\Users\SANNIDHI\OneDrive\Desktop\Terraform Scripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   5 weeks ago   78.1MB
PS C:\Users\SANNIDHI\OneDrive\Desktop\Terraform Scripts\Docker>
```

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

```
      + rm              = false
      + runtime         = (known after apply)
      + security_opts   = (known after apply)
      + shm_size        = (known after apply)
      + start           = true
      + stdin_open      = false
      + stop_signal     = (known after apply)
      + stop_timeout    = (known after apply)
      + tty             = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

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

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

  Enter a value: yes

docker_container.foo: Creating...
docker_container.foo: Creation complete after 0s [id=d8d3a7916204fe136ac6bd973622991daf1a338aa6bef3f22f7aa6efefb10e98]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\SANNIDHI\OneDrive\Desktop\Terraform Scripts\Docker>
```

AKRUTI DABAS/ D15A/ 11

```
PS C:\Users\SANNIDHI\OneDrive\Desktop\Terraform Scripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=d8d3a7916204fe136ac6bd973622991daf1a338aa6bef3f22f7aa6efefb10e98]

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

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach           = false -> null
      - command          = [
          - "tail",
          - "-f",
          - "/dev/null",
        ] -> null
      - cpu_shares       = 0 -> null
      - dns              = [] -> null
      - dns_opts         = [] -> null
      - dns_search       = [] -> null
      - entrypoint       = [] -> null
      - env              = [] -> null
      - gateway          = "172.17.0.1" -> null
      - group_add        = [] -> null
      - hostname         = "d8d3a7916204" -> null
      - id               = "d8d3a7916204fe136ac6bd973622991daf1a338aa6bef3f22f7aa6efefb10e98" -> null
      - image            = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - init             = false -> null
      - ip_address       = "172.17.0.2" -> null
      - ip_prefix_length = 16 -> null
      - ipc_mode         = "private" -> null
      - links            = [] -> null
      - log_driver       = "json-file" -> null
      - log_opts         = {} -> null
      - logs             = false -> null
      - max_retry_count  = 0 -> null
      - memory           = 0 -> null
      - memory_swap      = 0 -> null
      - must_run         = true -> null
      - name             = "foo" -> null
      - network_data     = [
          - {
              - gateway                   = "172.17.0.1"
              - global_ipv6_prefix_length = 0
              - ip_address                = "172.17.0.2"
              - ip_prefix_length          = 16
              - network_name              = "bridge"
                # (2 unchanged attributes hidden)
          },
        ]
      - read_only        = false -> null
      - remove_volumes   = true -> null
      - restart          = "no" -> null
      - rm               = false -> null
      - runtime          = "runc" -> null
      - security_opts    = [] -> null
      - shm_size         = 64 -> null
      - start            = true -> null
      - stdin_open       = false -> null
      - stop_timeout     = 0 -> null
      - storage_opts     = {} -> null
      - sysctls          = {} -> null
      - tmpfs            = {} -> null
      - tty              = false -> null
        # (8 unchanged attributes hidden)
    }

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

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

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

  Enter a value: yes

docker_container.foo: Destroying... [id=d8d3a7916204fe136ac6bd973622991daf1a338aa6bef3f22f7aa6efefb10e98]
docker_container.foo: Destruction complete after 1s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
```

Docker images After Executing Destroy step

```
Destroy complete! Resources: 2 destroyed.
PS C:\Users\SANNIDHI\OneDrive\Desktop\Terraform Scripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
PS C:\Users\SANNIDHI\OneDrive\Desktop\Terraform Scripts\Docker>
```

AKRUTI DABAS/ D15A/ 11

EXPERIMENT NO. 7

Static Application Security Testing (SAST)

SAST is a method of security testing that analyzes source code to identify vulnerabilities **without executing the program**. It is also known as **white-box testing**.

SAST Process Breakdown

1. **Code Parsing**
   - The source code is parsed to create an **Abstract Syntax Tree (AST)**, which represents the code structure.
2. **Pattern Matching**
   - The AST is analyzed using predefined rules to detect patterns that may indicate security vulnerabilities.
3. **Data Flow Analysis**
   - This step examines how data moves through the code to identify potential security issues like **SQL Injection** or **Cross-Site Scripting (XSS)**.
4. **Control Flow Analysis**
   - Involves analyzing the paths that the code execution might take to find logical errors or vulnerabilities.
5. **Reporting**
   - The tool generates a report highlighting the vulnerabilities found, their severity, and recommendations for fixing them.

Benefits of SAST

- **Early Detection**
  - Identifies vulnerabilities early in the development lifecycle, reducing the cost and effort required to fix them.
- **Comprehensive Coverage**
  - Can analyze **100% of the codebase**, including all possible execution paths.
- **Automated and Scalable**
  - Suitable for large codebases and can be integrated into **CI/CD pipelines** for continuous monitoring.

SonarQube and SAST

SonarQube is a popular tool that provides static code analysis to detect bugs, code smells, and security vulnerabilities. Here's how SonarQube fits into the SAST process:

---

1. **Integration**
   - SonarQube can be integrated into your **CI/CD pipeline** to automatically analyze code every time it is committed.

2. **Rule Sets**
   - It uses a comprehensive set of rules to detect **security vulnerabilities**, **coding standards violations**, and **code quality issues**.
3. **Detailed Reporting**
   - SonarQube generates detailed reports that help developers understand and fix the identified issues efficiently.
4. **Continuous Feedback**
   - Provides continuous feedback to developers, enabling them to maintain high code quality and security standards throughout the development process.
5. **Customization**
   - Allows customization of rule sets to match the specific needs and standards of your project or organization.

Implementation:

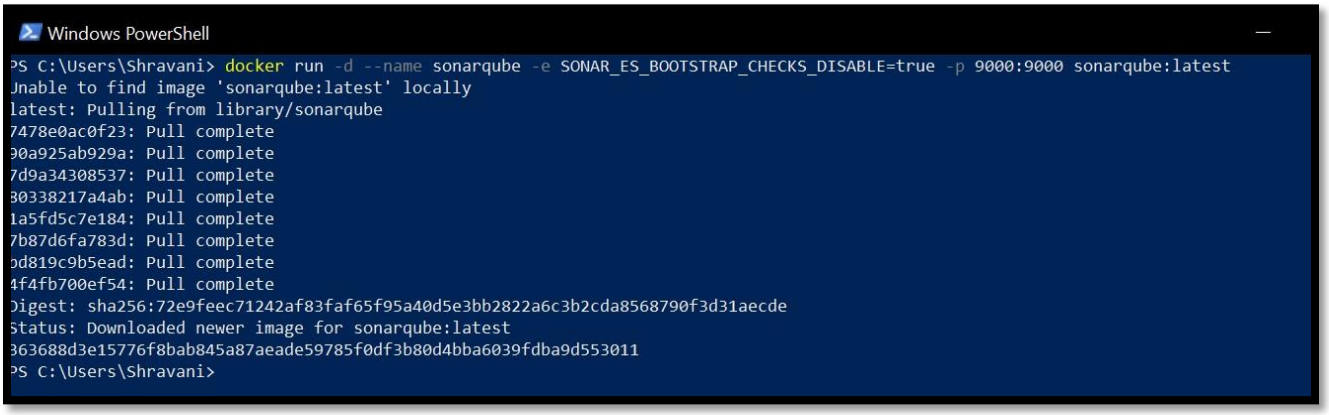
1. Open Jenkins Dashboard

- Access your Jenkins Dashboard by navigating to http://localhost:8080 (or the port you have configured Jenkins to run on).

2. Run SonarQube in a Docker Container

- Open a terminal and run the following command to start SonarQube in a Docker container

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



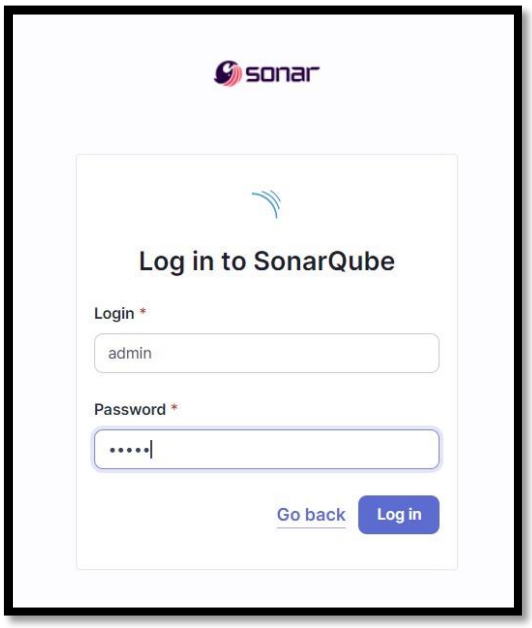
3. Check SonarQube Status

- Once the container is up and running, check the status of SonarQube by navigating to http://localhost:9000

4. Login to SonarQube

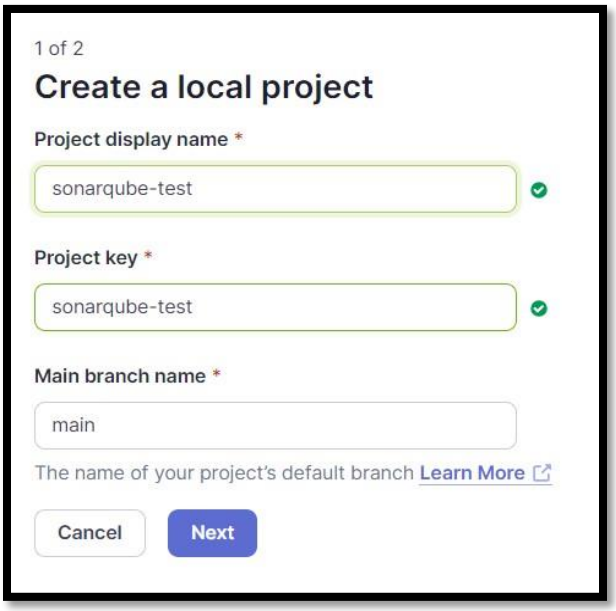
- Use the default credentials to log in:
  - Username: admin

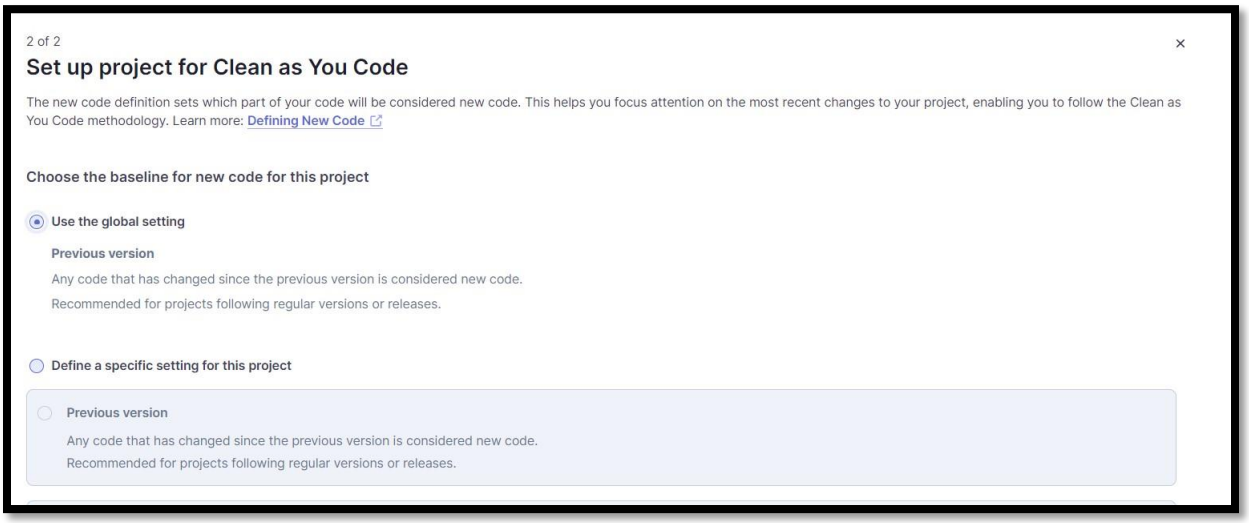
- Password: admin



5. Create a Project in SonarQube

- Create a new project manually in SonarQube and name it sonarqube

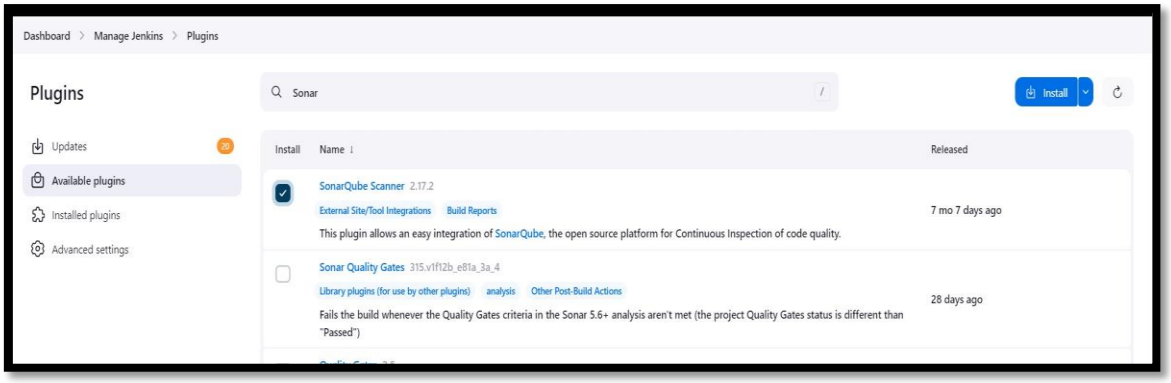


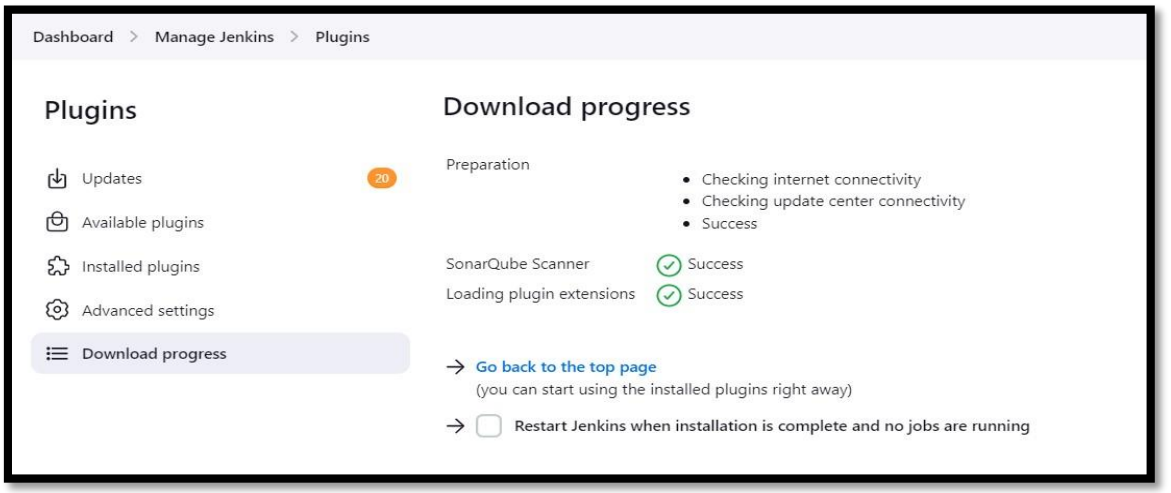


6. Install SonarQube Scanner for Jenkins

● Go back to the Jenkins Dashboard.

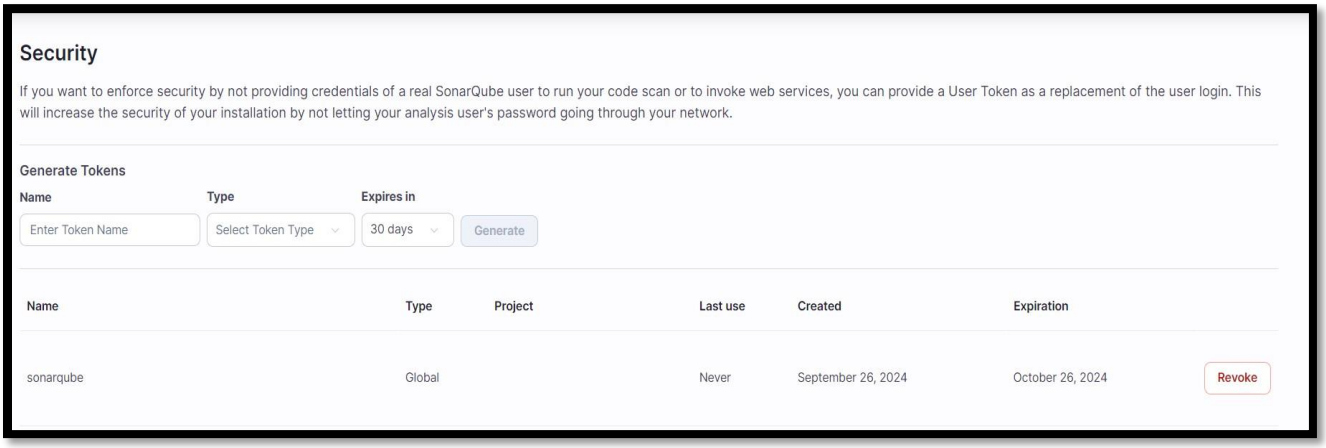
- Navigate to Manage Jenkins > Manage Plugins.
- Search for SonarQube Scanner for Jenkins and install it.





7. Configure SonarQube in Jenkins

- Go to Manage Jenkins > Configure System
- Scroll down to the SonarQube Servers section and enter the required details:
  - **Name**: Any name you prefer.
  - **Server URL**: http://localhost:9000
  - **Server Authentication Token**: (Generate this token in SonarQube under My Account > Security > Generate Tokens).
  - **Add Jenkins**: Select Kind - Secret Text > Secret (Paste Generated Token)



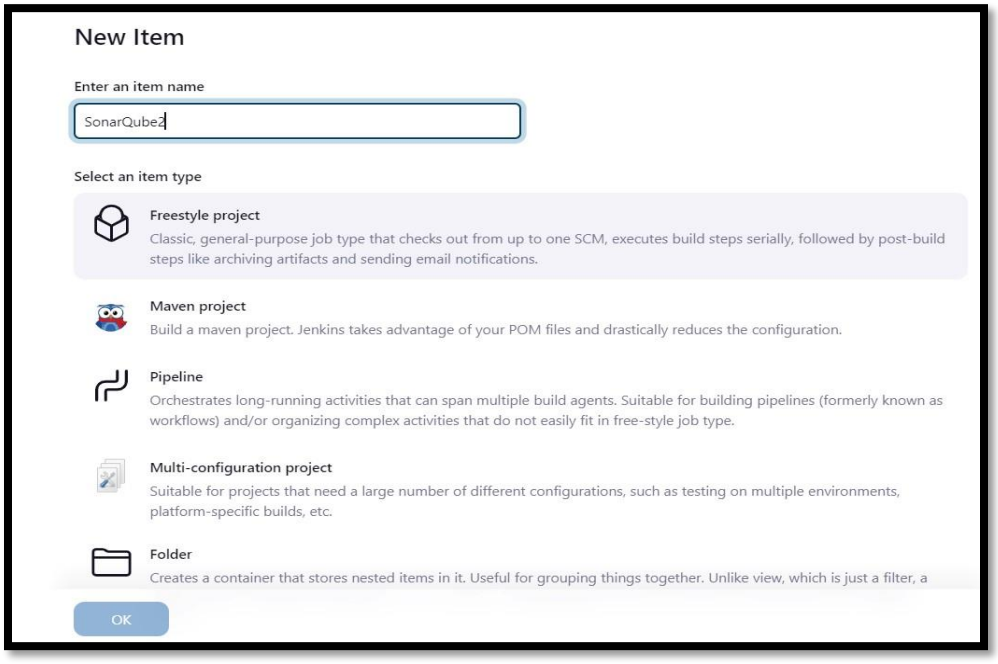
8. Configure SonarQube Scanner in Jenkins ●

Go to Manage Jenkins > Global Tool Configuration.

- Scroll down to SonarQube Scanner.
- Choose the latest version and select Install automatically

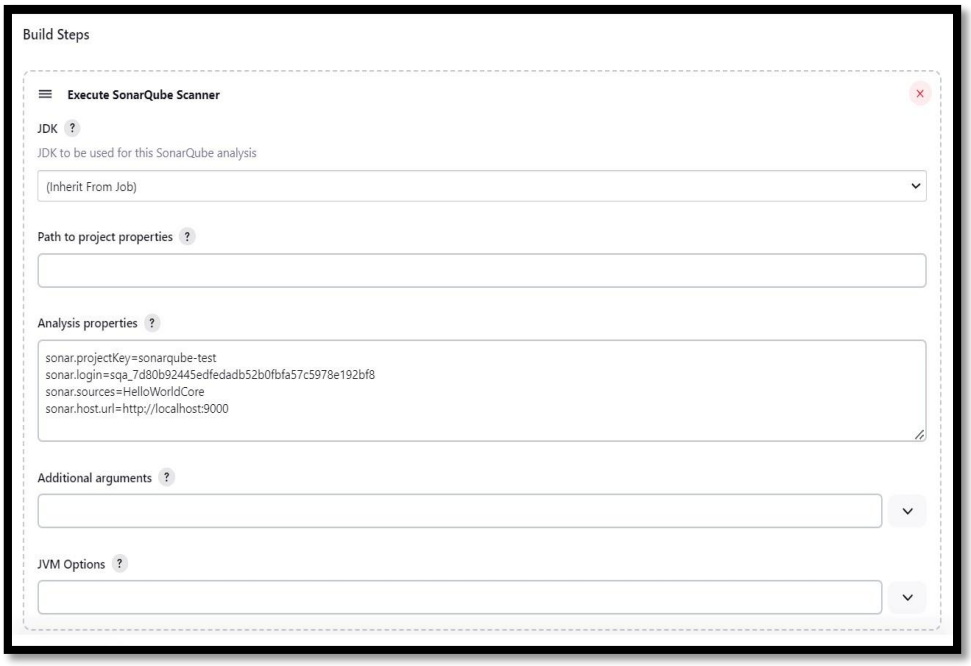
9. Create a New Jenkins Job

- In Jenkins, create a new item and select Freestyle project.
- Under Source Code Management, choose Git and enter the repository URL: https://github.com/shazforiot/MSBuild_firstproject.git



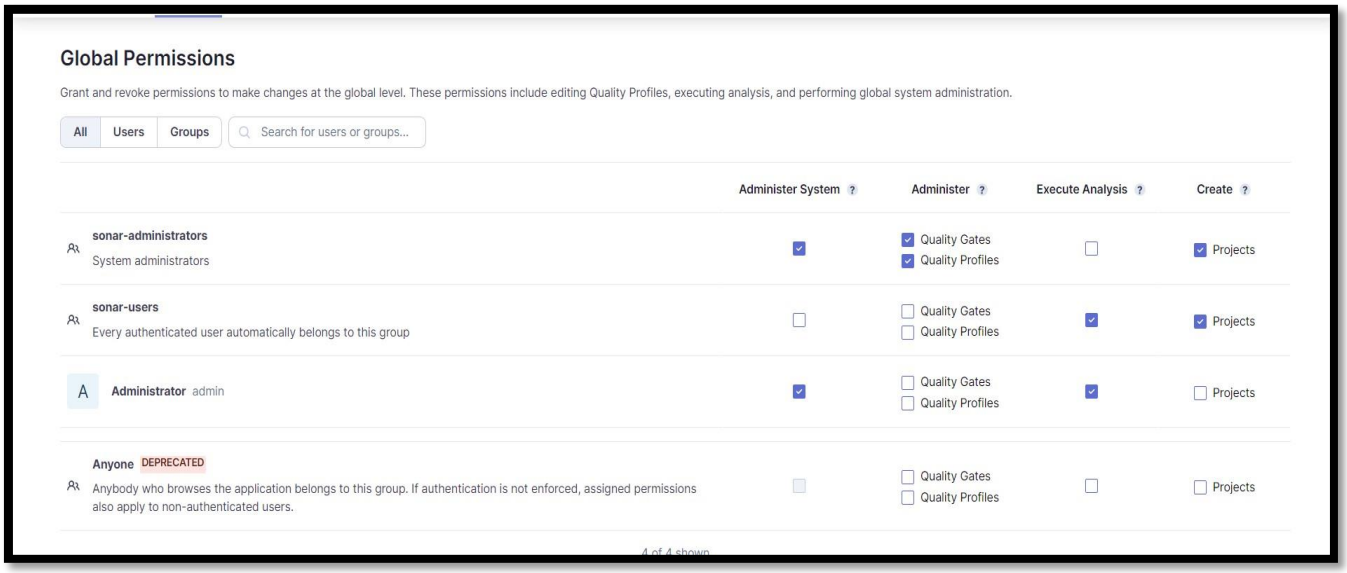
10. Configure Build Steps

- Under the Build section, add a build step to Execute SonarQube Scanner .
- Enter the following analysis properties:
  - sonar.projectKey=my_project_name
  - sonar.login=your_generated_token
  - sonar.sources=HelloWorldCore
  - sonar.host.url=http://localhost:9000



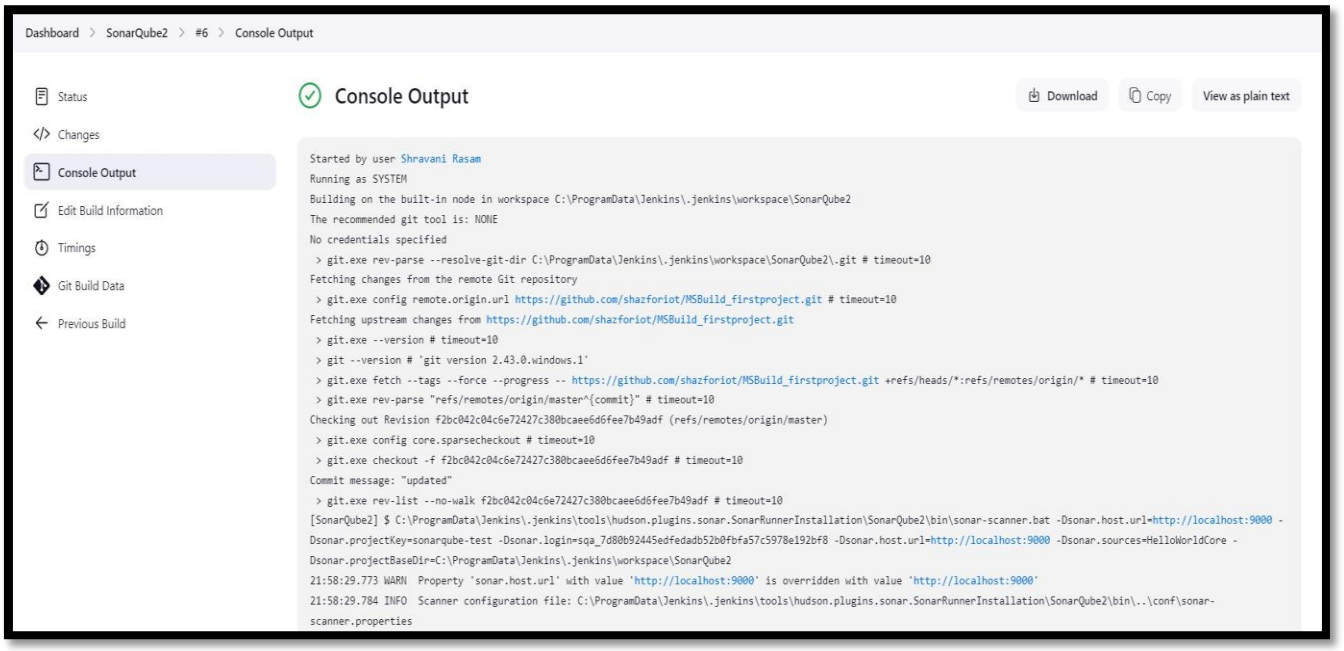
11. Set Permissions in SonarQube

- Navigate to http://localhost:9000/<user-name>/permissions.
- Allow Execute Permissions to the Admin user.



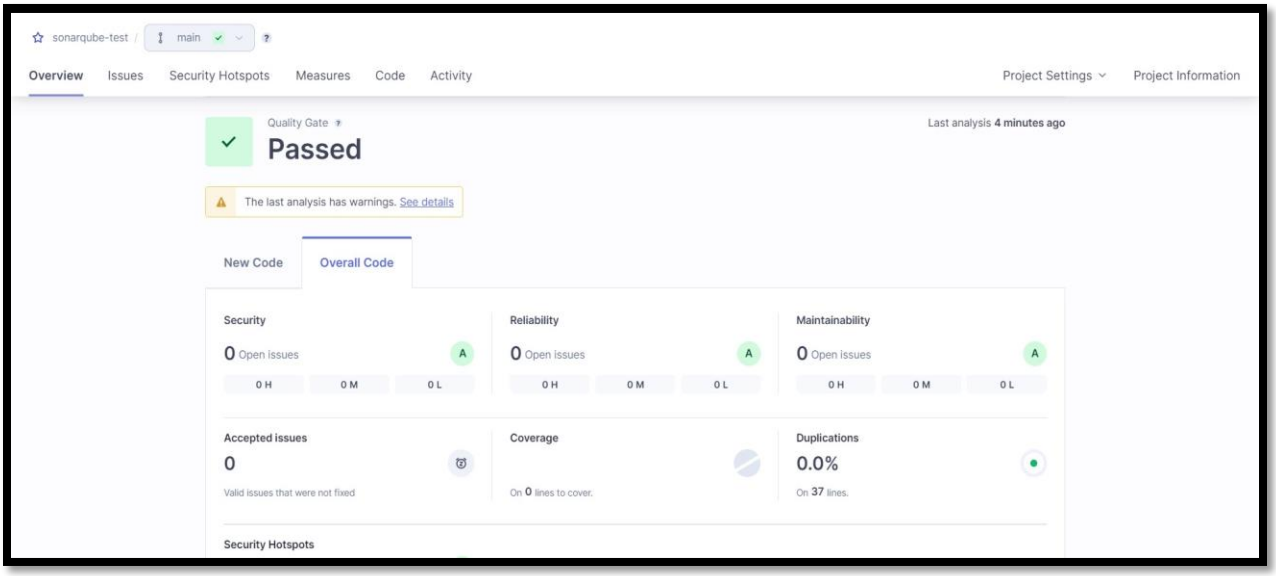
12. Run the Build

- Go back to Jenkins and run the build.
- Check the console output for any errors or issues.



13. Verify in SonarQube

- Once the build is complete, check the project in SonarQube to see the analysis results.



EXPERIMENT NO. 8

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

THEORY:

Static Application Security Testing (SAST)
SAST is a methodology for testing an application's source code to identify security vulnerabilities before the code is compiled. This type of testing, also referred to as white-box testing, helps improve application security by finding weaknesses early in development.

Problems SAST Solves

- **Early Detection**: SAST finds vulnerabilities early in the Software Development Life Cycle (SDLC), allowing developers to fix issues without affecting builds or passing vulnerabilities to the final release.
- **Real-Time Feedback**: Developers receive immediate feedback during coding, helping them address security issues before moving to the next stage of development.
- **Graphical Representations**: SAST tools often provide visual aids to help developers navigate the code and identify the exact location of vulnerabilities, offering suggestions for fixes.
- **Regular Scanning**: SAST tools can be configured to scan code regularly, such as during daily builds, code check-ins, or before releases.

Importance of SAST

- **Resource Efficiency**: With a larger number of developers than security experts, SAST allows full codebase analysis quickly and efficiently, without relying on manual code reviews.
- **Speed**: SAST tools can analyze millions of lines of code within minutes, detecting critical vulnerabilities such as buffer overflows, SQL injection, and cross-site scripting (XSS) with high accuracy.

CI/CD Pipeline
A Continuous Integration/Continuous Delivery (CI/CD) pipeline is a sequence of automated tasks designed to build, test, and deploy new software versions rapidly and consistently. It plays a crucial role in DevOps practices, ensuring fast and reliable software releases.

SonarQube

SonarQube is an open-source platform from SonarSource that performs continuous code quality inspections through static code analysis. It identifies bugs, code smells, security vulnerabilities, and code duplications in a wide range of programming languages. SonarQube is extendable with plugins and integrates seamlessly into CI/CD pipelines.

Benefits of SonarQube

- **Sustainability**: By reducing complexity and vulnerabilities, SonarQube extends the lifespan of applications and helps maintain cleaner code.
- **Increased Productivity**: SonarQube minimizes maintenance costs and risks, resulting in fewer code changes and a more stable codebase.
- **Quality Code**: Ensures code quality checks are integrated into the development process.
- **Error Detection**: Automatically identifies coding errors and alerts developers to resolve them before moving to production.
- **Consistency**: Helps maintain consistent code quality by detecting and reporting violations of coding standards.
- **Business Scaling**: SonarQube supports scaling as the business grows without any restrictions.

Implementation:

Prerequisites

1. Jenkins installed on your machine.
2. Docker installed to run SonarQube.
3. SonarQube installed via Docker

1. Set Up Jenkins

- Open Jenkins Dashboard on localhost:8080 or your configured port .● Install the necessary plugins:
  - SonarQube Scanner Plugin

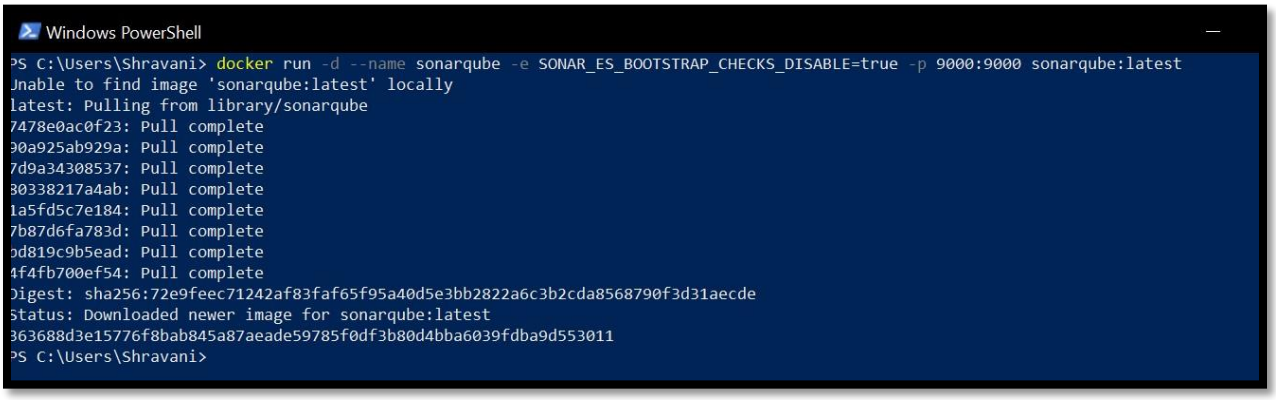
.2. Run SonarQube in Docker

Run the following command to start SonarQube in a Docker container:

command :

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

- Check SonarQube status at http://localhost:9000.
- Login with your credentials:

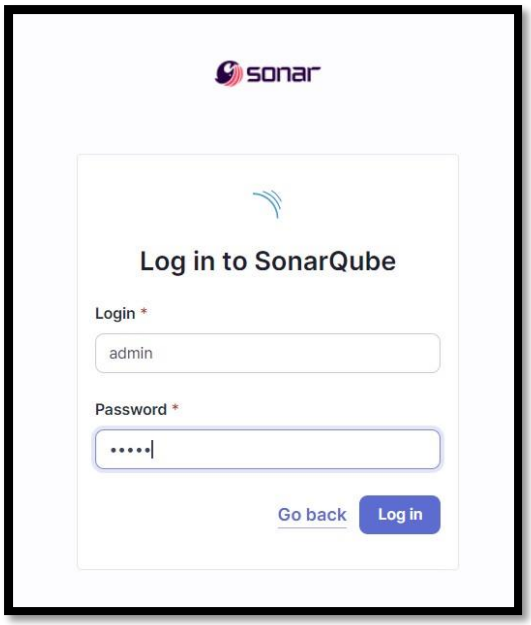


3. Create a Project in

SonarQube ● Go to Projects >

Create Project.

● Name the project (e.g., sonarqube-test)

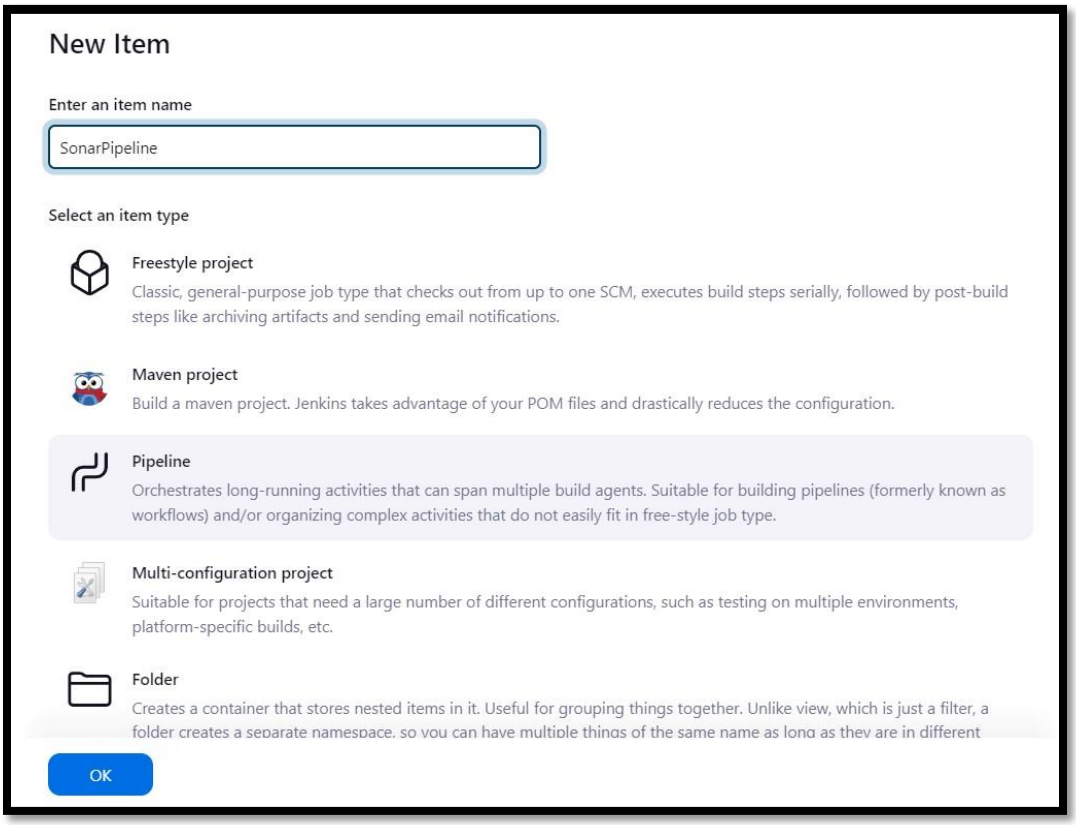


4. Generate SonarQube Token

- Go to My Account > Security > Generate Tokens.
- Copy the generated token for later use

5. Create a Jenkins Pipeline

● Go to Jenkins Dashboard, click New Item, and select Pipeline.



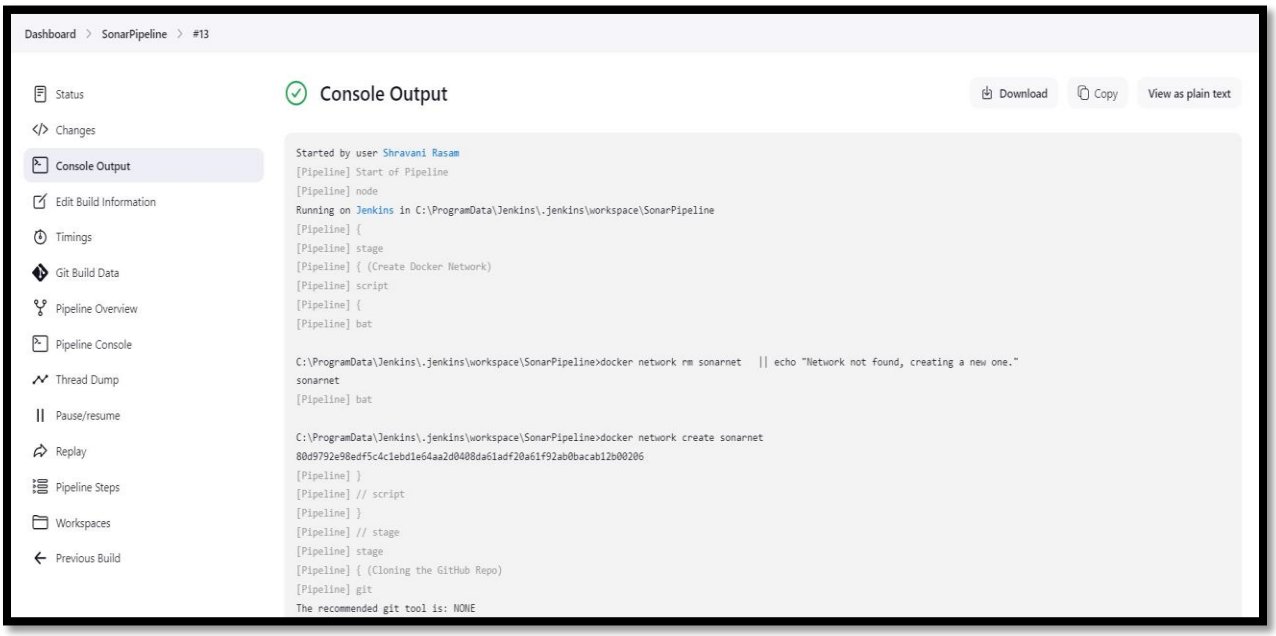
6. Under Pipeline Script, enter the following script:

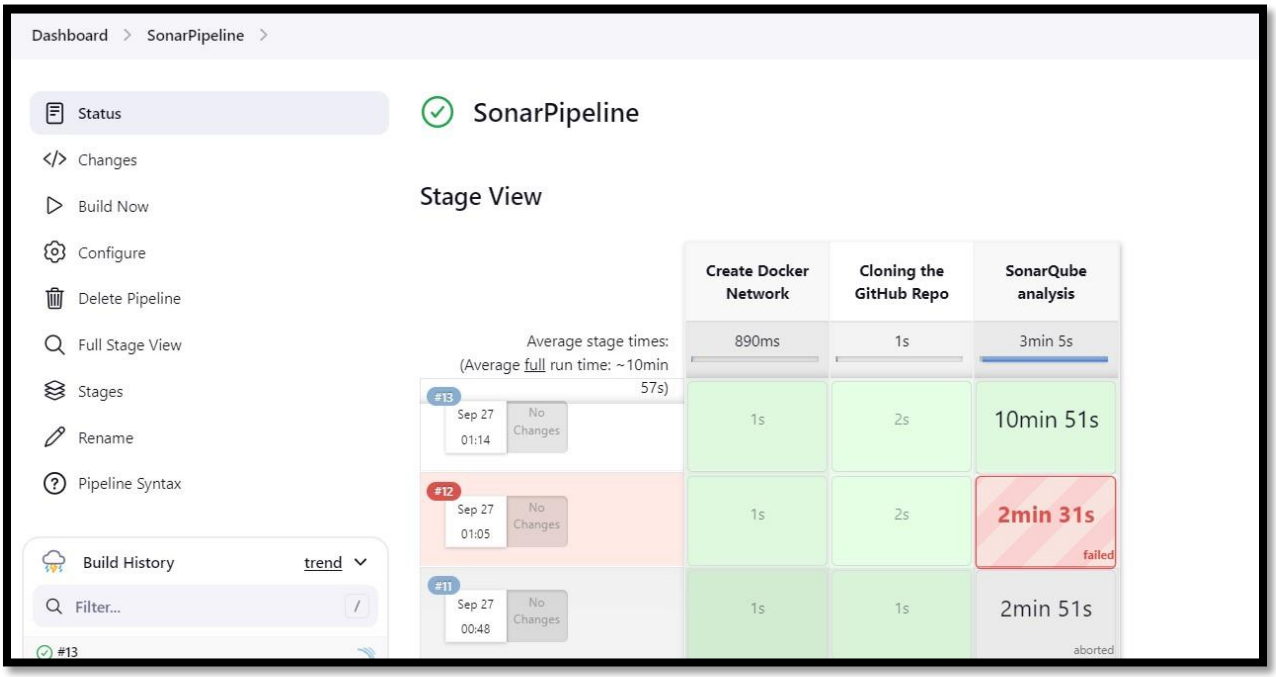
```
pipeline {
    agent any

    stages {
        stage('Create Docker Network') {
            steps {
                script {
                    bat 'docker network rm sonarnet || echo "Network not found, creating a new one."'
                    bat 'docker network create sonarnet'
                }
            }
        }

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

```
        }

        stage('SonarQube analysis') {
            steps {
                withSonarQubeEnv('sonarqube') {
                    bat """ docker run --rm --network
                    sonarnet ^
                        -e SONAR_HOST_URL=http://192.168.133.16:9000 ^
                        -e SONAR_LOGIN=admin ^
                        -e SONAR_PASSWORD=Shravani@0212 ^
                        -e SONAR_PROJECT_KEY=sonarqube-test ^ -
                        v ${WORKSPACE}:/usr/src ^
                        sonarsource/sonar-scanner-cli ^
                        -Dsonar.projectKey=sonarqube-test ^
                        -Dsonar.exclusions=vendor/**,resources/**,**/*.java ^
                        -Dsonar.login=admin ^
                        -Dsonar.password=Shravani@0212
                    """
                }
            }
        }
    }
}
```

7. Run the Pipeline

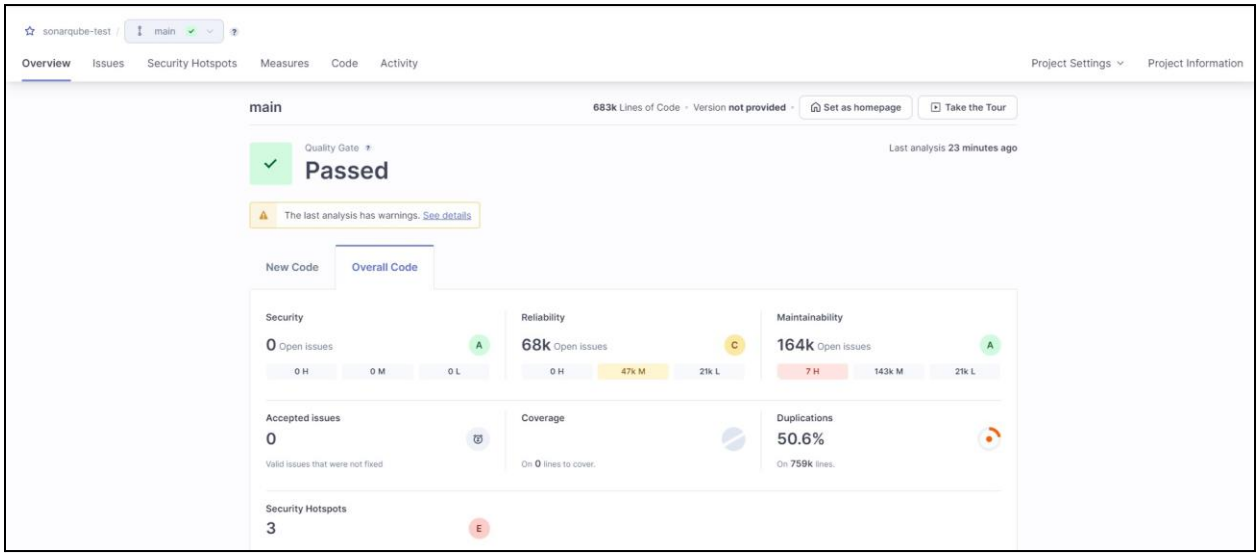
- Save the pipeline and click Build Now
- Monitor the console output for any errors





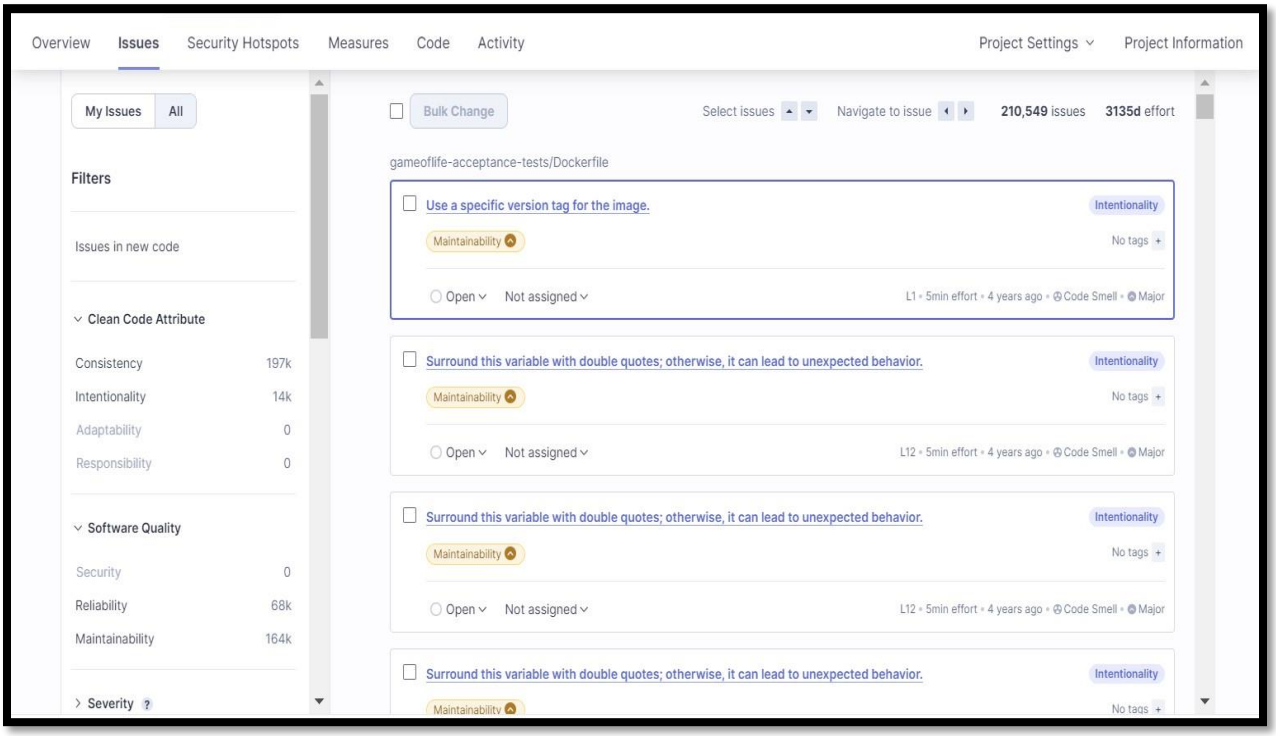
9. Check SonarQube for Analysis Results

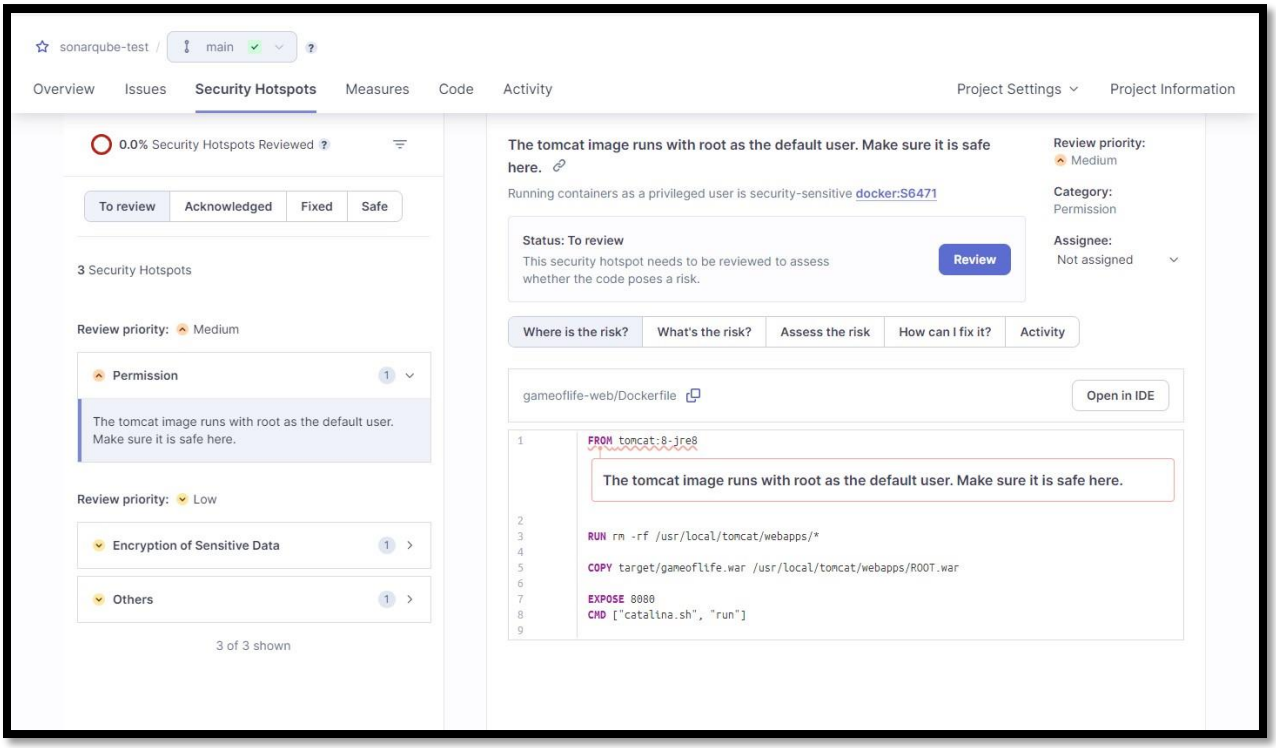
- Go to your SonarQube dashboard and check the project for issues such as bugs, code smells, and security vulnerabilities.

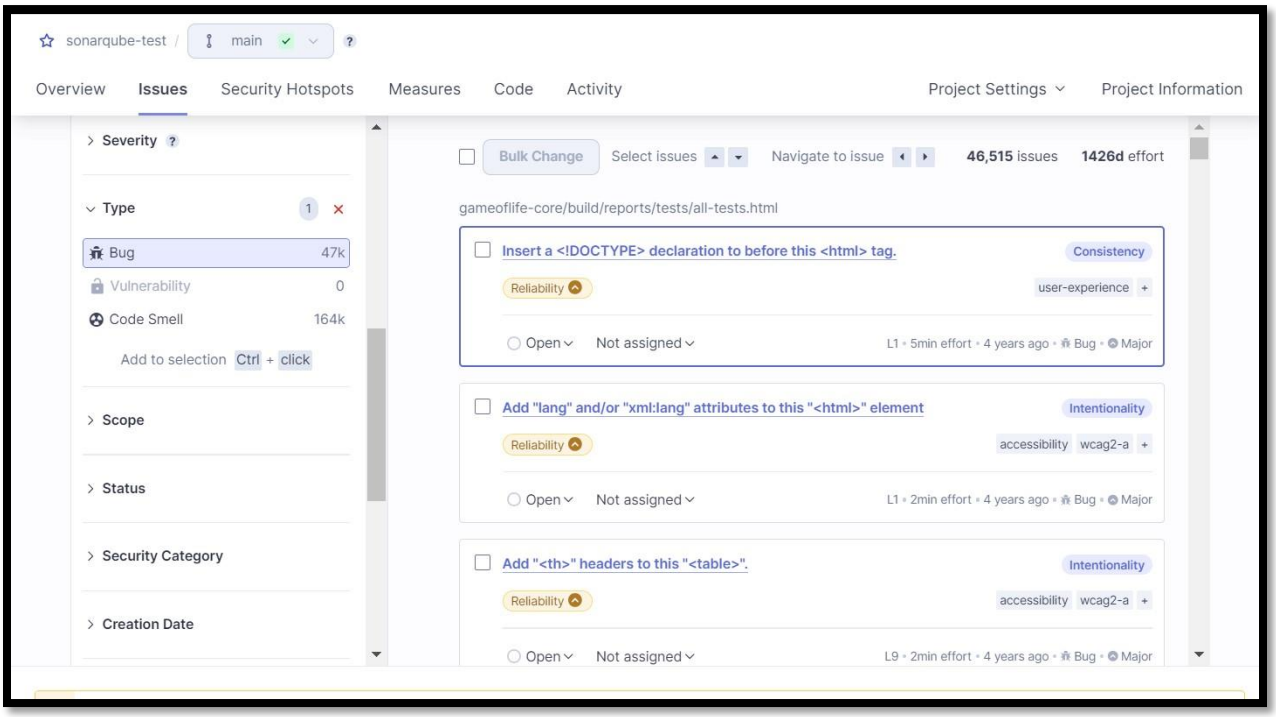


10. Checking SonarQube for Analysis Results of a Code File with Bugs , Code Smells, Security Vulnerabilities, Cyclomatic Complexities and Duplicates .

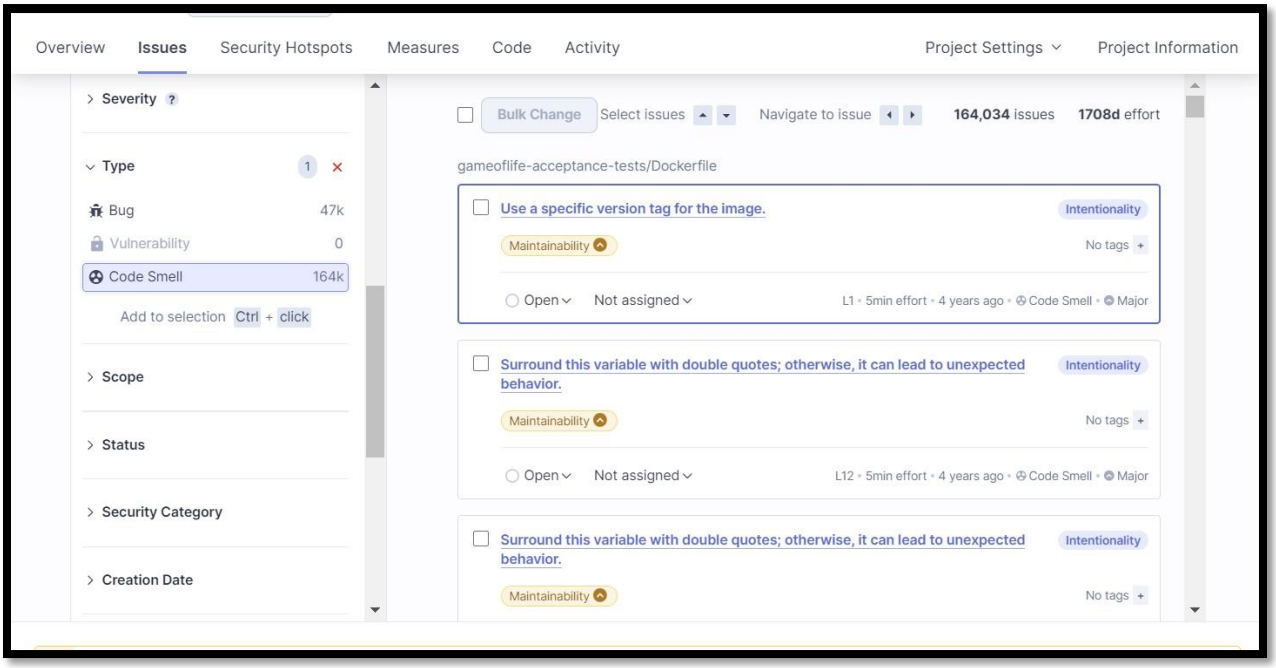
- Issues -



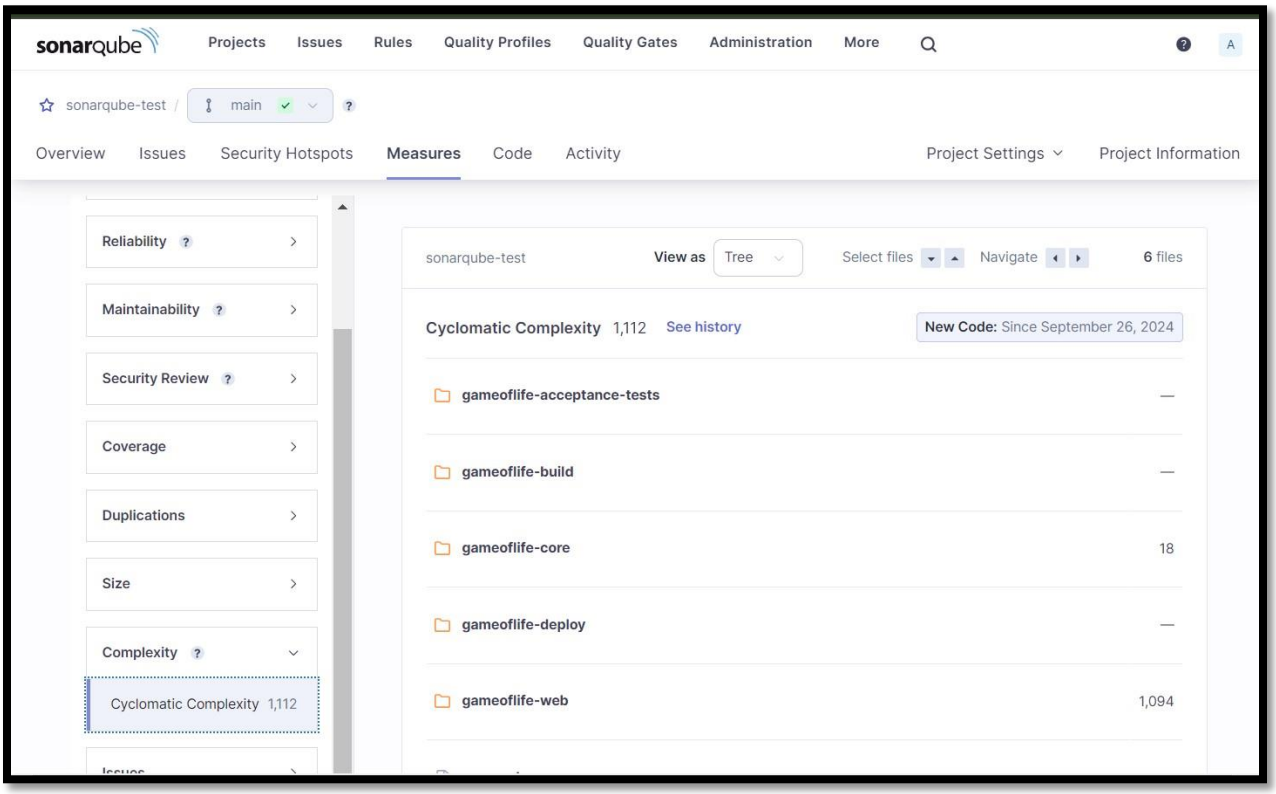


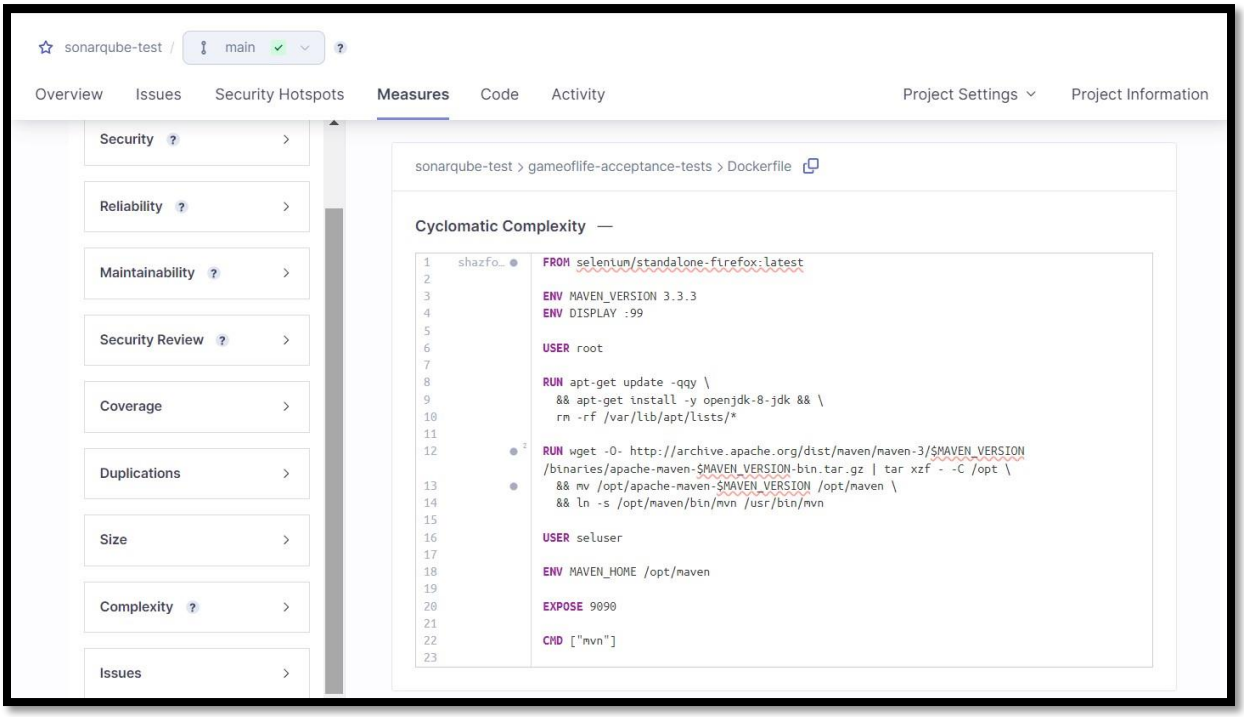


- Security Hotspot (Security Vulnerabilities) -

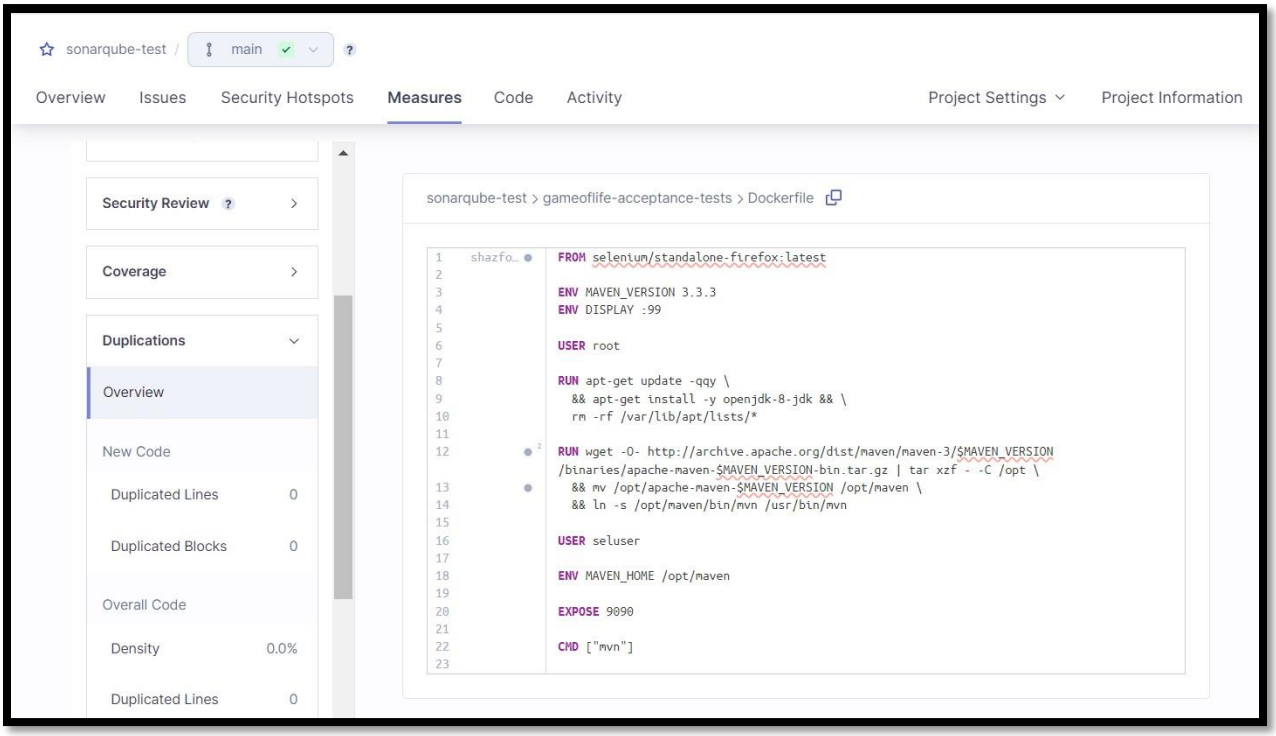


Cyclomatic Complexity -





- Duplications -



CONCLUSION:

In this experiment, we performed a static analysis of the code to detect bugs, code smells, and security vulnerabilities on our sample codes.

EXPERIMENT NO. 9

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

Theory:

What is Nagios?

Nagios is an open-source monitoring tool designed to monitor systems, networks, and infrastructure. It helps organizations identify and resolve IT infrastructure issues before they affect critical business processes. Nagios provides monitoring and alerting services for servers, switches, applications, and services.

Key Features of Nagios

- **Monitoring**: Nagios can monitor a wide range of network services (HTTP, SMTP, POP3, etc.), host resources (processor load, disk usage, system logs, etc.), and environmental factors (temperature, humidity, etc.).
- **Alerting:** When an issue is detected, Nagios can send alerts via email, SMS, or custom scripts to notify administrators.
- **Reporting:** Nagios provides detailed reports and logs of outages, events, notifications, and alert responses, helping in historical analysis and SLA compliance.
- **Scalability:** Nagios is designed to scale and can monitor large, complex environments. 5. Flexibility: With a wide range of plugins and add-ons, Nagios can be customized to meet specific monitoring needs.

How Nagios Works

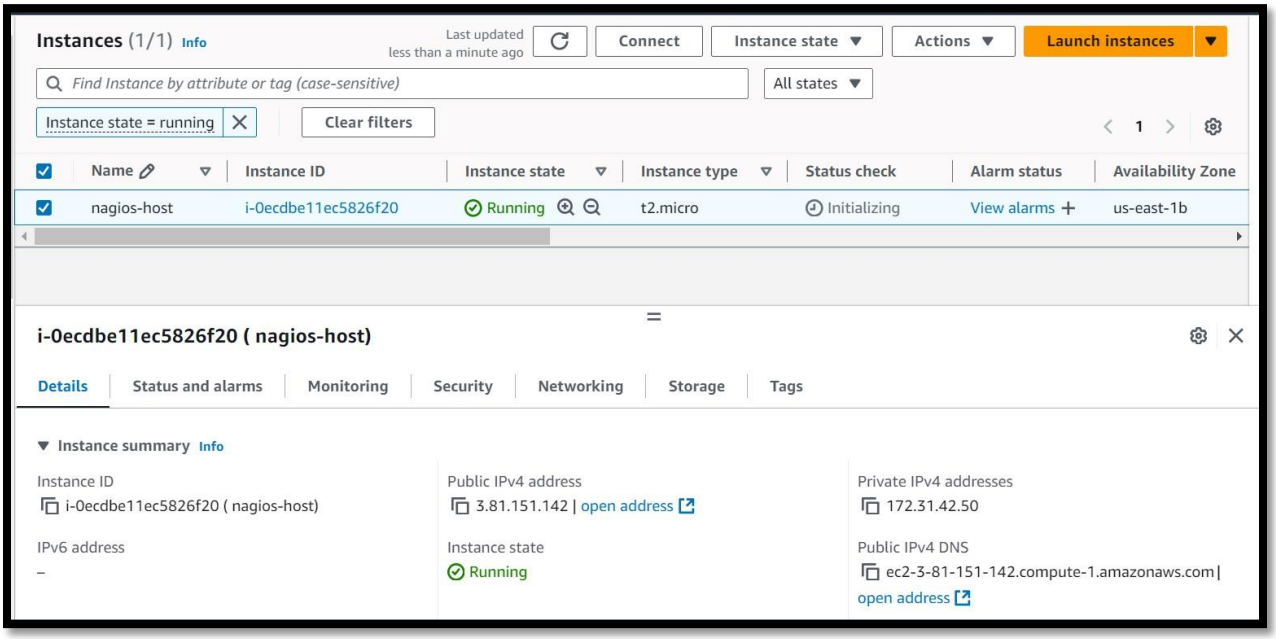
- **Configuration:** Administrators configure Nagios to monitor specific services and hosts. This involves defining what to monitor, how to monitor it, and what actions to take when issues are detected.
- **Plugins:** Nagios uses plugins to gather information about the status of various services and hosts. These plugins can be custom scripts or pre-built ones available in the Nagios community.
- **Scheduling:** Nagios schedules regular checks of the defined services and hosts using the configured plugins.
- **Alerting**: If a check indicates a problem, Nagios triggers an alert. Alerts can be configured to escalate if not acknowledged within a certain timeframe. 5. Web Interface: Nagios provides a web interface for viewing the status of monitored services and hosts, acknowledging alerts, and generating reports.

Setting Up Nagios

1. **Installation**: Install Nagios on a server, typically a Linux-based system.
2. **Configuration Files**: Edit configuration files to define what to monitor and how to monitor it. This includes defining hosts, services, contacts, and notification methods.
3. **Plugins**: Install and configure necessary plugins to monitor specific services and hosts.
4. **Web Interface**: Set up the web interface to allow easy access to monitoring data and alert management.
5. **Testing**: Test the configuration to ensure that Nagios is correctly monitoring the defined services and hosts and that alerts are being sent as expected

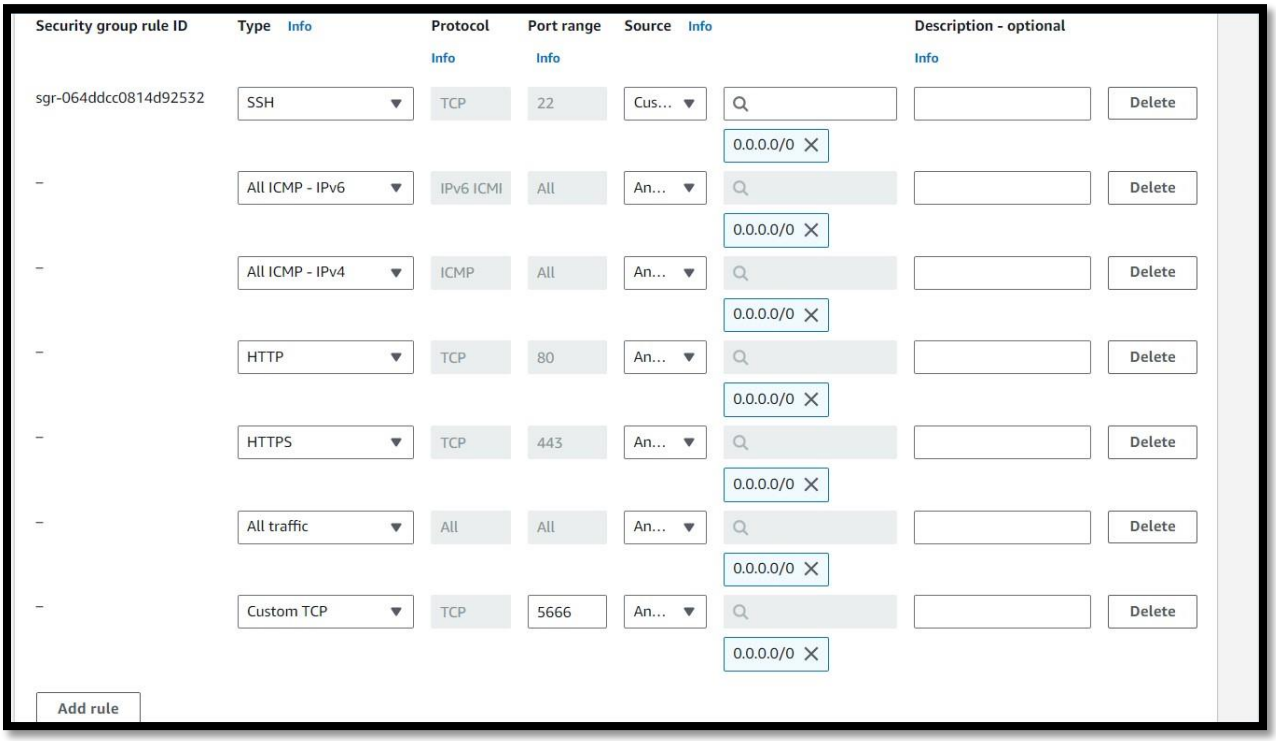
1. Create an Amazon Linux EC2 Instance

- Name it nagios-host.



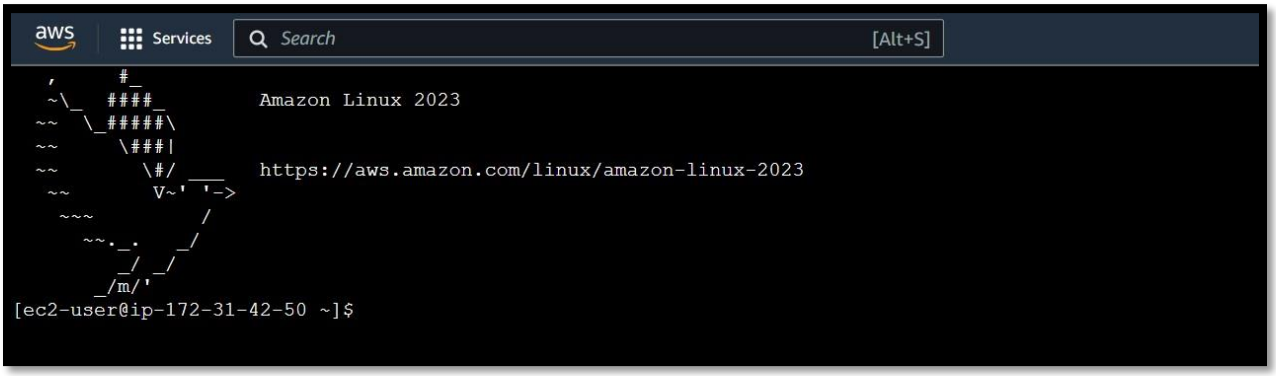
2. Configure Security Group

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



3. Connect to Your EC2 Instance

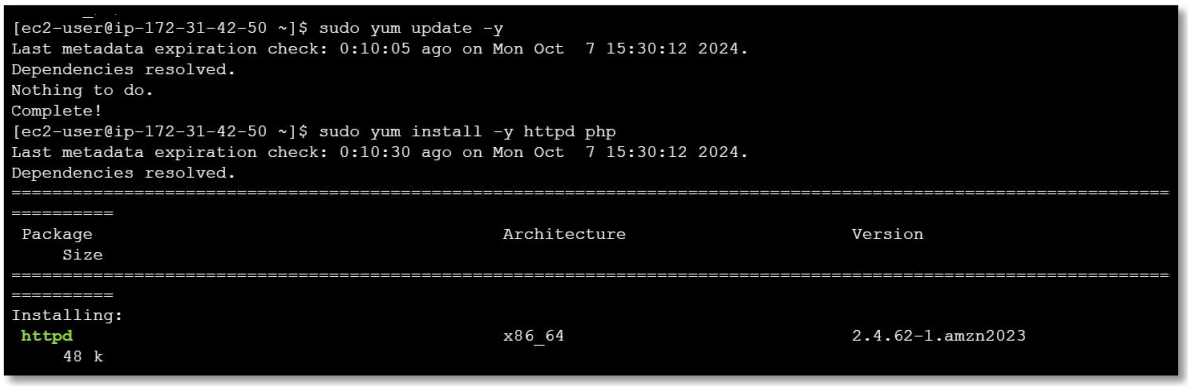
- SSH into your EC2 instance or use EC2 Instance Connect from the browse

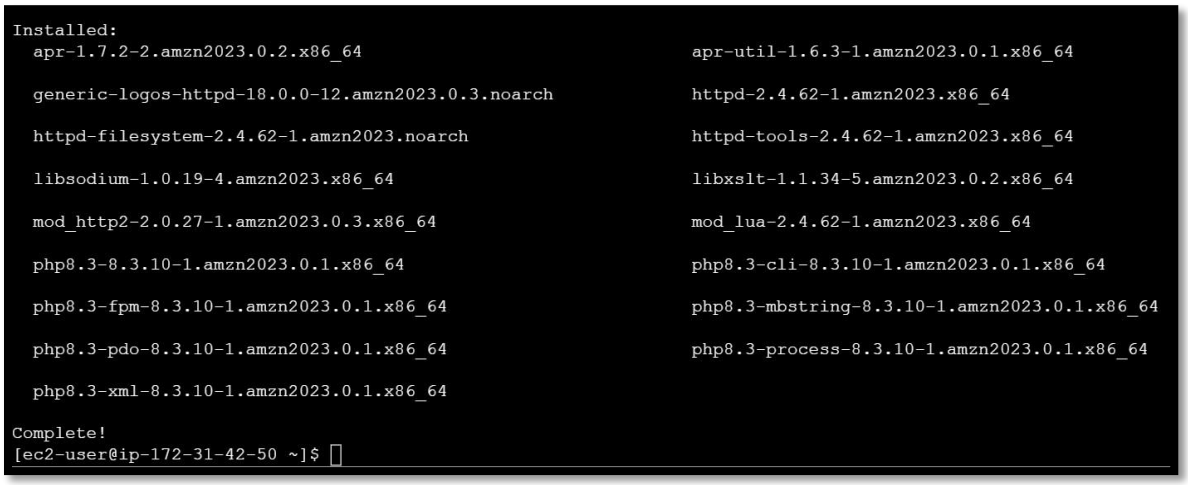


4. Update Package Indices and Install Required Packages

Commands -

- sudo yum update sudo yum install httpd php
- sudo yum install gcc glibc glibc-common
- sudo yum install gd gd-devel





```
[ec2-user@ip-172-31-42-50 ~]$ sudo yum install -y gcc glibc glibc-common
Last metadata expiration check: 0:13:52 ago on Mon Oct  7 15:30:12 2024.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Dependencies resolved.
================================================================================
==========
 Package                                    Architecture
     Size
================================================================================
==========
Installing:
 gcc                                        x86_64
     32 M
Installing dependencies:
 annobin-docs                               noarch
```

5. Create a New Nagios User

Commands ● sudo adduser -m nagios

- sudo passwd nagios

admin123

```
[ec2-user@ip-172-31-42-50 ~]$ sudo useradd nagios
useradd: user 'nagios' already exists
[ec2-user@ip-172-31-42-50 ~]$ sudo useradd nagios
useradd: user 'nagios' already exists
[ec2-user@ip-172-31-42-50 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
Sorry, passwords do not match.
New password:
[ec2-user@ip-172-31-42-50 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-42-50 ~]$
```

6. Create a New User Group

Commands ● sudo groupadd nagcmd

```
[ec2-user@ip-172-31-42-50 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-42-50 ~]$ sudo groupadd nagcmd
groupadd: group 'nagcmd' already exists
[ec2-user@ip-172-31-42-50 ~]$ sudo usermod -aG nagcmd nagios
sudo usermod -aG nagcmd apache
[ec2-user@ip-172-31-42-50 ~]$
```

7. Create a Directory for Nagios Downloads

Commands -

- mkdir ~/downloads
- cd ~/downloads

```
[ec2-user@ip-172-31-42-50 ~]$ mkdir ~/downloads
[ec2-user@ip-172-31-42-50 ~]$ cd ~/downloads
[ec2-user@ip-172-31-42-50 downloads]$
```

8. Download Nagios and Plugins Source Files

Commands -

- Wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz
- wget https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz

```
[ec2-user@ip-172-31-42-50 downloads]$ wget https://assets.nagios.com/downloads/nagioscore/releases/
--2024-10-07 16:07:16--  https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.g
Resolving assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00::f03c:92ff:fef7:45ce
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11333414 (11M) [application/x-gzip]
Saving to: 'nagios-4.4.6.tar.gz'

nagios-4.4.6.tar.gz                          0%[
nagios-4.4.6.tar.gz                          2%[=>
nagios-4.4.6.tar.gz                         19%[=================>
nagios-4.4.6.tar.gz                         49%[=============================================>
nagios-4.4.6.tar.gz                         76%[=============================================
nagios-4.4.6.tar.gz                        100%[=============================================
 1.0s

2024-10-07 16:07:18 (11.1 MB/s) - 'nagios-4.4.6.tar.gz' saved [11333414/11333414]
```

9. Extract the Nagios Source File

Commands -

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

```
[ec2-user@ip-172-31-42-50 downloads]$ tar zxvf nagios-4.4.6.tar.gz
nagios-4.4.6/
nagios-4.4.6/.gitignore
nagios-4.4.6/.travis.yml
nagios-4.4.6/CONTRIBUTING.md
nagios-4.4.6/Changelog
nagios-4.4.6/INSTALLING
nagios-4.4.6/LEGAL
nagios-4.4.6/LICENSE
nagios-4.4.6/Makefile.in
nagios-4.4.6/README.md
nagios-4.4.6/THANKS
nagios-4.4.6/UPGRADING
nagios-4.4.6/aclocal.m4
nagios-4.4.6/autoconf-macros/
nagios-4.4.6/autoconf-macros/.gitignore
nagios-4.4.6/autoconf-macros/CHANGELOG.md
nagios-4.4.6/autoconf-macros/LICENSE
```

10. Run the Configuration Script Commands

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

```
[ec2-user@ip-172-31-42-50 downloads]$ cd nagios-4.4.6
[ec2-user@ip-172-31-42-50 nagios-4.4.6]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for ANSI C header files... yes
```

11. Compile the Source Code

Commands make all

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

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

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

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

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

For more information on obtaining support for Nagios, visit:

       https://support.nagios.com

************************************************************
```

12. Install Binaries, Init Script, and Sample Config Files

Commands -

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

```
[ec2-user@ip-172-31-42-50 nagios-4.4.6]$ sudo make install
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.4.6/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.4.6/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.4.6/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.4.6/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
```

```
[ec2-user@ip-172-31-42-50 nagios-4.4.6]$ sudo make install-init
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/
[ec2-user@ip-172-31-42-50 nagios-4.4.6]$ sudo make install-config
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/n
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagi
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templat
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/command
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contact
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeper
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localho
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.
```

```
[ec2-user@ip-172-31-42-50 nagios-4.4.6]$ sudo make install-commandmode
/usr/bin/install -c -m 775 -o nagios -g nagcmd -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

[ec2-user@ip-172-31-42-50 nagios-4.4.6]$
```

14. Edit the Config File to Change the Email Address

Commands -

- sudo nano /usr/local/nagios/etc/objects/contacts.cfg
- Change the email address in the contacts.cfg file to your preferred email.

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

define contact {

    contact_name            nagiosadmin             ; Short name of user
    use                     generic-contact         ; Inherit default values from generic-contact template (defined above)
    alias                   Nagios Admin            ; Full name of user
    email                   shravanirasam0212@gmail.com  ; <<***** CHANGE THIS TO YOUR EMAIL ADDRESS ******
}



##############################################################################
#
# CONTACT GROUPS
#
```

15. Configure the Web Interface

Commands -

sudo make install-webconf

```
[ec2-user@ip-172-31-42-50 nagios-4.4.6]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

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

[ec2-user@ip-172-31-42-50 nagios-4.4.6]$
```

16.Create a Nagios Admin Account

Commands -

- sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
- You will be prompted to enter and confirm the password for the nagiosadmin user

```
[ec2-user@ip-172-31-42-50 nagios-4.4.6]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-42-50 nagios-4.4.6]$
```

admin123

17. Restart Apache

Commands -

- sudo systemctl restart httpd

18. Extract the Plugins Source File

Commands ● cd ~/downloads

- tar zxvf nagios-plugins-2.3.3.tar.gz cd nagios-plugins-2.3.3

```
[ec2-user@ip-172-31-42-50 nagios-4.4.6]$ sudo systemctl restart httpd
[ec2-user@ip-172-31-42-50 nagios-4.4.6]$ cd ~/downloads
[ec2-user@ip-172-31-42-50 downloads]$ tar zxvf nagios-plugins-2.3.3.tar.gz
nagios-plugins-2.3.3/
nagios-plugins-2.3.3/perlmods/
nagios-plugins-2.3.3/perlmods/Config-Tiny-2.14.tar.gz
nagios-plugins-2.3.3/perlmods/parent-0.226.tar.gz
nagios-plugins-2.3.3/perlmods/Test-Simple-0.98.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile.in
nagios-plugins-2.3.3/perlmods/version-0.9903.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile.am
nagios-plugins-2.3.3/perlmods/Module-Runtime-0.013.tar.gz
nagios-plugins-2.3.3/perlmods/Module-Metadata-1.000014.tar.gz
nagios-plugins-2.3.3/perlmods/Params-Validate-1.08.tar.gz
nagios-plugins-2.3.3/perlmods/Class-Accessor-0.34.tar.gz
```

19. Compile and Install Plugins Commands -

- ./configure --with-nagios-user=nagios --with-nagios-group=nagios make
- sudo make install

```
[ec2-user@ip-172-31-42-50 downloads]$ cd nagios-plugins-2.3.3
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
sudo make install
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether to disable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
```

20. Start Nagios

Commands

- sudo chkconfig --add nagios
- sudo chkconfig nagios on
- sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg ● sudo systemctl start nagios

```
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$ sudo chkconfig --add nagios
sudo chkconfig nagios on
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
sudo systemctl start nagios
error reading information on service nagios: No such file or directory
Note: Forwarding request to 'systemctl enable nagios.service'.
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/l

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...
```

21. Check the Status of Nagios

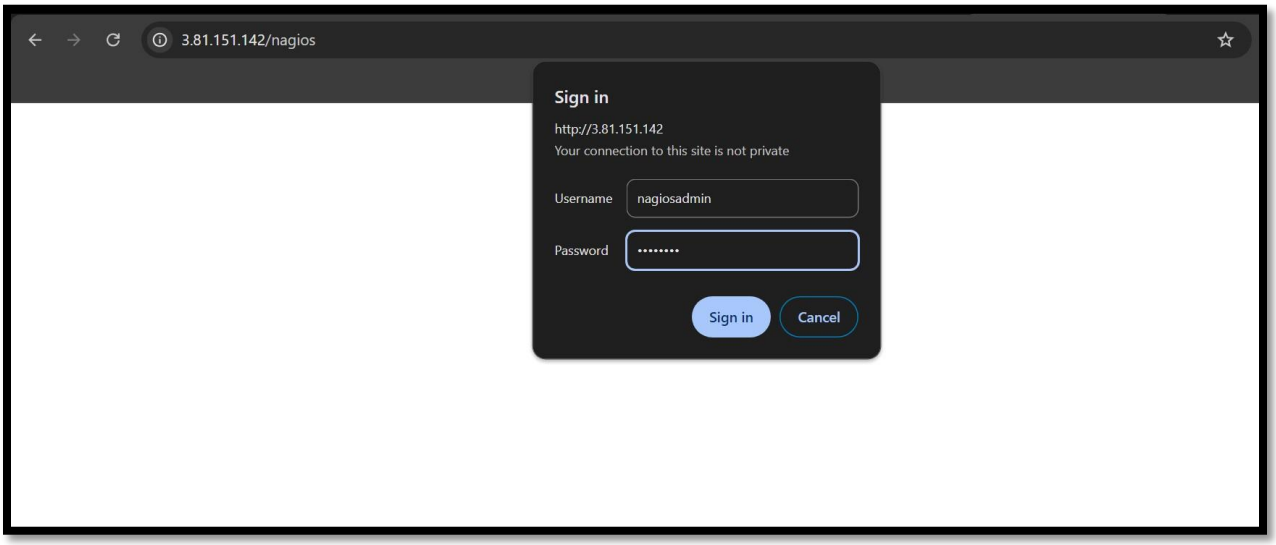
Commands -

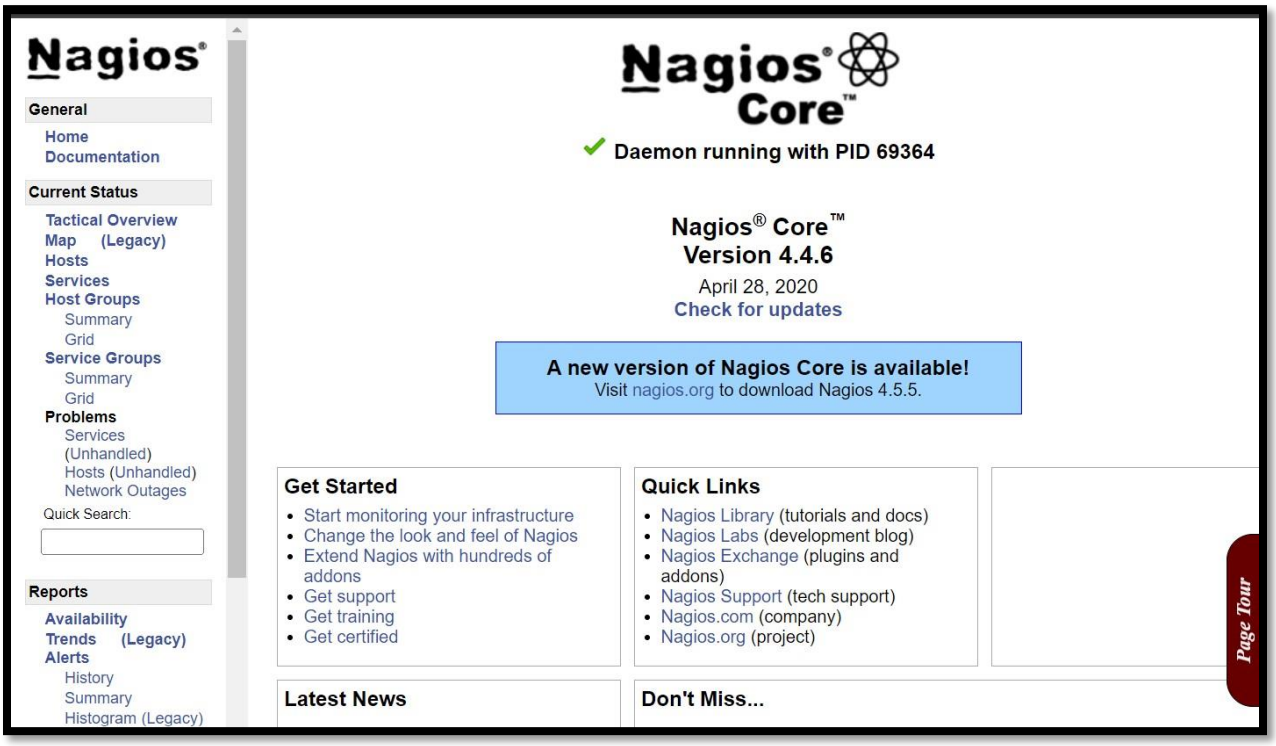
- sudo systemctl status nagios

```
Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Mon 2024-10-07 16:28:45 UTC; 38s ago
       Docs: https://www.nagios.org/documentation
    Process: 69362 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
    Process: 69363 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (c
   Main PID: 69364 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 2.1M
        CPU: 22ms
     CGroup: /system.slice/nagios.service
             ├─69364 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─69365 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─69366 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─69367 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─69368 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─69369 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
```

22. Access Nagios Web Interface

- Copy the Public IP address of your EC2 instance.
  - Open your browser and navigate to http:///nagios.
- Enter the username nagiosadmin and the password you set in Step 16





Conclusion:

After installing and configuring Nagios Core, Plugins, and NRPE on a Linux machine, We have a robust continuous monitoring setup, ensuring proactive issue detection and optimal system performance.

EXPERIMENT NO. 10

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

Theory:

Nagios is a comprehensive monitoring and alerting platform designed to keep track of IT infrastructure, networks, and applications. It provides real-time monitoring, alerting, and reporting capabilities to ensure the health and performance of critical systems.

Key Components of Nagios

1. **Nagios Core**: The open-source foundation of the Nagios monitoring system. It provides the basic framework for monitoring and alerting.
2. **Nagios XI**: A commercial version of Nagios that offers advanced features, a more user-friendly interface, and additional support options.
3. **Nagios Log Server**: A tool for centralized log management, allowing you to view, analyze, and archive logs from various sources.
4. **Nagios Network Analyzer**: Provides detailed insights into network traffic and bandwidth usage.
5. **Nagios Fusion**: Centralizes monitoring data from multiple Nagios instances, providing a unified view of the entire network.

Monitoring Capabilities

1. **Port Monitoring**: Nagios can monitor specific network ports to ensure they are open and responsive. This is crucial for services that rely on these ports.
2. **Service Monitoring**: Nagios checks the status of various services (e.g., web servers, databases) to ensure they are running smoothly.
3. **Server Monitoring:** Nagios can monitor both Windows and Linux servers using agents like NSClient++ for Windows and NRPE for Linux. This includes metrics like CPU usage, memory usage, disk space, and more.

How Nagios Works

1. **Configuration**: Administrators define what to monitor and how to monitor it using configuration files.

2. **Plugins:** Nagios uses plugins to gather information about the status of various services and hosts. These plugins can be custom scripts or pre-built ones.
3. **Scheduling:** Nagios schedules regular checks of the defined services and hosts using the configured plugins.
4. **Alerting:** If a check indicates a problem, Nagios triggers an alert. Alerts can be configured to escalate if not acknowledged within a certain timeframe. 5. Log Management: Centralizing and analyzing logs from various sources to detect issues and ensure compliance.

Implementation :

Prerequisites

- AWS Free Tier
- Nagios Server running on an Amazon Linux Machine

1. Confirm Nagios is Running on the Server

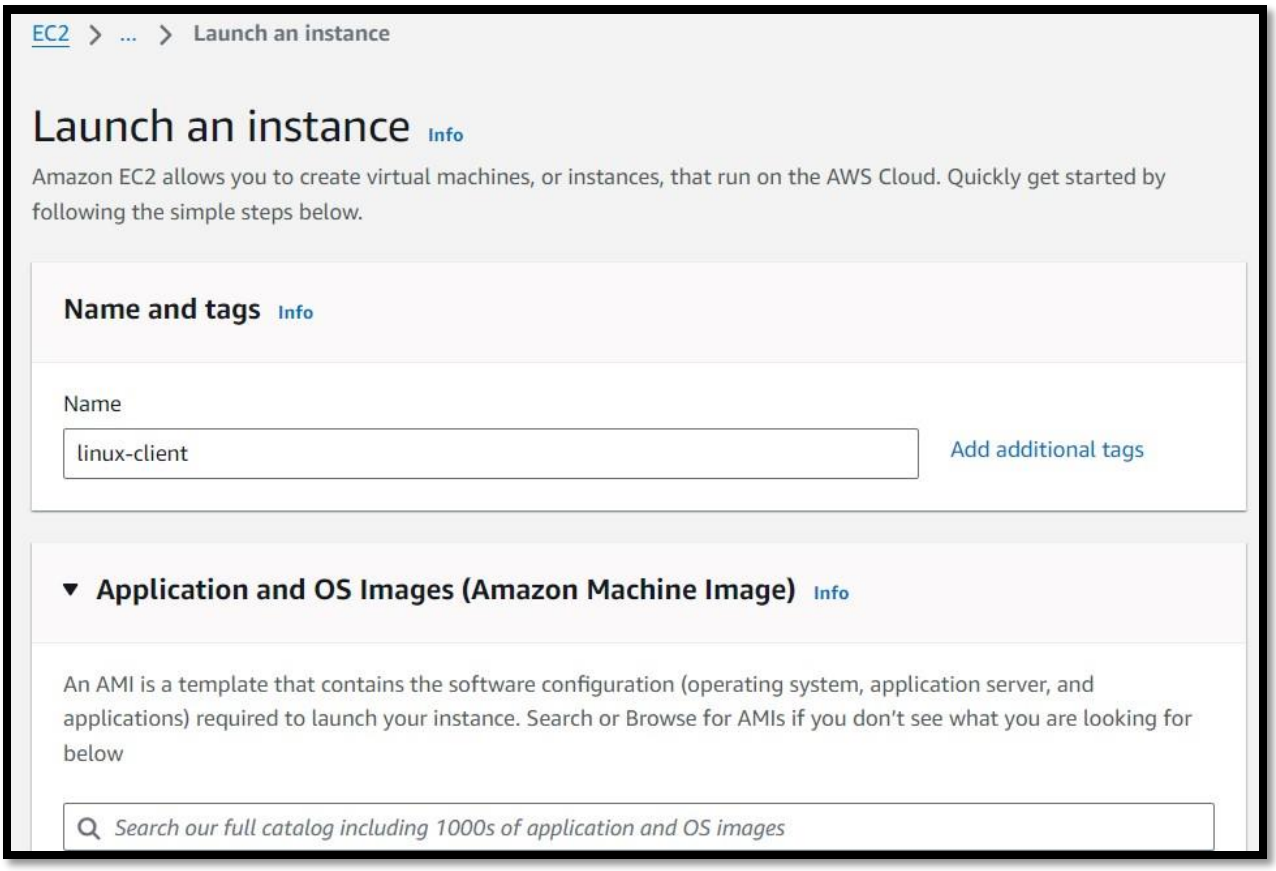
Commands -

- sudo systemctl status nagios
- Proceed if you see that Nagios is active and running.

```
Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Mon 2024-10-07 16:28:45 UTC; 38s ago
       Docs: https://www.nagios.org/documentation
    Process: 69362 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
    Process: 69363 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (c
   Main PID: 69364 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 2.1M
        CPU: 22ms
     CGroup: /system.slice/nagios.service
             ├─69364 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─69365 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─69366 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─69367 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─69368 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─69369 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
```

2. Create an Ubuntu 20.04 Server EC2 Instance

- Name it linux-client.
- Use the same security group as the Nagios Host



3. Verify Nagios Process on the Server

Commands

- - ps -ef | grep nagios

```
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$ ps -ef | grep nagios
nagios     69364       1  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios -d
nagios     69365   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --
nagios     69366   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --
nagios     69367   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --
nagios     69368   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios --
nagios     69369   69364  0 16:28 ?        00:00:00 /usr/local/nagios/bin/nagios -d
ec2-user   70969    2909  0 16:55 pts/0    00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$
```

4. Become Root User and Create Directories

Commands ● sudo su

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

```
[ec2-user@ip-172-31-42-50 nagios-plugins-2.3.3]$ sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-42-50 nagios-plugins-2.3.3]#
```

5. Copy Sample Configuration File

Commands -

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

```
[root@ip-172-31-42-50 ec2-user]# cp /usr/local/nagios/etc/objects/localhost.cfg /usr/local/nagios/etc/objects/monitorhosts/
[root@ip-172-31-42-50 ec2-user]# nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg
[root@ip-172-31-42-50 ec2-user]#
```

6. Edit the Configuration File

Commands -

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

- Change hostname to linuxserver everywhere in the file.
- Change address to the public IP address of your linux-client.

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

# Define a host for the local machine

define host {

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

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

7. Update Nagios Configuration

Commands -

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

- Add the following line: cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
- Change hostgroup_name under hostgroup to linux-servers1

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

# Define an optional hostgroup for Linux machines

define hostgroup {

    hostgroup_name          linux-servers1          ; The name of the hostgroup
    alias                   Linux Servers           ; Long name of the group
    members                 localhost               ; Comma separated list of hosts that belong to this group
}



##############################################################################
#
```

8. Verify Configuration Files

Commands -

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

ERROR OCCURRED

```
[root@ip-172-31-42-50 ec2-user]# 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...
Error: Could not find any host matching 'linuxerver' (config file '/usr/local/nagios/etc/objects/monito
Error: Failed to expand host list 'linuxerver' for service 'Total Processes' (/usr/local/nagios/etc/obj
   Error processing object config files!


***> One or more problems was encountered while processing the config files...

     Check your configuration file(s) to ensure that they contain valid
     directives and data definitions.  If you are upgrading from a previous
     version of Nagios, you should be aware that some variables/definitions
     may have been removed or modified in this version.  Make sure to read
     the HTML documentation regarding the config files, as well as the
     'Whats New' section to find out what has changed.
```

Error resolved

```
[root@ip-172-31-42-50 ec2-user]# 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 16 services.
       Checked 2 hosts.
       Checked 2 host groups.
       Checked 0 service groups.
       Checked 1 contacts.
       Checked 1 contact groups.
       Checked 24 commands.
       Checked 5 time periods.
       Checked 0 host escalations.
       Checked 0 service escalations.
```

```
Checking 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-172-31-42-50 ec2-user]#
```

9. **Restart Nagios Service** Commands -
   - sudo systemctl restart nagios

10. SSH into the Client Machine

- Use SSH or EC2 Instance Connect to access the linux-client.

11. Update Package Index and Install Required Packages

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

```
ubuntu@ip-172-31-33-27:~$ 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 InR
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports I
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe am
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Pack
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Tr
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe am
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe am
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse
```

12. Edit NRPE Configuration File

Commands -

sudo nano /etc/nagios/nrpe.cfg

● Add your Nagios host IP address under allowed_hosts: allowed_hosts=

```
# supported.
#
# Note: The daemon only does rudimentary checking of the client's IP
# address.  I would highly recommend adding entries in your /etc/hosts.allow
# file to allow only the specified host to connect to the port
# you are running this daemon on.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

allowed_hosts=127.0.0.1,3.81.151.142



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

14. Check Nagios Dashboard

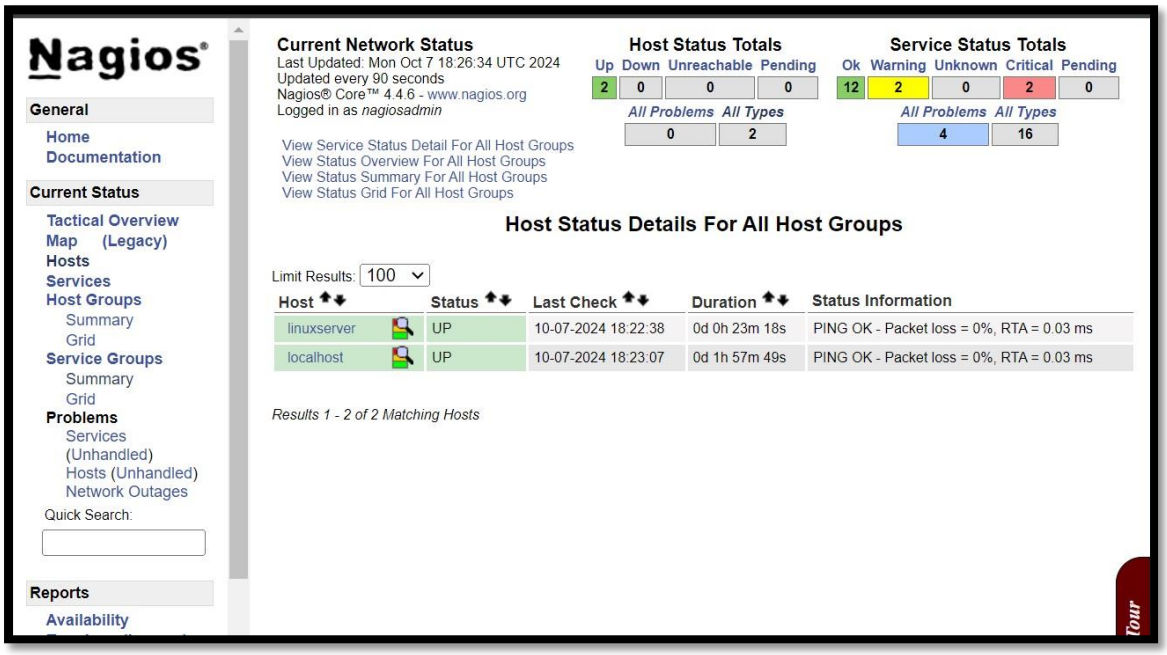
● Open your browser and navigate to http:///nagios.

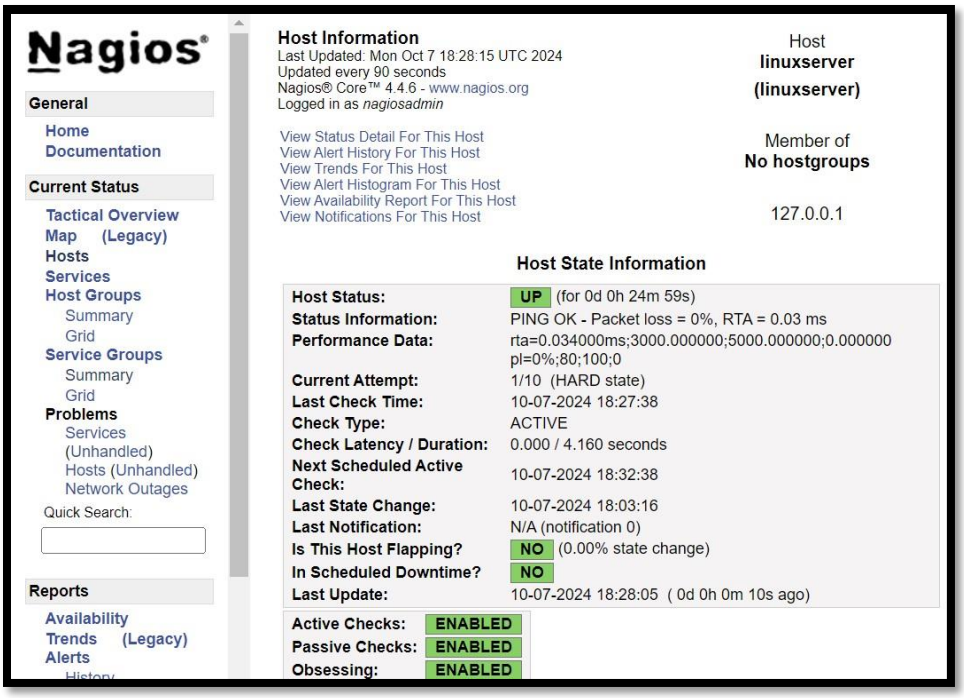
● Log in with nagiosadmin and the password you set earlier.

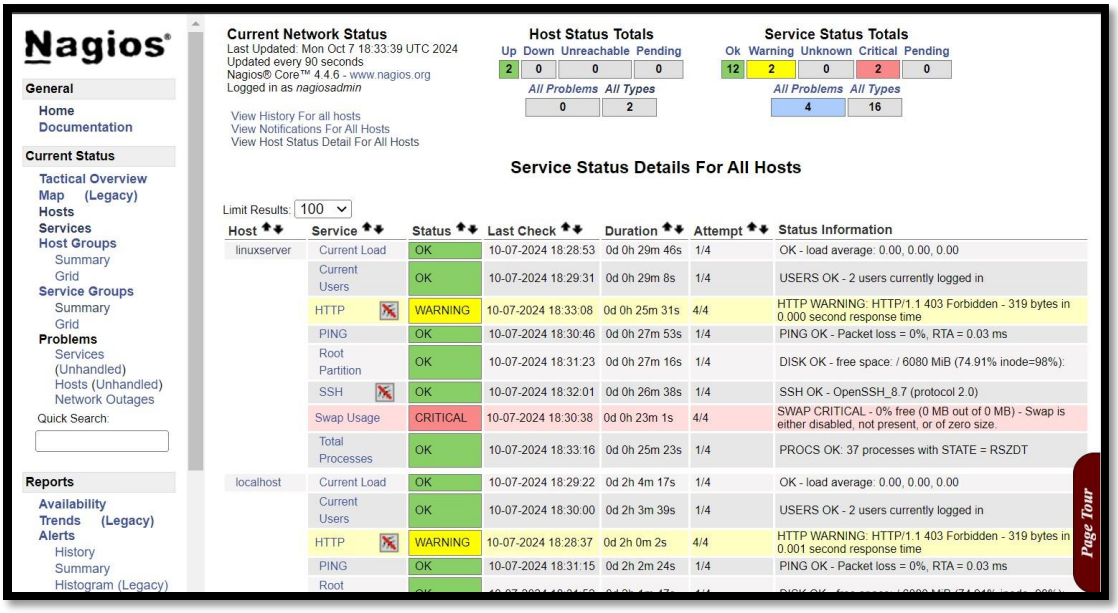
● You should see the new host linuxserver added.

● Click on Hosts to see the host details.

- Click on Services to see all services and ports being monitored







Conclusion:

To perform port, service, and Windows/Linux server monitoring using Nagios, configure the necessary plugins and agents, define the monitoring parameters in the configuration files, and set up alerting mechanisms to ensure timely notifications of any issues. This comprehensive approach ensures robust monitoring and quick response to potential problems, maintaining the health and performance of your IT infrastructure.

Advance Devops-11

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

Theory:

AWS Lambda

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

The concept of "serverless" computing refers to not needing to maintain your own servers to run these functions. AWS Lambda is a fully managed service that takes care of all the infrastructure

for you. And so "serverless" doesn't mean that there are no servers involved: it just means that the servers, the operating systems, the network layer and the rest of the infrastructure have already been taken care of so that you can focus on writing application code.

Features of AWS Lambda

- AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved.
- It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateway,

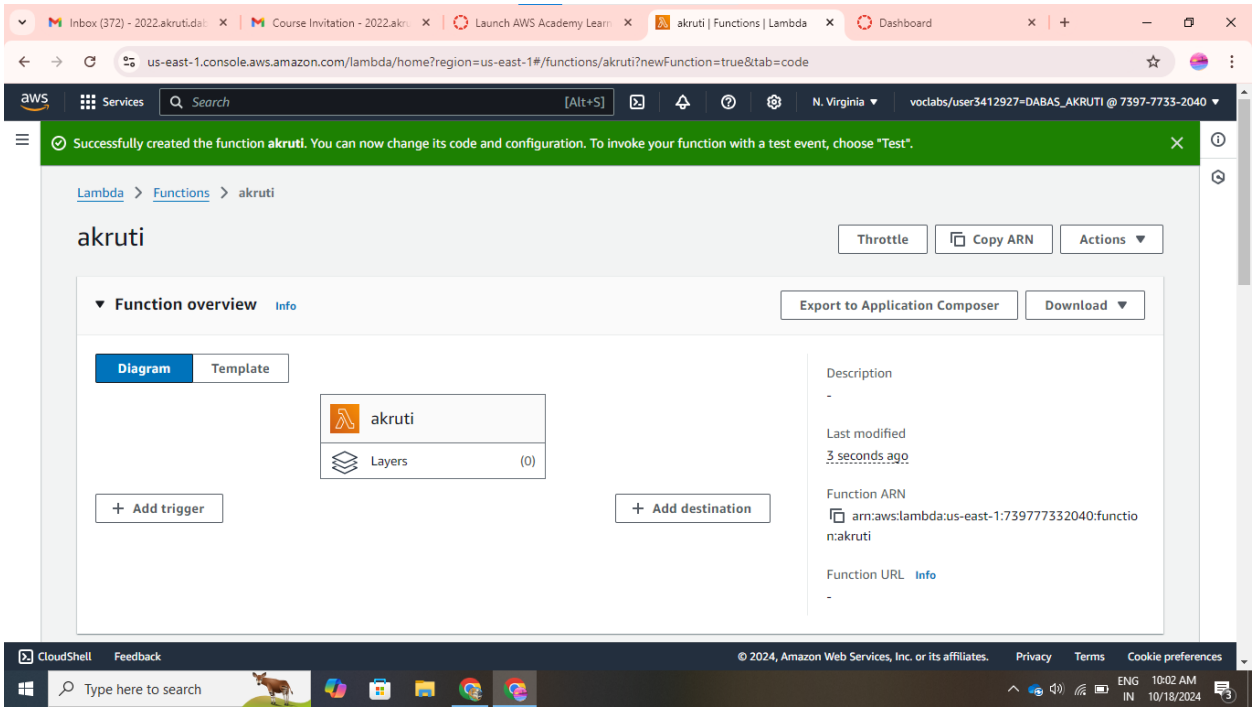
Kinesis,

CodeCommit, and many more to trigger an event.

- You don't need to invest upfront. You pay only for the memory used by the lambda function and minimal cost on the number of requests hence cost-efficient.
- AWS Lambda is secure. It uses AWS IAM to define all the roles and security policies.
  - It offers fault tolerance for both services running the code and the function. You do not have to worry about the application down.

Steps to create an AWS Lambda function

1. Open up the Lambda Console and click on the Create button. Be mindful of where you create your functions since Lambda is region-dependent.

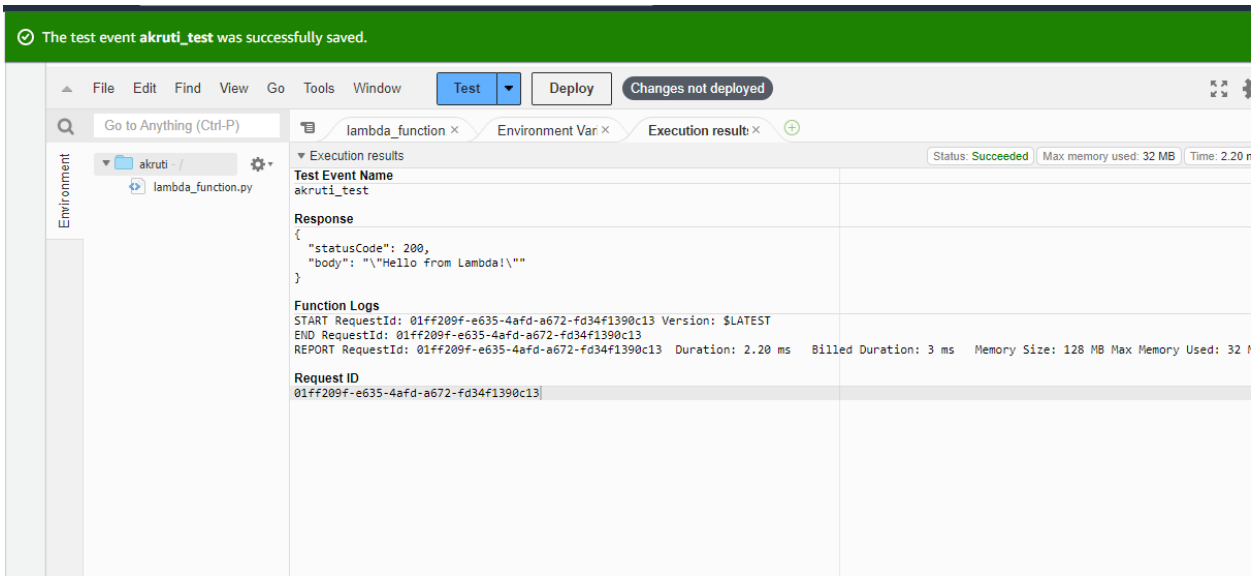


2. Choose to create a function from scratch or use a blueprint, i.e templates defined by AWS for you with all configuration presets required for the most common use cases.

Then, choose a runtime env for your function, under the dropdown, you can see all the options AWS supports, Python, Nodejs, .NET and Java being the most popular ones. After that, choose to create a new role with basic Lambda permissions if you don't have an existing one.

Click on the Create button.

3. This process will take a while to finish and after that, you'll get a message that your function was successfully created.

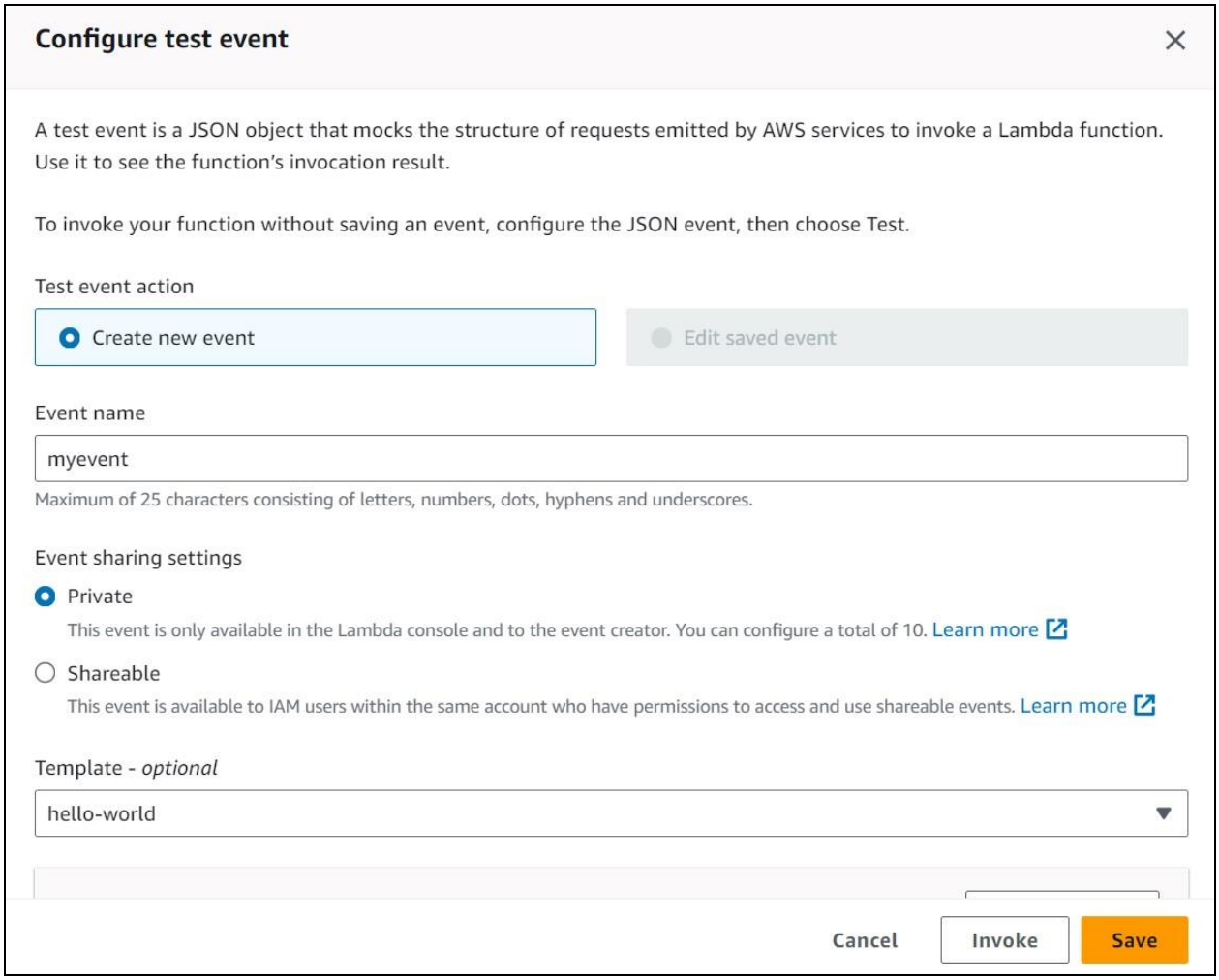


4. To change the configuration, open up the Configuration tab and under General Configuration, choose Edit.

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

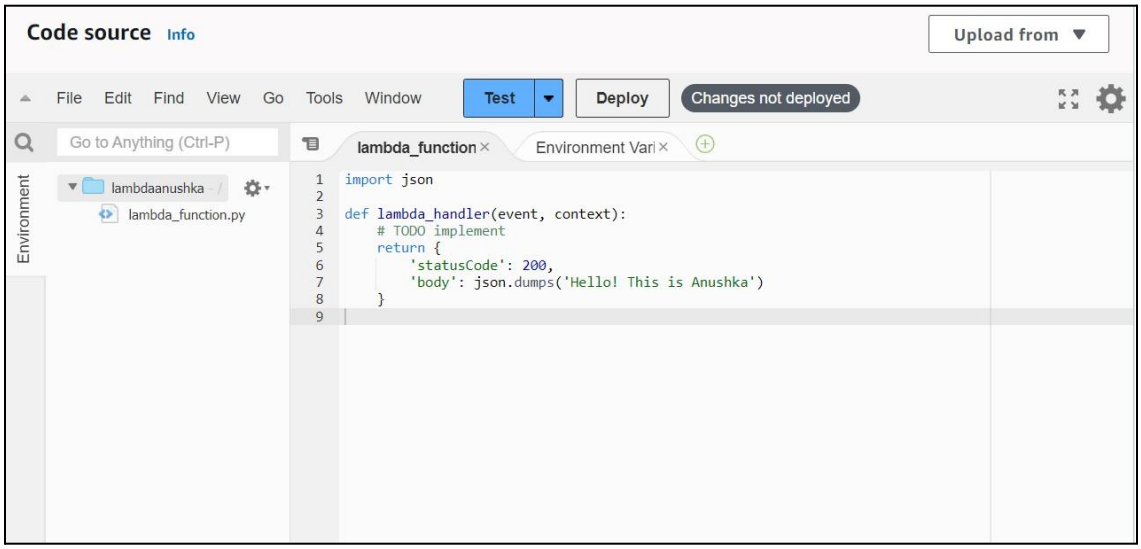
5. You can make changes to your function inside the code editor. You can also upload a zip file of your function or upload one from an S3 bucket if needed.

Press Ctrl + S to save the file and click Deploy to deploy the changes.



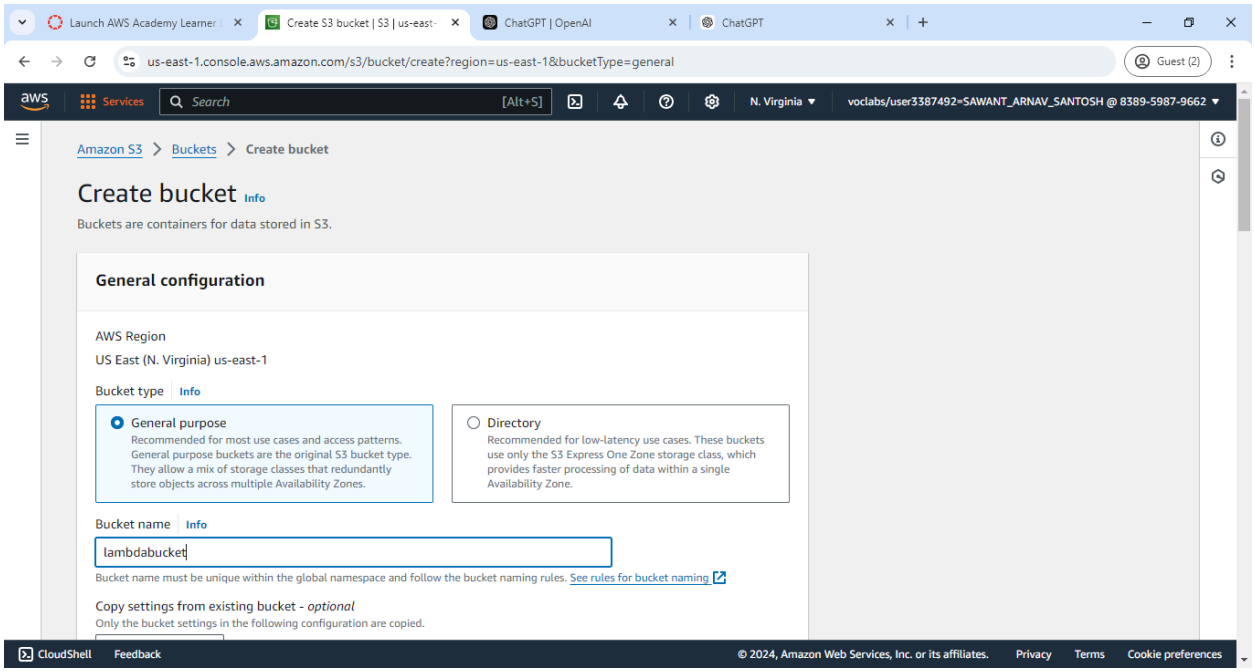
6. Click on Test and you can change the configuration, like so. If you do not have anything in the request body, it is important to specify two curly braces as valid JSON, so make sure they are there.

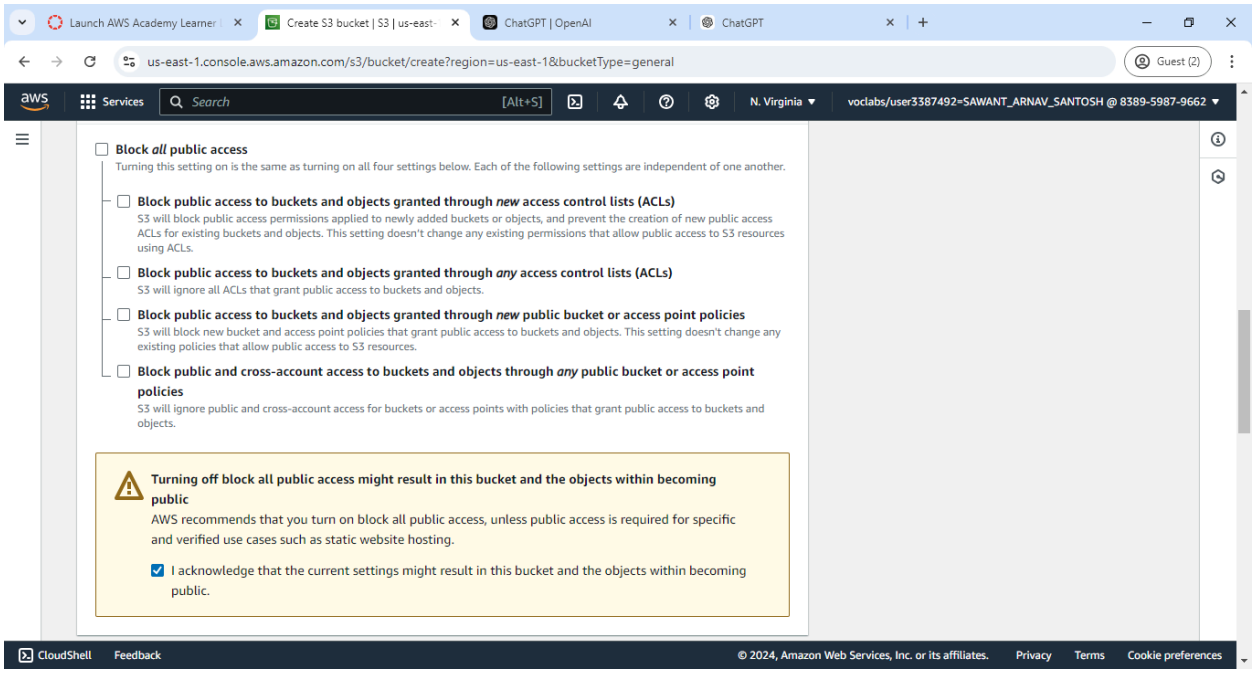
7. Now click on Test and you should be able to see the results.

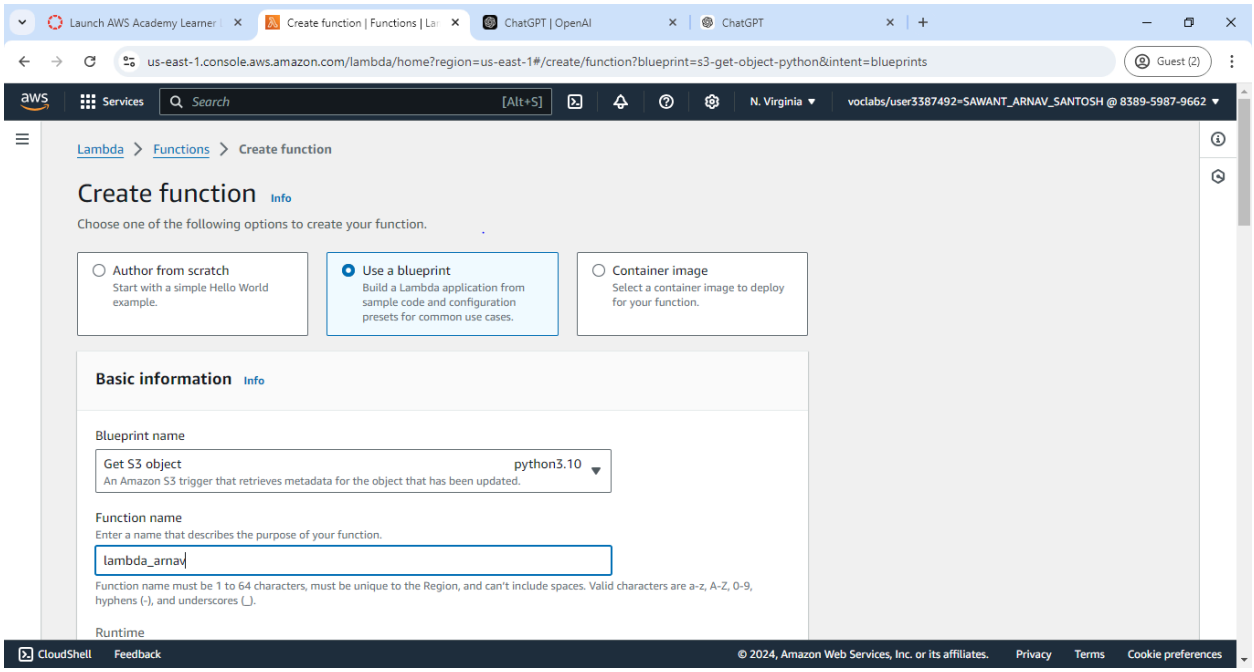


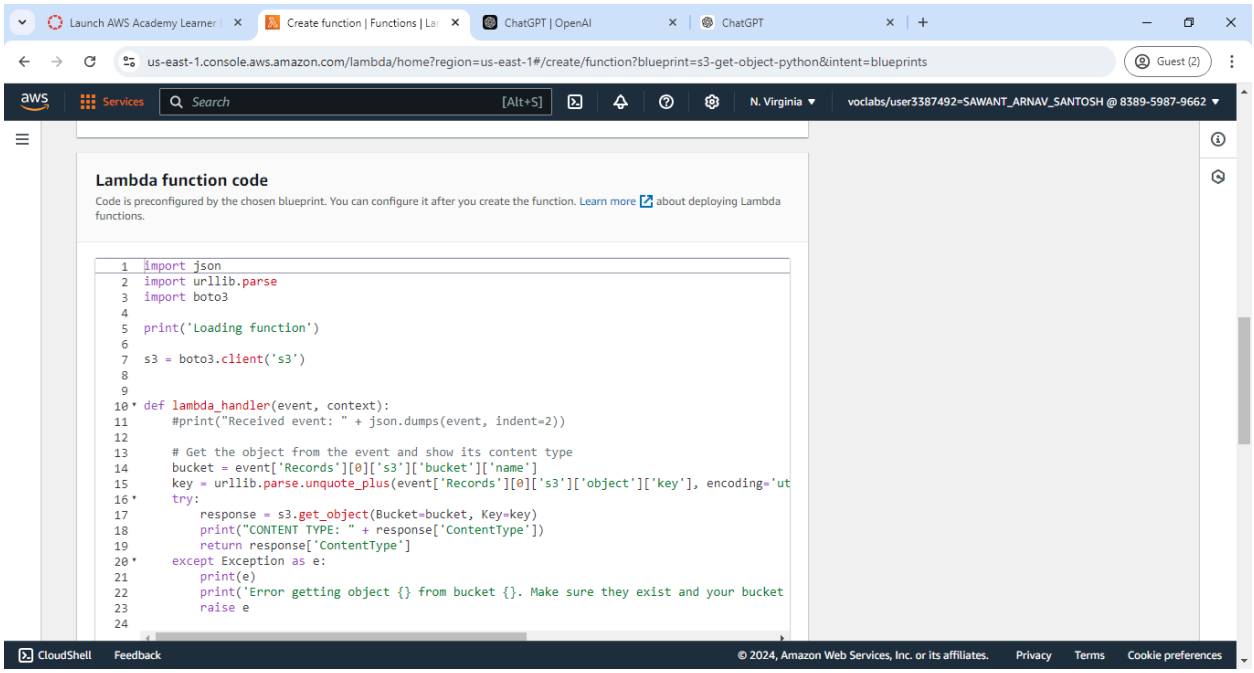
Conclusion:

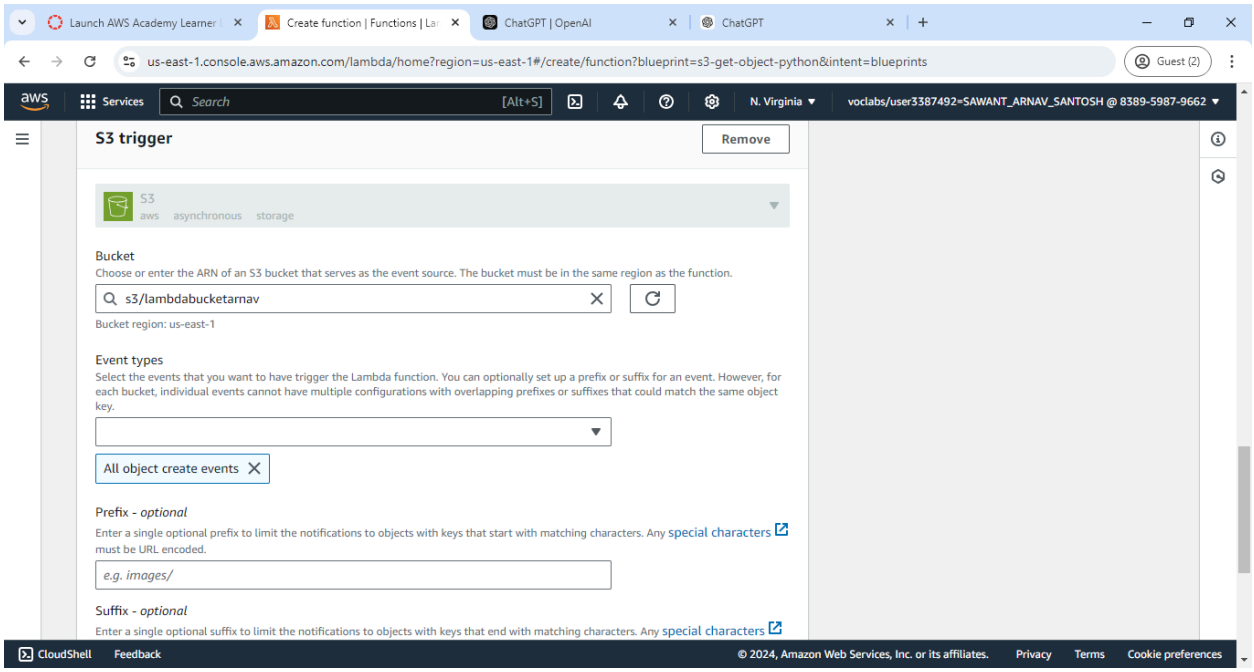
AWS Lambda is a serverless computing service that allows you to run code without managing servers, making it highly scalable, cost-effective, and easy to use. It automatically manages the compute resources, executes your code in response to specific events such as API calls, file uploads, or database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

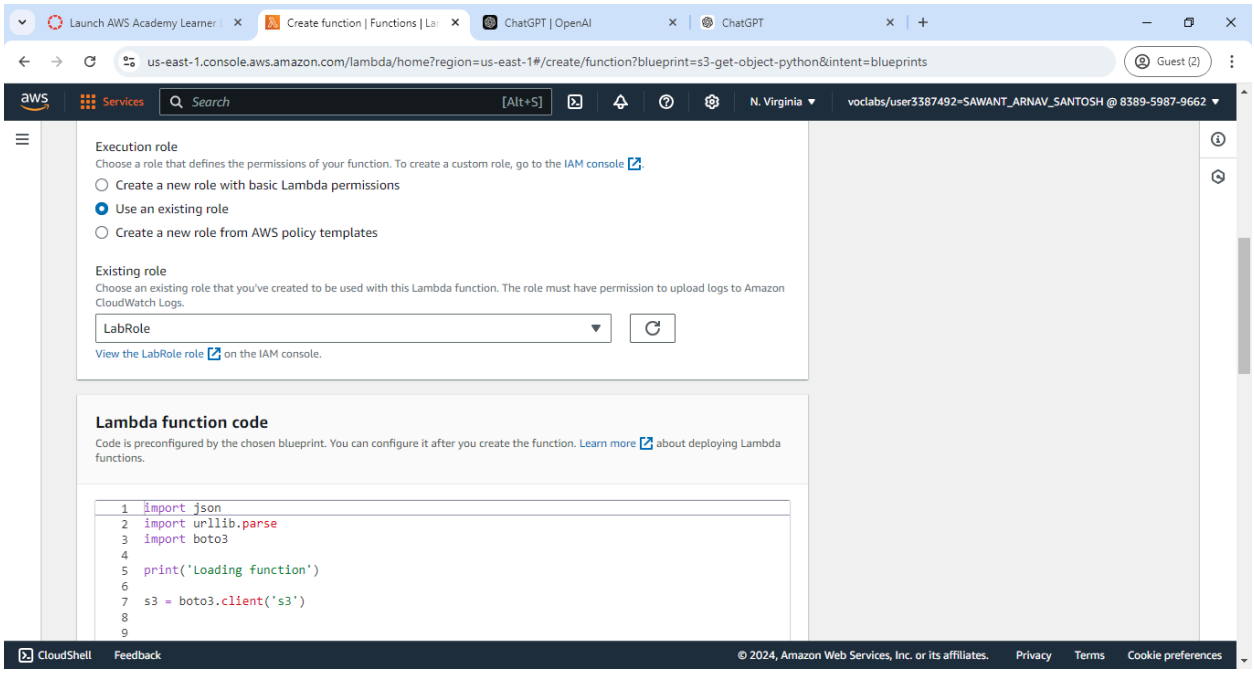

Launch AWS Academy Learner
Create S3 bucket | S3 | us-east-
ChatGPT | OpenAI
ChatGPT
us-east-1.console.aws.amazon.com/s3/bucket/create?region=us-east-1&bucketType=general
Guest (2)
aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3387492=SAWANT_ARNAV_SANTOSH @ 8389-5987-9662
Amazon S3 > Buckets > Create bucket
Create bucket Info
Buckets are containers for data stored in S3.
General configuration
AWS Region
US East (N. Virginia) us-east-1
Bucket type Info
General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.
Bucket name Info
lambdabucket
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming
Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

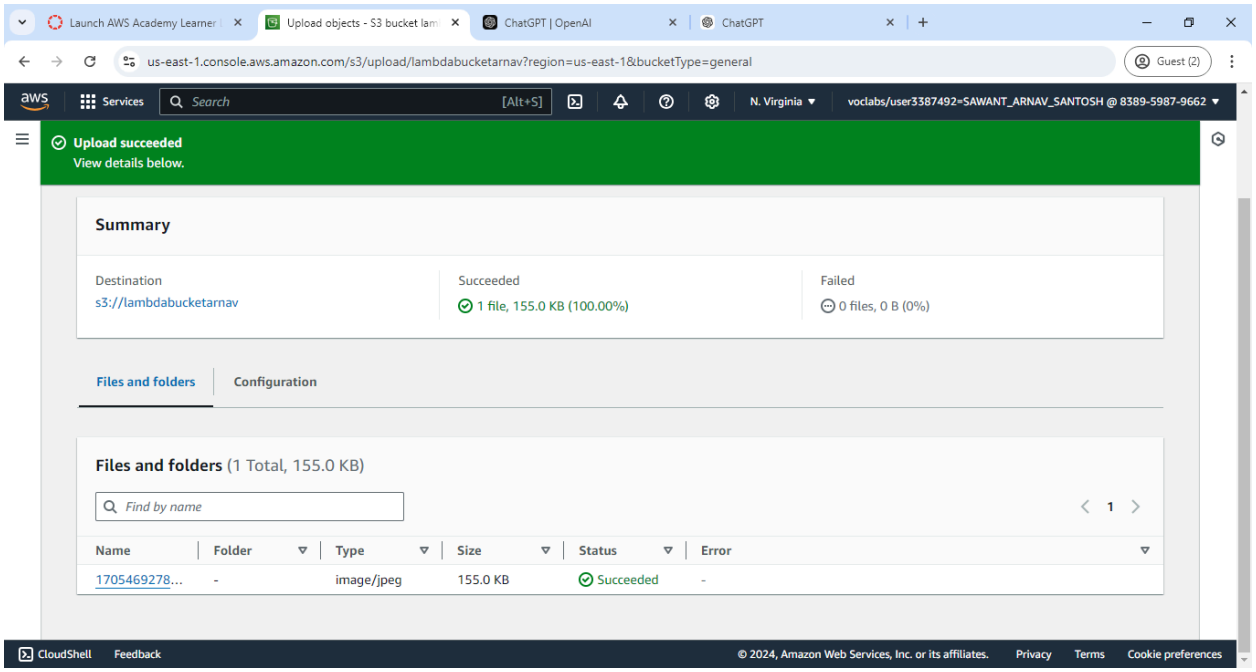

Launch AWS Academy Learner
Create S3 bucket | S3 | us-east-
ChatGPT | OpenAI
ChatGPT
us-east-1.console.aws.amazon.com/s3/bucket/create?region=us-east-1&bucketType=general
Guest (2)
aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3387492=SAWANT_ARNAV_SANTOSH @ 8389-5987-9662
Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
Block public access to buckets and objects granted through new access control lists (ACLs)
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
Block public access to buckets and objects granted through any access control lists (ACLs)
S3 will ignore all ACLs that grant public access to buckets and objects.
Block public access to buckets and objects granted through new public bucket or access point policies
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
Block public and cross-account access to buckets and objects through any public bucket or access point policies
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.
Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.
I acknowledge that the current settings might result in this bucket and the objects within becoming public.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

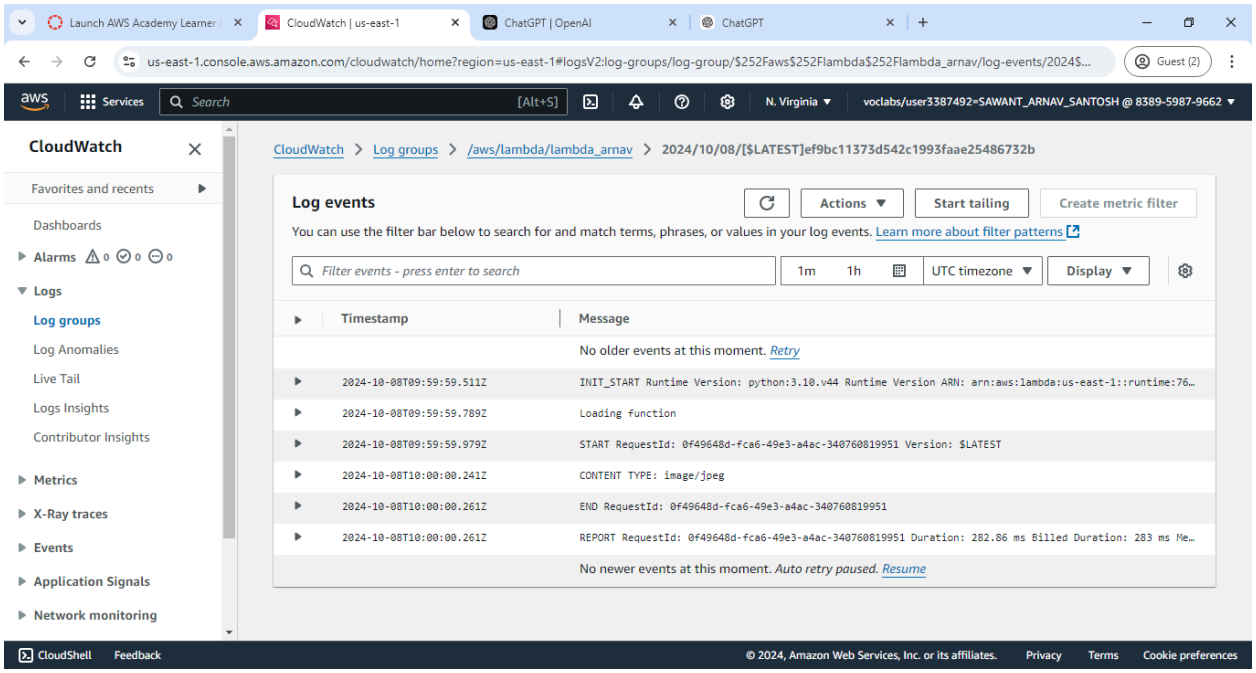

Launch AWS Academy Learner
Create function | Functions | La
ChatGPT | OpenAI
ChatGPT
us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/create/function?blueprint=s3-get-object-python&intent=blueprints
Guest (2)
Services
Search
[Alt+S]
N. Virginia
voclabs/user3387492=SAWANT_ARNAV_SANTOSH @ 8389-5987-9662
Lambda > Functions > Create function
Create function Info
Choose one of the following options to create your function.
Author from scratch
Start with a simple Hello World example.
Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.
Container image
Select a container image to deploy for your function.
Basic information Info
Blueprint name
Get S3 object
python3.10
An Amazon S3 trigger that retrieves metadata for the object that has been updated.
Function name
Enter a name that describes the purpose of your function.
lambda_arnav
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).
Runtime
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Launch AWS Academy Learner
Create function | Functions | La
ChatGPT | OpenAI
ChatGPT
us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/create/function?blueprint=s3-get-object-python&intent=blueprints
Guest (2)
Services
Search
[Alt+S]
N. Virginia
voclabs/user3387492=SAWANT_ARNAV_SANTOSH @ 8389-5987-9662
Lambda function code
Code is preconfigured by the chosen blueprint. You can configure it after you create the function. Learn more about deploying Lambda functions.
import json
import urllib.parse
import boto3
print('Loading function')
s3 = boto3.client('s3')
def lambda_handler(event, context):
#print("Received event: " + json.dumps(event, indent=2))
# Get the object from the event and show its content type
bucket = event['Records'][0]['s3']['bucket']['name']
key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], encoding='ut
try:
response = s3.get_object(Bucket=bucket, Key=key)
print("CONTENT TYPE: " + response['ContentType'])
return response['ContentType']
except Exception as e:
print(e)
print('Error getting object {} from bucket {}. Make sure they exist and your bucket
raise e
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Launch AWS Academy Learner
Create function | Functions | La
ChatGPT | OpenAI
ChatGPT
us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/create/function?blueprint=s3-get-object-python&intent=blueprints
Guest (2)
Services
Search
[Alt+S]
N. Virginia
voclabs/user3387492=SAWANT_ARNAV_SANTOSH @ 8389-5987-9662
S3 trigger
Remove
S3
aws asynchronous storage
Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/lambdabucketarnav
Bucket region: us-east-1
Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
All object create events
Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any special characters must be URL encoded.
e.g. images/
Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any special characters
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Launch AWS Academy Learner
Create function | Functions | La
ChatGPT | OpenAI
ChatGPT
us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/create/function?blueprint=s3-get-object-python&intent=blueprints
Guest (2)
Services
Search
[Alt+S]
N. Virginia
voclabs/user3387492=SAWANT_ARNAV_SANTOSH @ 8389-5987-9662
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
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
LabRole
View the LabRole role on the IAM console.
Lambda function code
Code is preconfigured by the chosen blueprint. You can configure it after you create the function. Learn more about deploying Lambda functions.
1 import json
2 import urllib.parse
3 import boto3
4
5 print('Loading function')
6
7 s3 = boto3.client('s3')
8
9
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Launch AWS Academy Learner
Upload objects - S3 bucket lam
ChatGPT | OpenAI
ChatGPT
us-east-1.console.aws.amazon.com/s3/upload/lambdabucketarnav?region=us-east-1&bucketType=general
Guest (2)
Services
Search
[Alt+S]
N. Virginia
voclabs/user3387492=SAWANT_ARNAV_SANTOSH @ 8389-5987-9662
Upload succeeded
View details below.
Summary
Destination
s3://lambdabucketarnav
Succeeded
1 file, 155.0 KB (100.00%)
Failed
0 files, 0 B (0%)
Files and folders
Configuration
Files and folders (1 Total, 155.0 KB)
Find by name
1
Name
Folder
Type
Size
Status
Error
1705469278...
-
image/jpeg
155.0 KB
Succeeded
-
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Launch AWS Academy Learner
CloudWatch | us-east-1
ChatGPT | OpenAI
ChatGPT
us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#logsV2:log-groups/log-group/$252Faws$252Flambda$252Flambda_arnav/log-events/2024$...
Guest (2)
Services
Search
[Alt+S]
N. Virginia
voclabs/user3387492=SAWANT_ARNAV_SANTOSH @ 8389-5987-9662
CloudWatch
Favorites and recents
Dashboards
Alarms
Logs
Log groups
Log Anomalies
Live Tail
Logs Insights
Contributor Insights
Metrics
X-Ray traces
Events
Application Signals
Network monitoring
CloudWatch > Log groups > /aws/lambda/lambda_arnav > 2024/10/08/[$LATEST]ef9bc11373d542c1993faae25486732b
Log events
Actions
Start tailing
Create metric filter
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns
Filter events - press enter to search
1m
1h
UTC timezone
Display
Timestamp
Message
No older events at this moment. Retry
2024-10-08T09:59:59.511Z
INIT_START Runtime Version: python:3.10.v44 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:76...
2024-10-08T09:59:59.789Z
Loading function
2024-10-08T09:59:59.979Z
START RequestId: 0f49648d-fca6-49e3-a4ac-340760819951 Version: $LATEST
2024-10-08T10:00:00.241Z
CONTENT TYPE: image/jpeg
2024-10-08T10:00:00.261Z
END RequestId: 0f49648d-fca6-49e3-a4ac-340760819951
2024-10-08T10:00:00.261Z
REPORT RequestId: 0f49648d-fca6-49e3-a4ac-340760819951 Duration: 282.86 ms Billed Duration: 283 ms Me...
No newer events at this moment. Auto retry paused. Resume
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences