Vivekanand Education Society's

Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai,, Approved by AICTE & Recognized by Govt. of Maharashtra
Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Department of Information Technology

A.Y. 2024-25

Advance DevOps Lab
Experiment 03

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

Roll No.	42
Name	NAIKWADI YASH SHIVDAS
Class	D15B
Subject	Advance DevOps Lab
LO Mapped	LO1: To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements. LO2: To deploy single and multiple container applications and manage application deployments with rollouts in Kubernetes
Grade:	

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

Theory :

Introduction to Kubernetes

Kubernetes, originally developed by Google and now an open-source project under the Cloud Native Computing Foundation (CNCF), is the leading container orchestration platform. It is designed to automate the deployment, scaling, and management of containerized applications, offering developers and operations teams a robust toolset to handle the complexities introduced by container-based microservices architectures.

Container-Based Microservices Architecture

With the rise of containerization, companies have moved towards microservices architectures, allowing them to deploy applications more efficiently and scale individual services independently. Containers encapsulate an application and its dependencies, ensuring consistency across development, testing, and production environments. However, the large-scale adoption of containers has also introduced new challenges, particularly in terms of managing and orchestrating hundreds or thousands of containers across multiple environments.

The Role of Kubernetes

Kubernetes addresses these challenges by providing a unified platform for managing containerized workloads. It abstracts the underlying infrastructure, enabling developers to focus on building applications while Kubernetes manages the complexities of deployment and scaling.

Key features of Kubernetes include:

- Resource Management: Kubernetes ensures that applications do not exceed resource limits set by the administrator, helping to prevent any single application from monopolizing system resources.
- Load Balancing: Kubernetes automatically distributes network traffic across the various instances of a service, ensuring high availability and reliability.
- Self-Healing: Kubernetes can detect when an application is not functioning correctly and automatically restart it or move it to a healthy node in the cluster.
- Automated Rollouts and Rollbacks: Kubernetes can seamlessly roll out new versions of applications and automatically roll back to a previous version in case of failure.
- Scaling: Kubernetes can automatically scale applications up or down based on traffic and resource utilization.
- Cluster Management: Kubernetes simplifies the process of adding or removing nodes from a cluster, automatically redistributing workloads as necessary.

Kubernetes Cluster Architecture

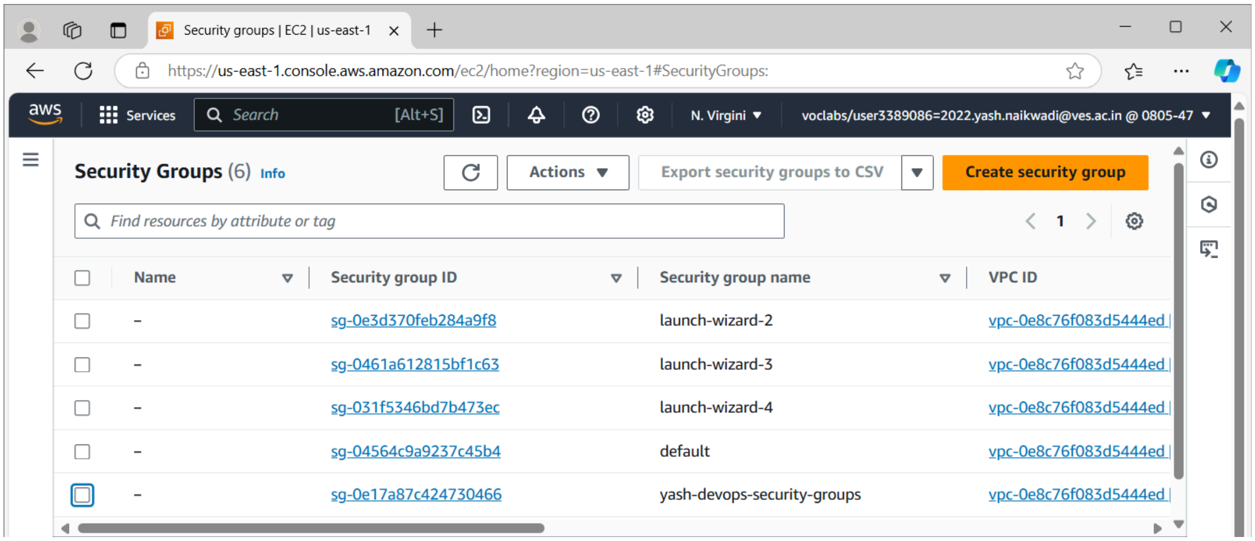
A Kubernetes cluster typically consists of a control plane (master) and worker nodes:

- Control Plane (Master Node): This manages the cluster and contains components like the API server, scheduler, and controller manager. The master node is responsible for maintaining the desired state of the cluster, such as which applications are running and the number of replicas.

Worker Nodes: These nodes run the applications. Each worker node contains components like the Kubelet, which communicates with the control plane, and a container runtime (e.g., Docker) for running the containers.

Create Security Groups for Instances

Log in to your **AWS account** and navigate to Security Groups (in EC2).

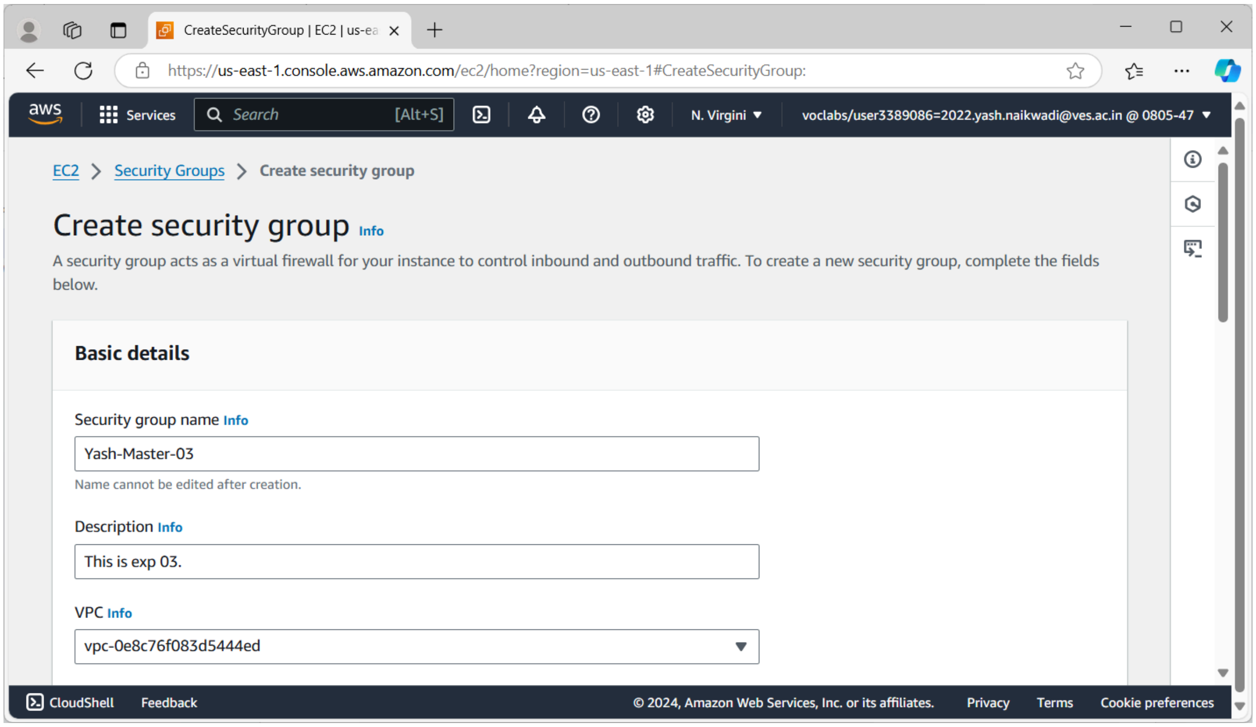


https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#SecurityGroups:

Security Groups (6)

Name	Security group ID	Security group name	VPC ID
–	sg-0e3d370feb284a9f8	launch-wizard-2	vpc-0e8c76f083d5444ed
–	sg-0461a612815bf1c63	launch-wizard-3	vpc-0e8c76f083d5444ed
–	sg-031f5346bd7b473ec	launch-wizard-4	vpc-0e8c76f083d5444ed
–	sg-04564c9a9237c45b4	default	vpc-0e8c76f083d5444ed
–	sg-0e17a87c424730466	yash-devops-security-groups	vpc-0e8c76f083d5444ed

Click on create security group and give the name and description to the security group.



https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateSecurityGroup:

EC2 > Security Groups > Create security group

Create security group

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

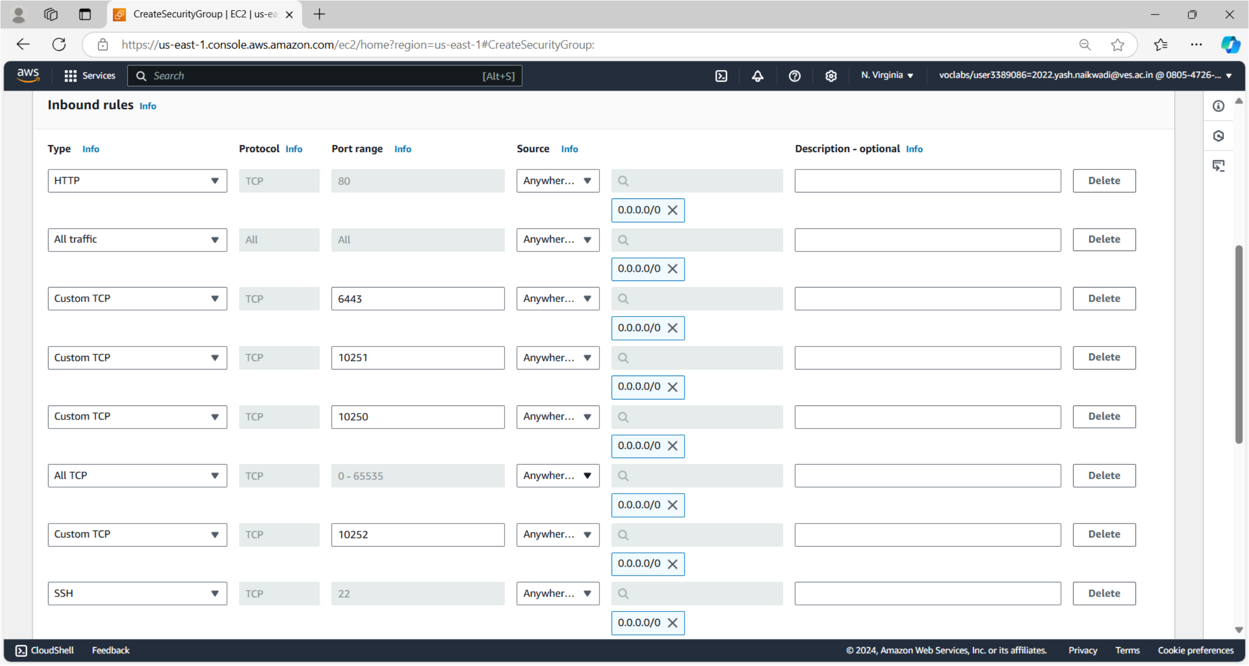
Security group name: Yash-Master-03

Name cannot be edited after creation.

Description: This is exp 03.

VPC: vpc-0e8c76f083d5444ed

Add the following inbound rules to the security group.

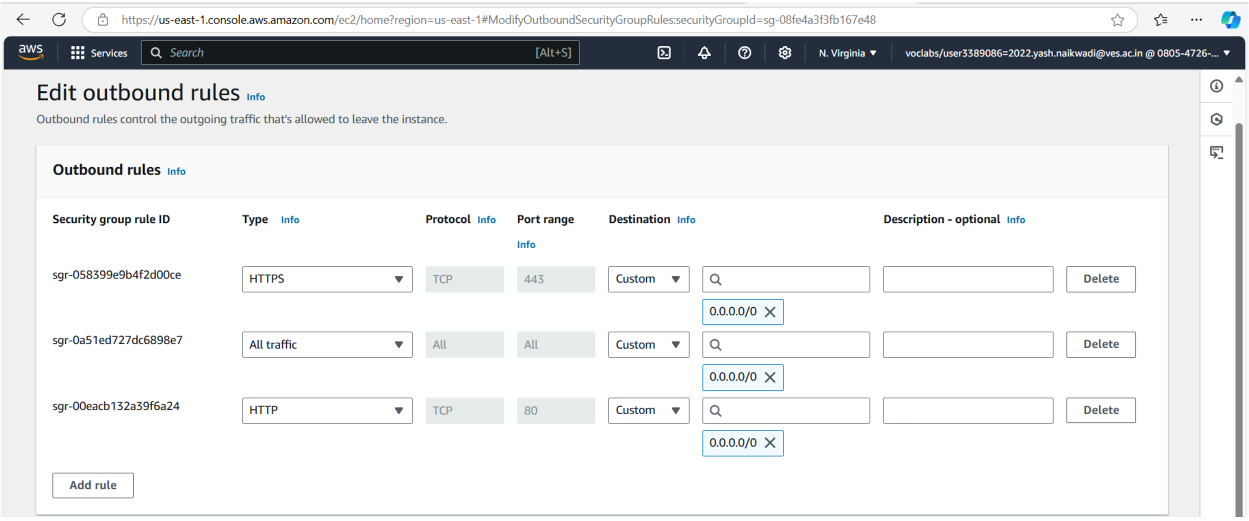


CreateSecurityGroup | EC2 | us-ea
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateSecurityGroup:
Services
Search [Alt+S]
N. Virginia
voclabs/user3389086=2022.yash.naikwadi@ves.ac.in @ 0805-4726-...

Inbound rules Info

Type	Protocol	Port range	Source		Description - optional	
HTTP	TCP	80	Anywher...	0.0.0.0/0		Delete
All traffic	All	All	Anywher...	0.0.0.0/0		Delete
Custom TCP	TCP	6443	Anywher...	0.0.0.0/0		Delete
Custom TCP	TCP	10251	Anywher...	0.0.0.0/0		Delete
Custom TCP	TCP	10250	Anywher...	0.0.0.0/0		Delete
All TCP	TCP	0 - 65535	Anywher...	0.0.0.0/0		Delete
Custom TCP	TCP	10252	Anywher...	0.0.0.0/0		Delete
SSH	TCP	22	Anywher...	0.0.0.0/0		Delete

CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyOutboundSecurityGroupRules:securityGroupId=sg-08fe4a3f3fb167e48
Services
Search [Alt+S]
N. Virginia
voclabs/user3389086=2022.yash.naikwadi@ves.ac.in @ 0805-4726-...

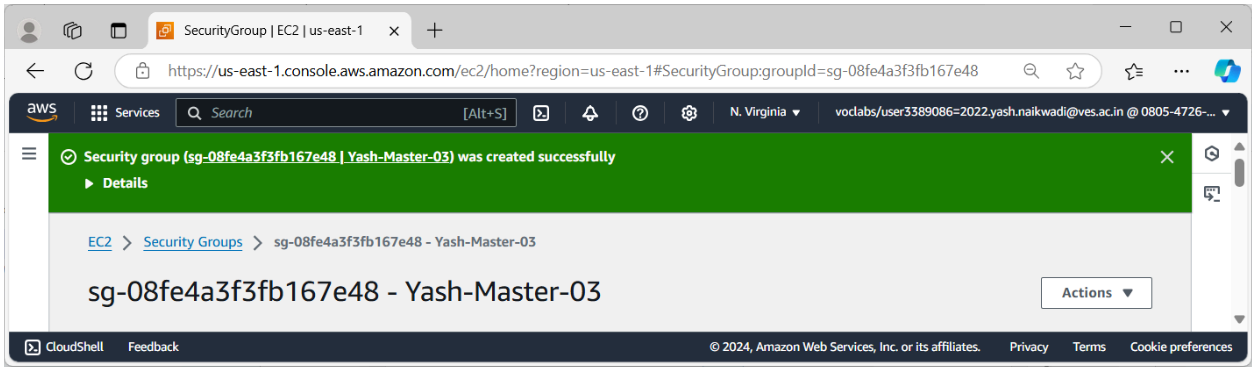
Edit outbound rules Info

Outbound rules control the outgoing traffic that's allowed to leave the instance.

Outbound rules Info

Security group rule ID	Type	Protocol	Port range	Destination		Description - optional	
sgr-058399e9b4f2d00ce	HTTPS	TCP	443	Custom	0.0.0.0/0		Delete
sgr-0a51ed727dc6898e7	All traffic	All	All	Custom	0.0.0.0/0		Delete
sgr-00eacb132a39f6a24	HTTP	TCP	80	Custom	0.0.0.0/0		Delete

Create a security group.



SecurityGroup | EC2 | us-east-1
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#SecurityGroup:groupId=sg-08fe4a3f3fb167e48
Services
Search [Alt+S]
N. Virginia
voclabs/user3389086=2022.yash.naikwadi@ves.ac.in @ 0805-4726-...

Security group (sg-08fe4a3f3fb167e48 | Yash-Master-03) was created successfully

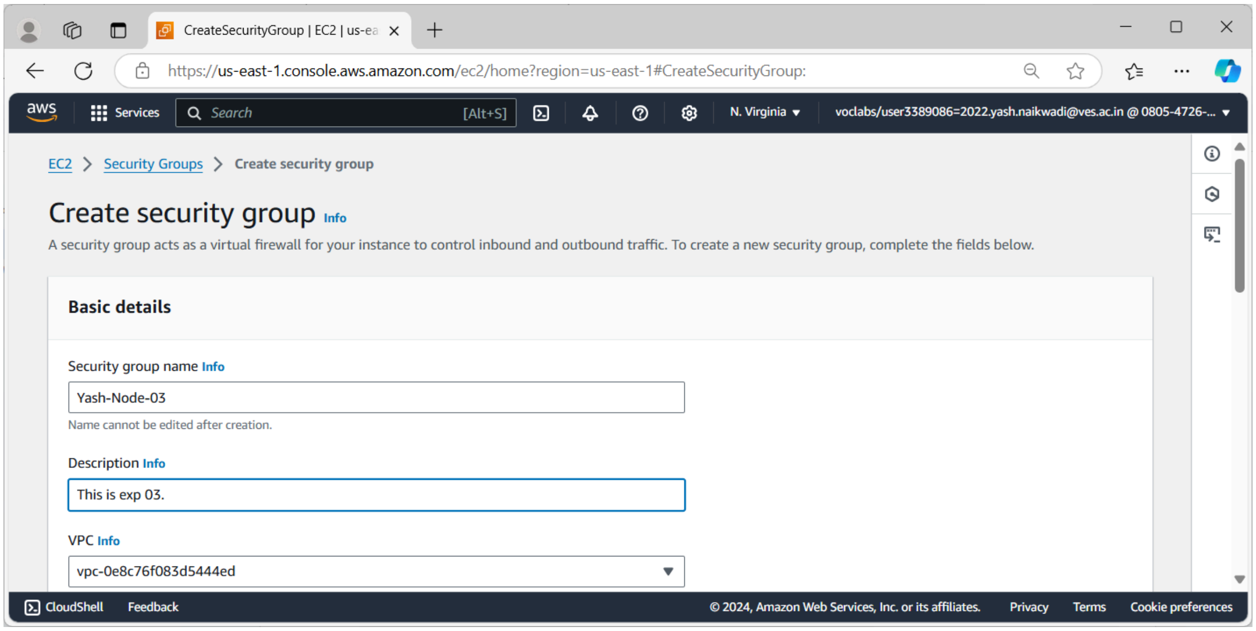
Details

EC2 > Security Groups > sg-08fe4a3f3fb167e48 - Yash-Master-03

sg-08fe4a3f3fb167e48 - Yash-Master-03

Actions
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Similarly, create one more security group (for Node).


CreateSecurityGroup | EC2 | us-ea
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateSecurityGroup:
aws
Services
Search [Alt+S]
N. Virginia
voclabs/user3389096=2022.yash.naikwadi@ves.ac.in @ 0805-4726-...
EC2 > Security Groups > Create security group
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name Info

Yash-Node-03

Name cannot be edited after creation.

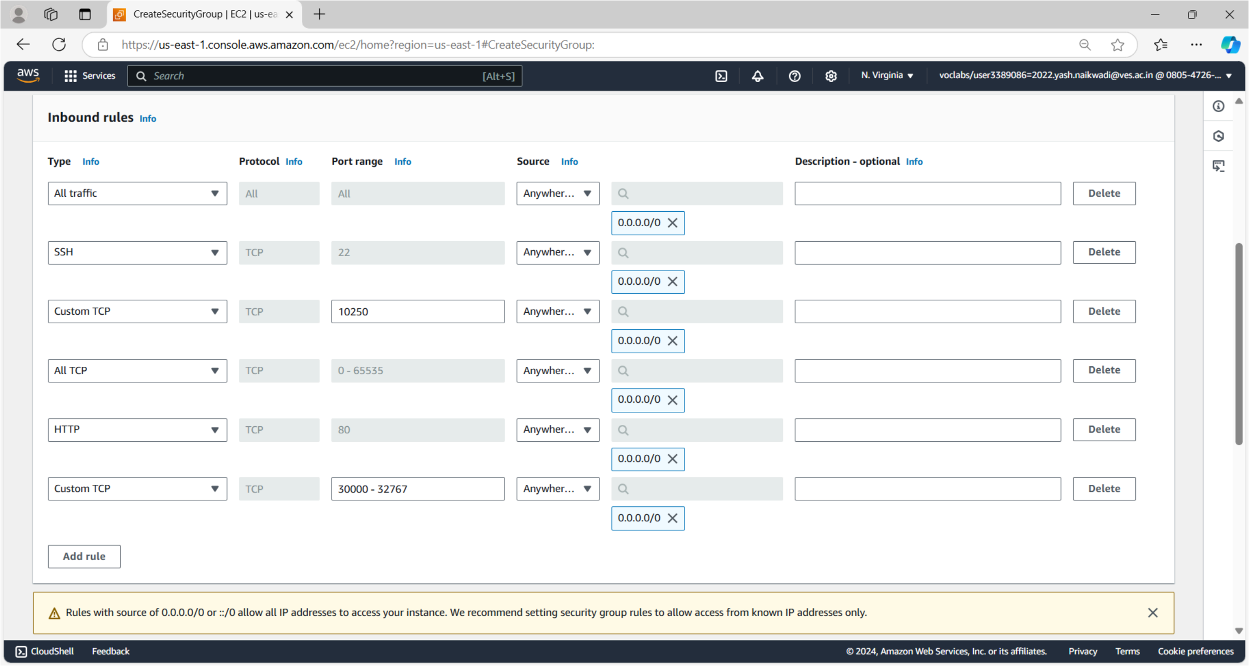
Description Info

This is exp 03.

VPC Info

vpc-0e8c76f083d5444ed

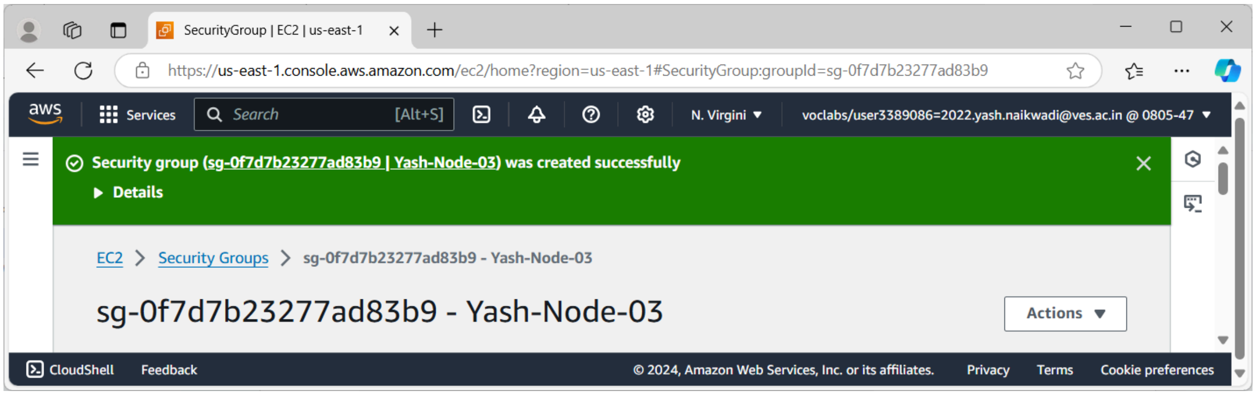
Add the inbound rules for Node security group.


CreateSecurityGroup | EC2 | us-ea
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateSecurityGroup:
aws
Services
Search [Alt+S]
N. Virginia
voclabs/user3389096=2022.yash.naikwadi@ves.ac.in @ 0805-4726-...
Add rule
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Inbound rules Info

Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
All traffic	All	All	Anywher...	0.0.0.0/0		Delete
SSH	TCP	22	Anywher...	0.0.0.0/0		Delete
Custom TCP	TCP	10250	Anywher...	0.0.0.0/0		Delete
All TCP	TCP	0 - 65535	Anywher...	0.0.0.0/0		Delete
HTTP	TCP	80	Anywher...	0.0.0.0/0		Delete
Custom TCP	TCP	30000 - 32767	Anywher...	0.0.0.0/0		Delete

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

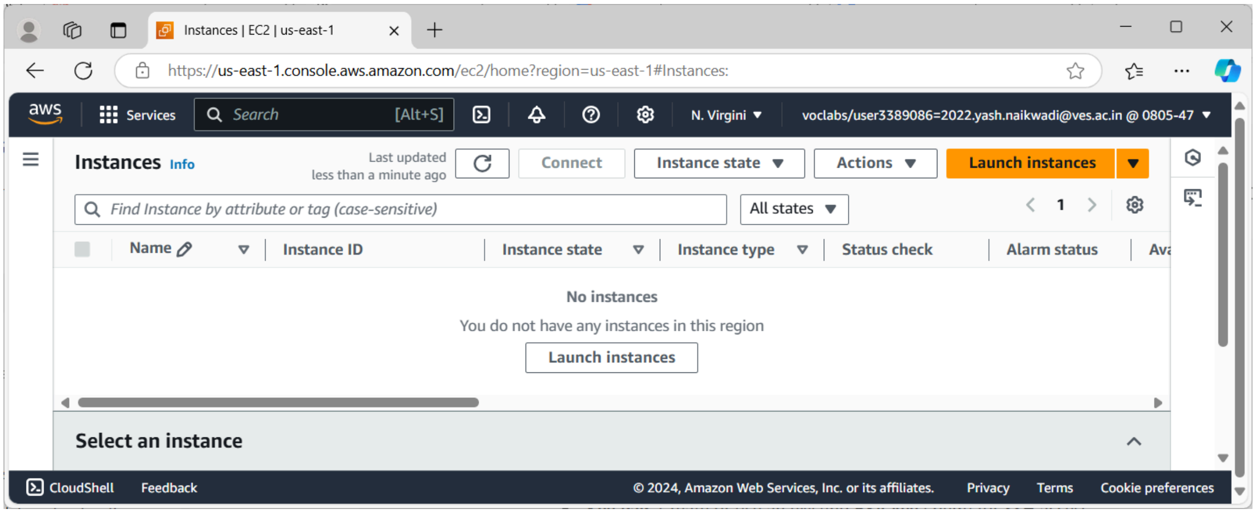

SecurityGroup | EC2 | us-east-1
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#SecurityGroup:groupId=sg-0f7d7b23277ad83b9
aws
Services
Search [Alt+S]
N. Virginia
voclabs/user3389096=2022.yash.naikwadi@ves.ac.in @ 0805-47
EC2 > Security Groups > sg-0f7d7b23277ad83b9 - Yash-Node-03
Actions
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Security group (sg-0f7d7b23277ad83b9 | Yash-Node-03) was created successfully

▶ Details

sg-0f7d7b23277ad83b9 - Yash-Node-03

Navigate to **EC2**.


https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:

Instances Info

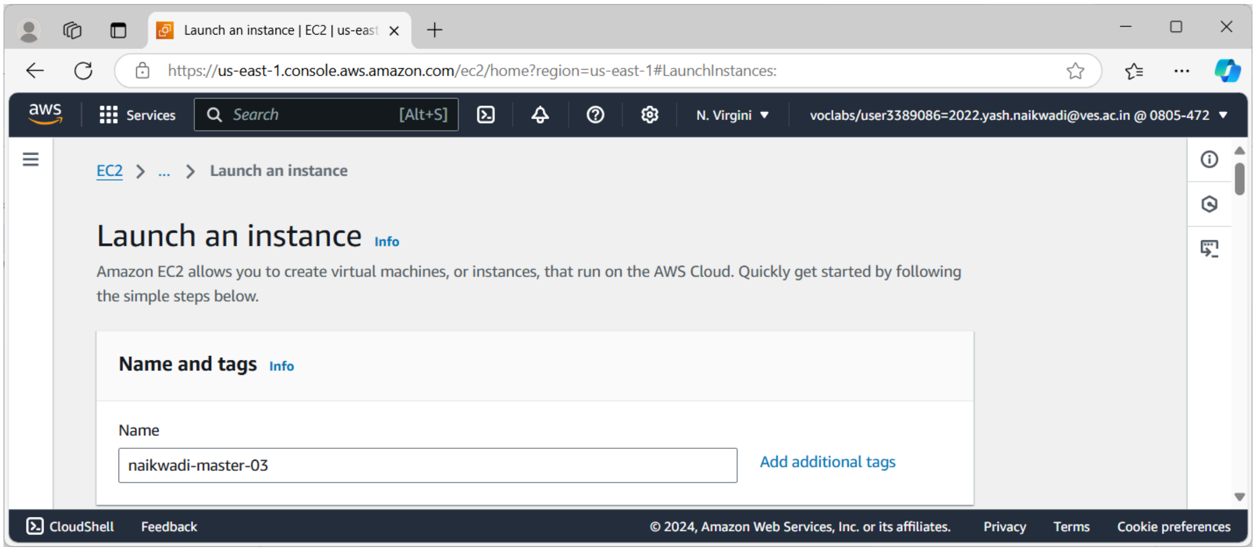
No instances

You do not have any instances in this region

Launch EC2 Instances

MASTER

Launch **a new EC2 instance for master** with the following settings.

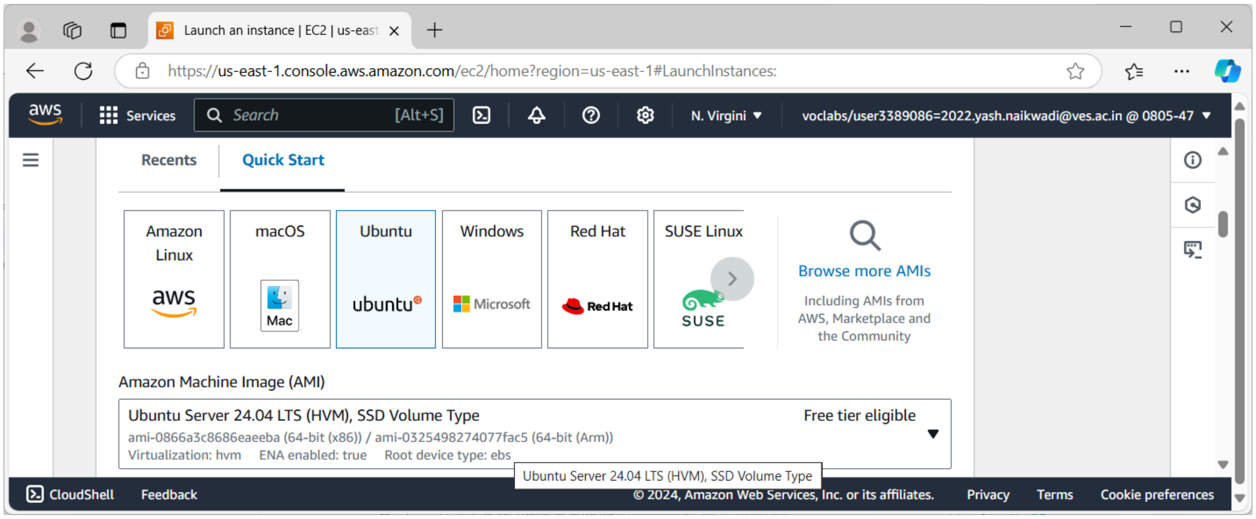

https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:

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: naikwadi-master-03

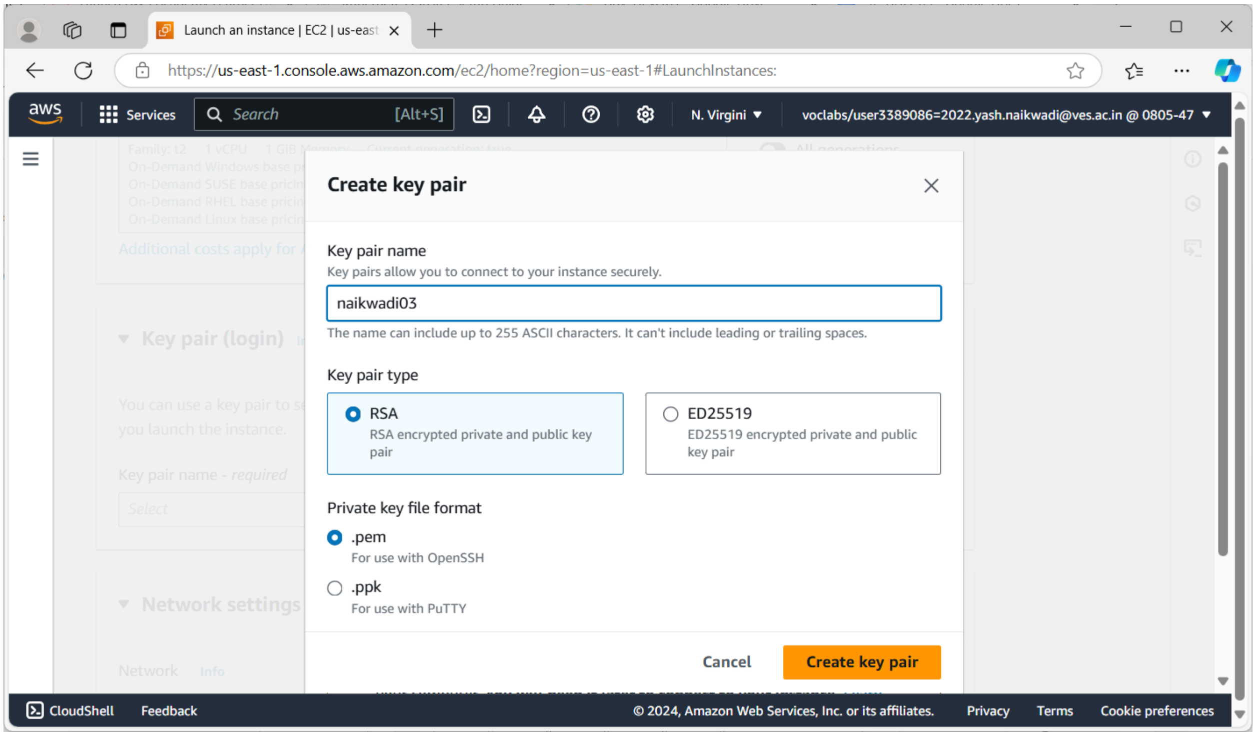

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type — Free tier eligible

ami-0866a3c8686eaeeba (64-bit (x86)) / ami-0325498274077fac5 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Generate the .pem file (or key-value pair) once while configuring the master ec2.


Launch an instance | EC2 | us-east
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
aws
Services
Search [Alt+S]
N. Virgini
voclabs/user3389086=2022.yash.naikwadi@ves.ac.in @ 0805-47
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

naikwadi03

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

● RSA
RSA encrypted private and public key pair

○ ED25519
ED25519 encrypted private and public key pair

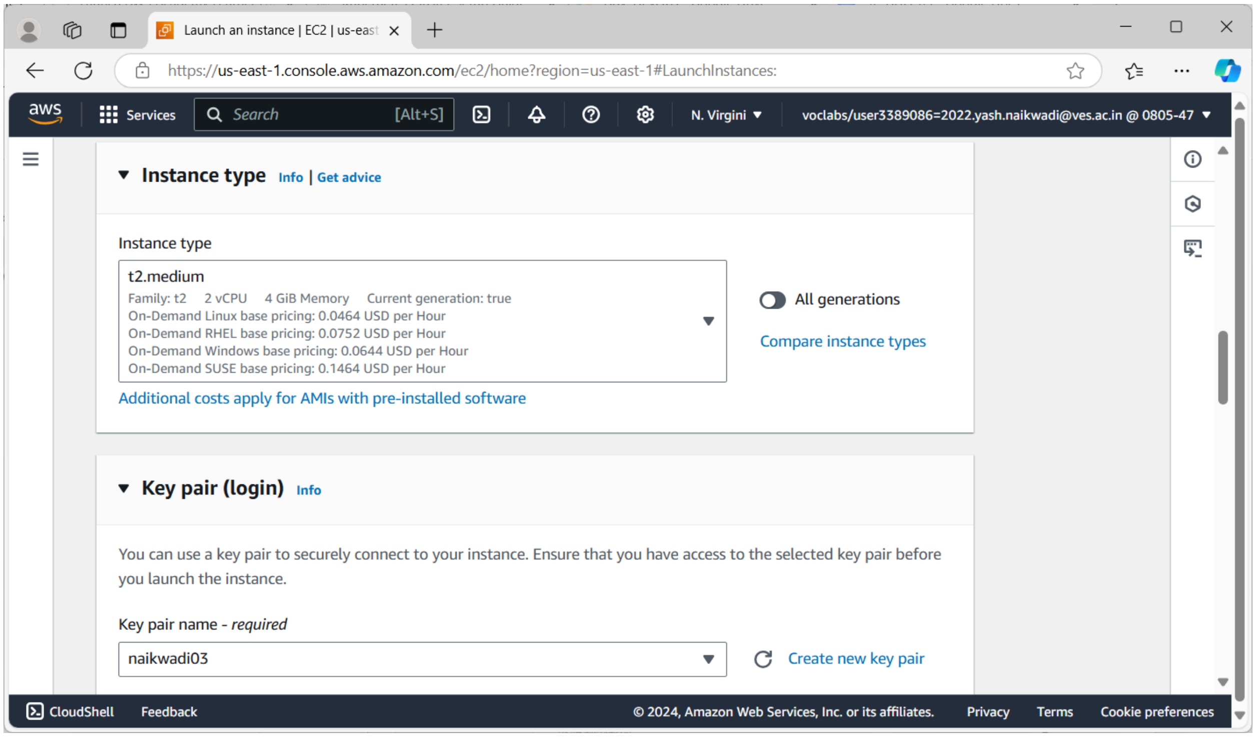
Private key file format

● .pem
For use with OpenSSH

○ .ppk
For use with PuTTY

Cancel | Create key pair

Keep the instance type as t2.medium


Launch an instance | EC2 | us-east
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
aws
Services
Search [Alt+S]
N. Virgini
voclabs/user3389086=2022.yash.naikwadi@ves.ac.in @ 0805-47
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

▼ **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

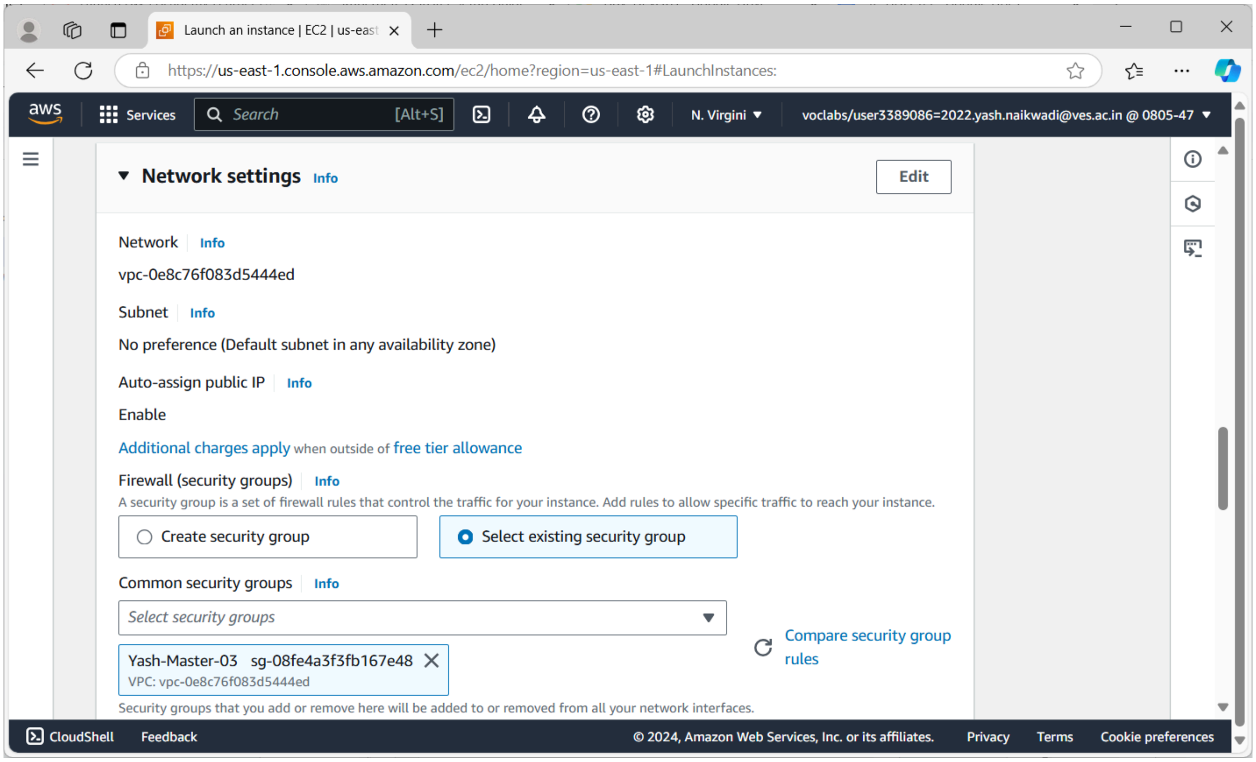
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

naikwadi03

Create new key pair

Select the existing security group under the network setting tab in which you have to select the group of master which was made in the beginning of the experiment.


Launch an instance | EC2 | us-east
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
aws
Services
Search [Alt+S]
N. Virgini ▼
voclabs/user3389086=2022.yash.naikwadi@ves.ac.in @ 0805-47 ▼
Edit
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

▼ **Network settings** Info

Network Info

vpc-0e8c76f083d5444ed

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

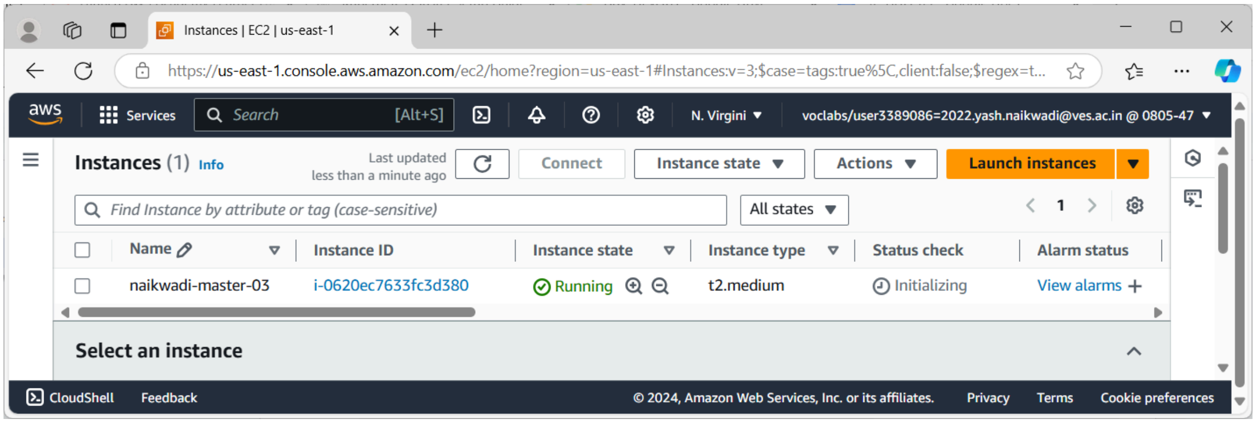
Select security groups ▼

Yash-Master-03 sg-08fe4a3f3fb167e48 ✕
VPC: vpc-0e8c76f083d5444ed

Compare security group rules

Security groups that you add or remove here will be added to or removed from all your network interfaces.

Successfully launched the ec2 instance.


Instances | EC2 | us-east-1
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:v=3;$case=tags:true%5C,client:false;$regex=t...
aws
Services
Search [Alt+S]
N. Virgini ▼
voclabs/user3389086=2022.yash.naikwadi@ves.ac.in @ 0805-47 ▼
Connect
Instance state ▼
Actions ▼
Launch instances ▼
Find Instance by attribute or tag (case-sensitive)
All states ▼
1
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Instances (1) Info

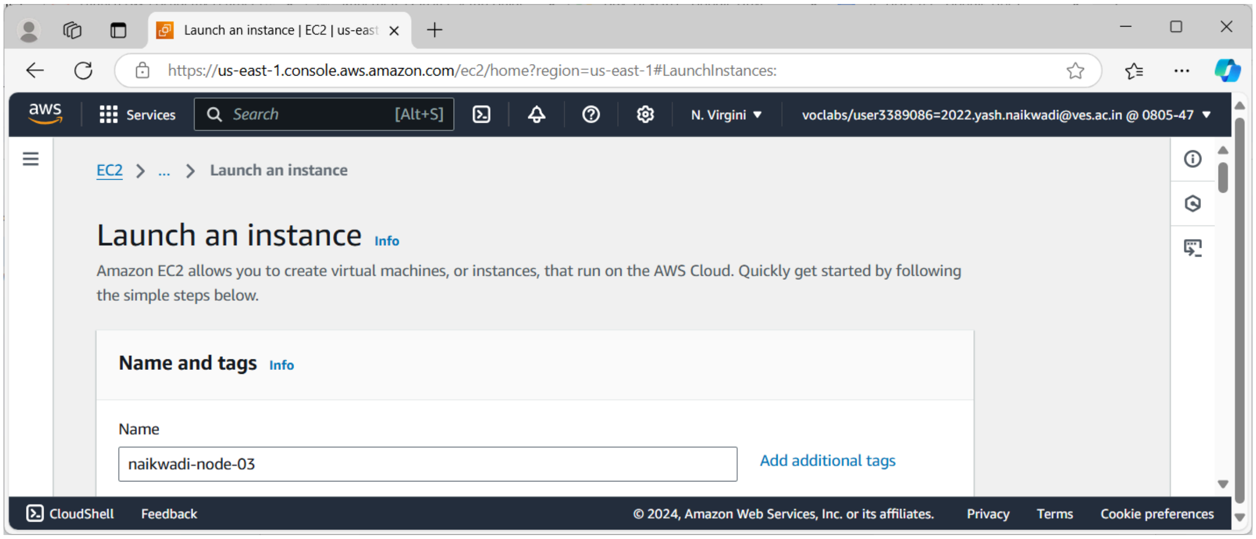
Last updated less than a minute ago

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
□	naikwadi-master-03	i-0620ec7633fc3d380	✓ Running	t2.medium	Initializing	View alarms +

Select an instance

NODE

Similarly, launch 2 new ec2 instances for nodes.


Launch an instance | EC2 | us-east
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
aws
Services
Search [Alt+S]
N. Virgini ▼
voclabs/user3389086=2022.yash.naikwadi@ves.ac.in @ 0805-47 ▼
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

EC2 > ... > Launch an instance

Launch an instance Info

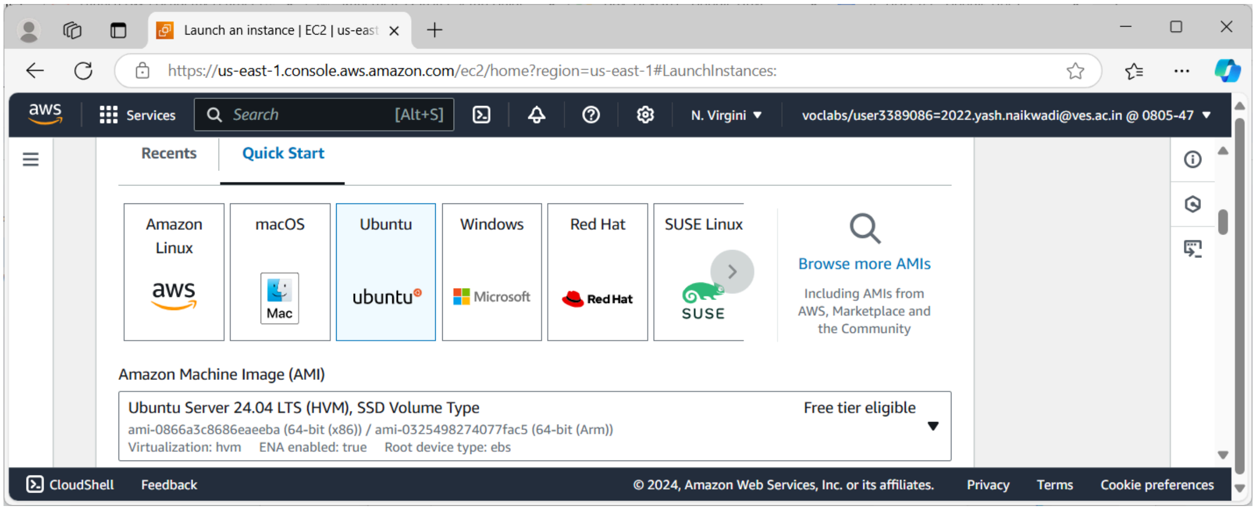
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

naikwadi-node-03

Add additional tags

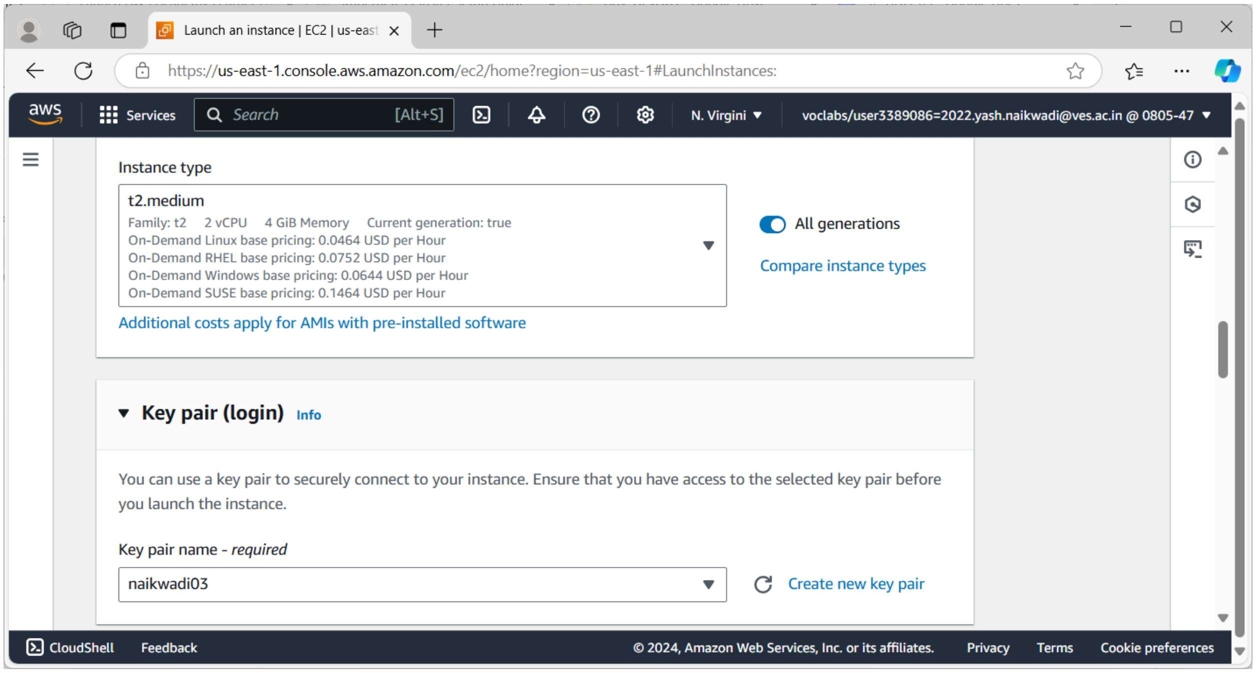


Launch an instance | EC2 | us-east
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
aws
Services
Search [Alt+S]
N. Virgini
voclabs/user3389086=2022.yash.naikwadi@ves.ac.in @ 0805-47
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Linux
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-0866a3c8686eaeeba (64-bit (x86)) / ami-0325498274077fac5 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Keep the instance type of node as t2.medium and select the same key-value pair which was created during the master configuration.



Launch an instance | EC2 | us-east
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
aws
Services
Search [Alt+S]
N. Virgini
voclabs/user3389086=2022.yash.naikwadi@ves.ac.in @ 0805-47
All generations
Compare instance types
Additional costs apply for AMIs with pre-installed software
Info
Create new key pair
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

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

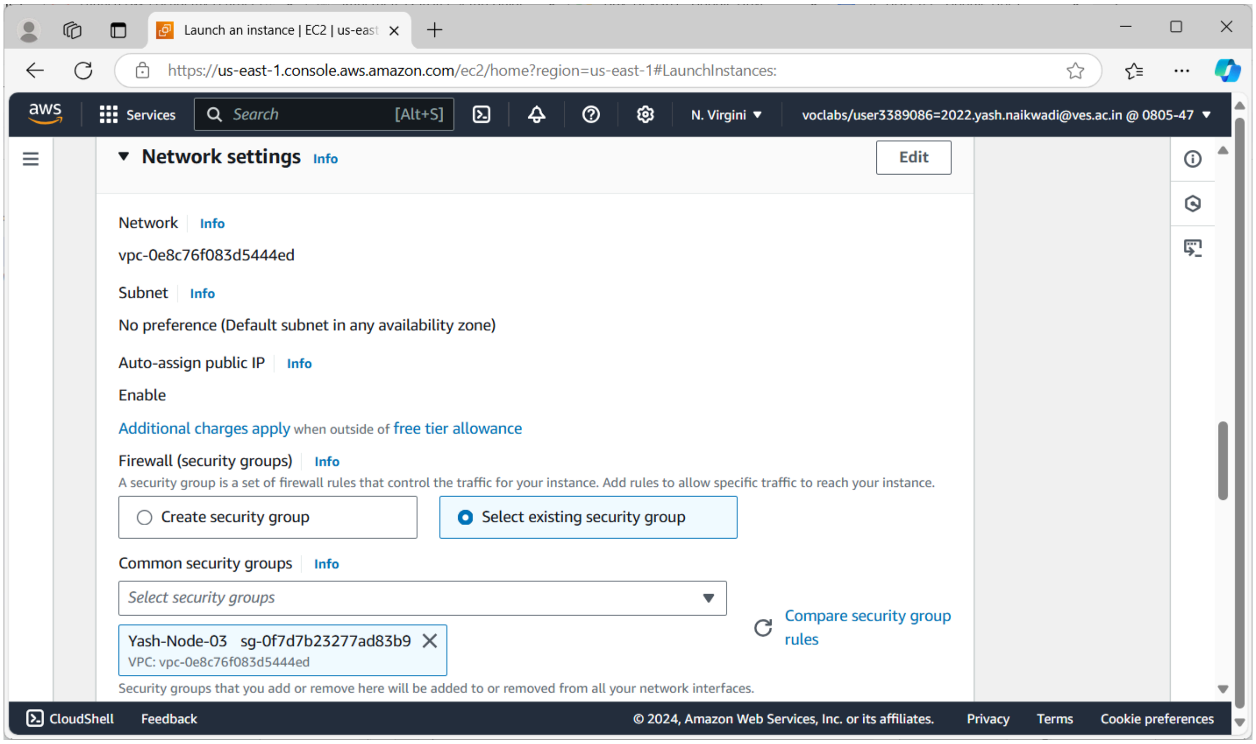
Key pair (login)

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*

naikwadi03

Select the existing security group which was made for Node in the beginning of the experiment.



https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:

N. Virgini ▾ voclabs/user3389086=2022.yash.naikwadi@ves.ac.in @ 0805-47 ▾

▾ **Network settings** Info

Network Info

vpc-0e8c76f083d5444ed

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 ▾

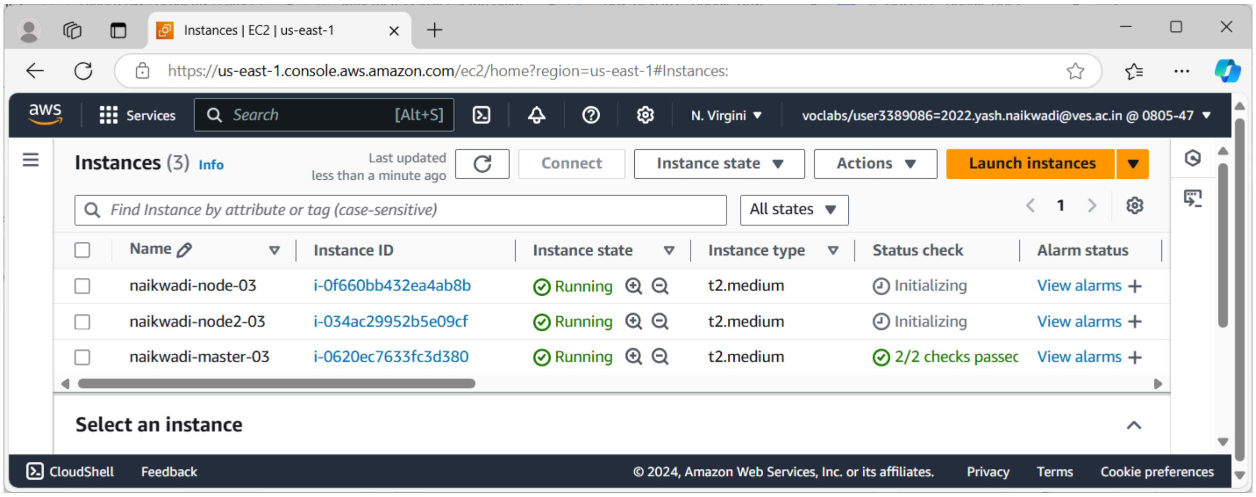
Yash-Node-03 sg-0f7d7b23277ad83b9 ✕
VPC: vpc-0e8c76f083d5444ed

Compare security group rules

Security groups that you add or remove here will be added to or removed from all your network interfaces.

Launch an instance | EC2 | us-east
aws
Services
Search [Alt+S]
Edit
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Successfully, created the 3 ec2 instances (1 for master & 2 for nodes).



https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:

N. Virgini ▾ voclabs/user3389086=2022.yash.naikwadi@ves.ac.in @ 0805-47 ▾

Instances (3) Info Last updated less than a minute ago

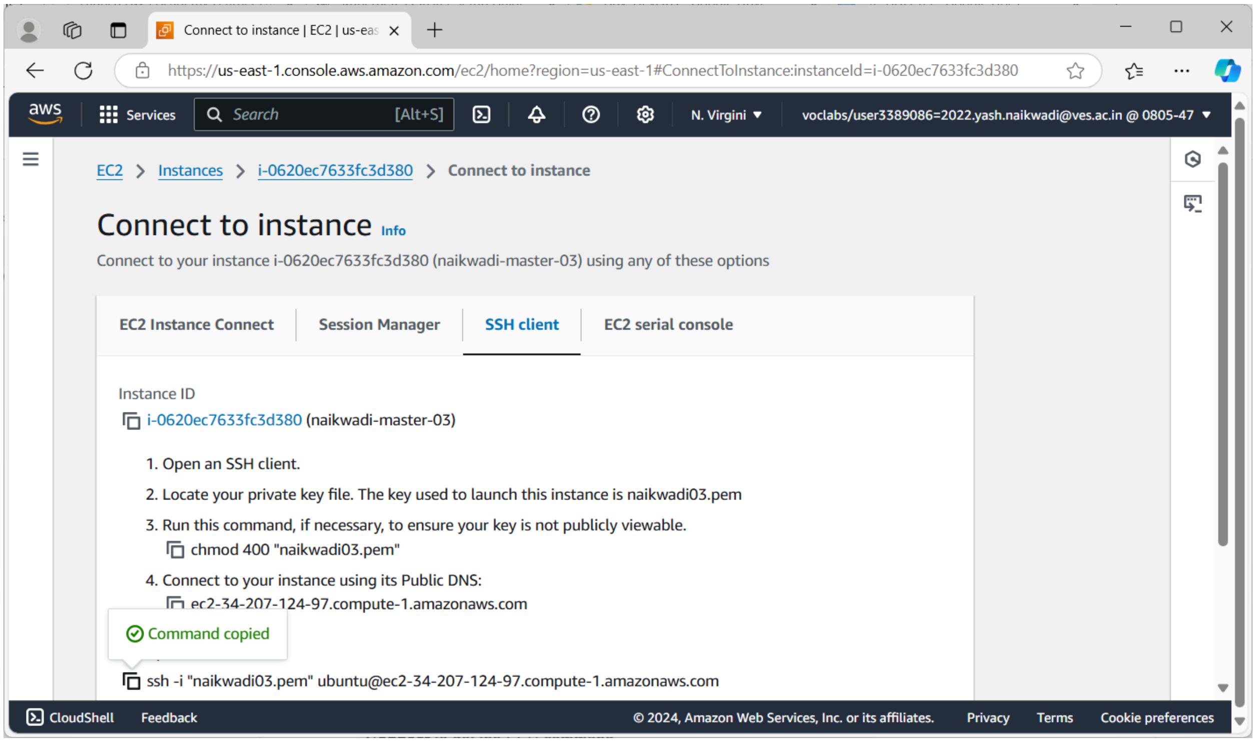
Find Instance by attribute or tag (case-sensitive) All states ▾

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
□	naikwadi-node-03	i-0f660bb432ea4ab8b	✓ Running	t2.medium	Initializing	View alarms +
□	naikwadi-node2-03	i-034ac29952b5e09cf	✓ Running	t2.medium	Initializing	View alarms +
□	naikwadi-master-03	i-0620ec7633fc3d380	✓ Running	t2.medium	✓ 2/2 checks passec	View alarms +

Select an instance

Connect to the Instances

After launching the instances, go to the **EC2 dashboard**, select each instance, and click **Connect** to get the SSH command.



Connect to instance | EC2 | us-eas
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ConnectToInstance:instanceId=i-0620ec7633fc3d380
aws
Services
Search [Alt+S]
N. Virgini
voclabs/user3389086=2022.yash.naikwadi@ves.ac.in @ 0805-47
EC2 > Instances > i-0620ec7633fc3d380 > Connect to instance

Connect to instance Info

Connect to your instance i-0620ec7633fc3d380 (naikwadi-master-03) using any of these options

EC2 Instance Connect
Session Manager
SSH client
EC2 serial console

Instance ID

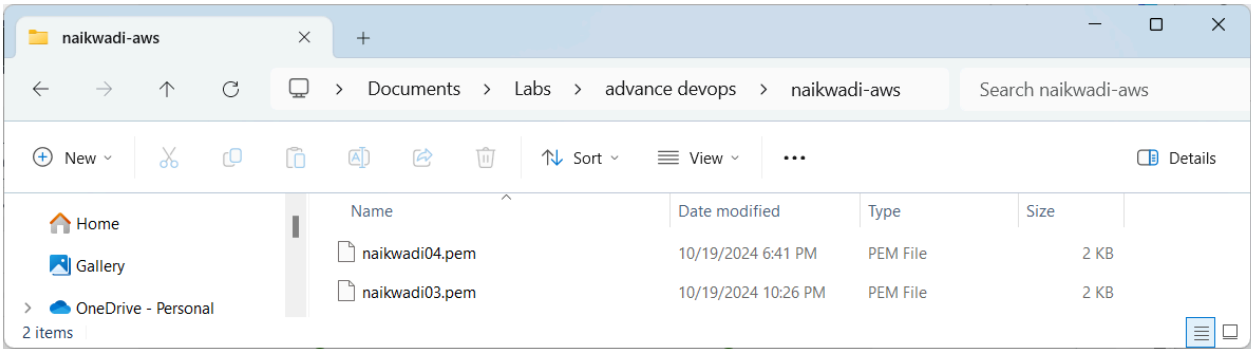
i-0620ec7633fc3d380 (naikwadi-master-03)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is naikwadi03.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "naikwadi03.pem"
4. Connect to your instance using its Public DNS:
   ec2-34-207-124-97.compute-1.amazonaws.com

Command copied

ssh -i "naikwadi03.pem" ubuntu@ec2-34-207-124-97.compute-1.amazonaws.com

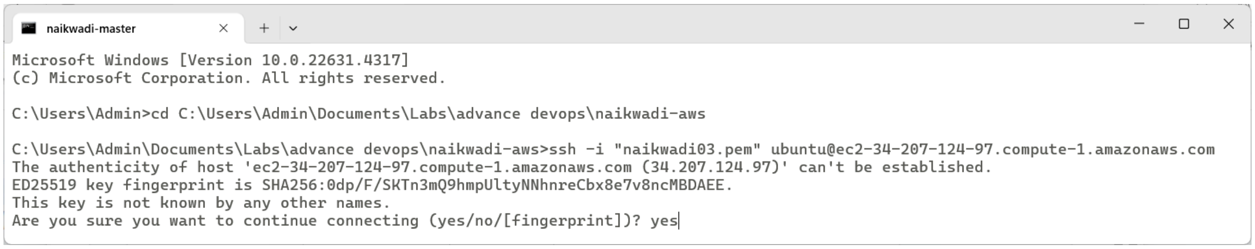
Locate the folder in which the .pem file is placed.



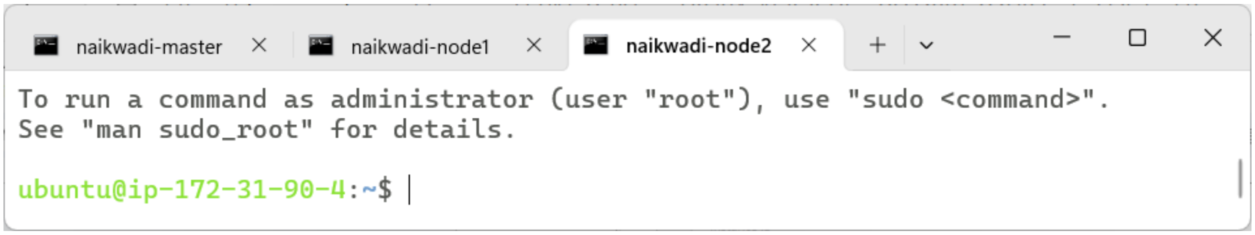
Name	Date modified	Type	Size
naikwadi04.pem	10/19/2024 6:41 PM	PEM File	2 KB
naikwadi03.pem	10/19/2024 10:26 PM	PEM File	2 KB

Open your terminal, navigate to the folder where your **.pem key** is stored, and connect to each instance using:

ssh -i "your-key.pem" ubuntu@your-ec2-public-ip



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd C:\Users\Admin\Documents\Labs\advance devops\naikwadi-aws

C:\Users\Admin\Documents\Labs\advance devops\naikwadi-aws>ssh -i "naikwadi03.pem" ubuntu@ec2-34-207-124-97.compute-1.amazonaws.com
The authenticity of host 'ec2-34-207-124-97.compute-1.amazonaws.com (34.207.124.97)' can't be established.
ED25519 key fingerprint is SHA256:0dp/F/SKTn3mQ9hmpUltyNNhnreCbx8e7v8ncMBDAEE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```



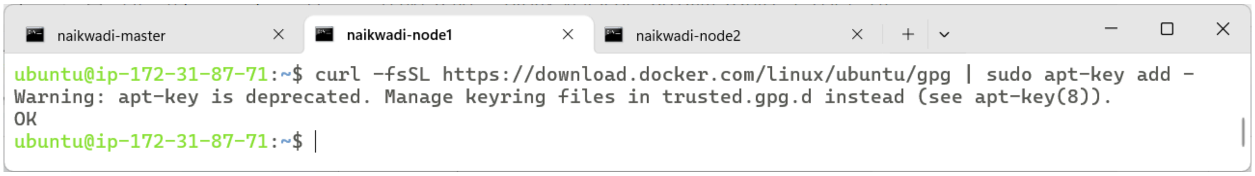
```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-90-4:~$
```

Install Docker on All Instances (Master and Nodes)

On each instance (Master, Node 1, Node 2), run the following commands to install Docker:

# Add Docker's official GPG key

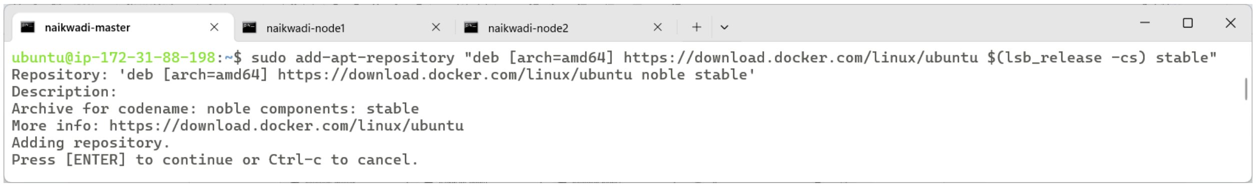
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -



```
ubuntu@ip-172-31-87-71:~$ 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-172-31-87-71:~$ |
```

# Add Docker's APT repository

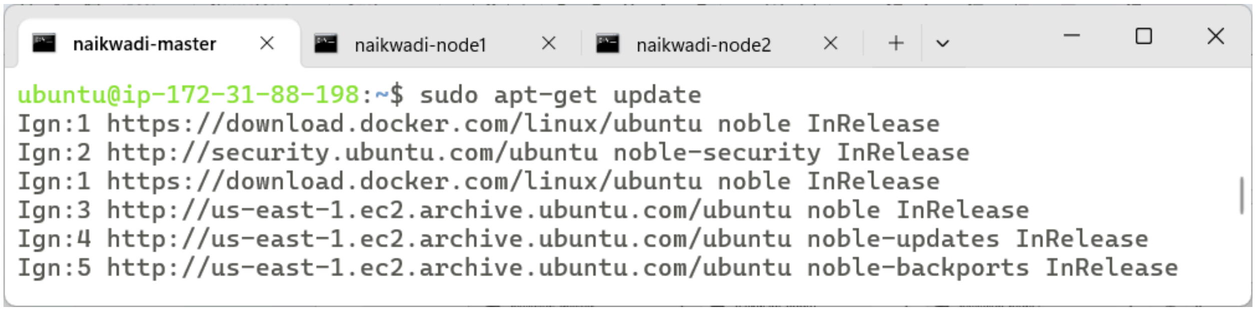
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"



```
ubuntu@ip-172-31-88-198:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
```

# Update the package index

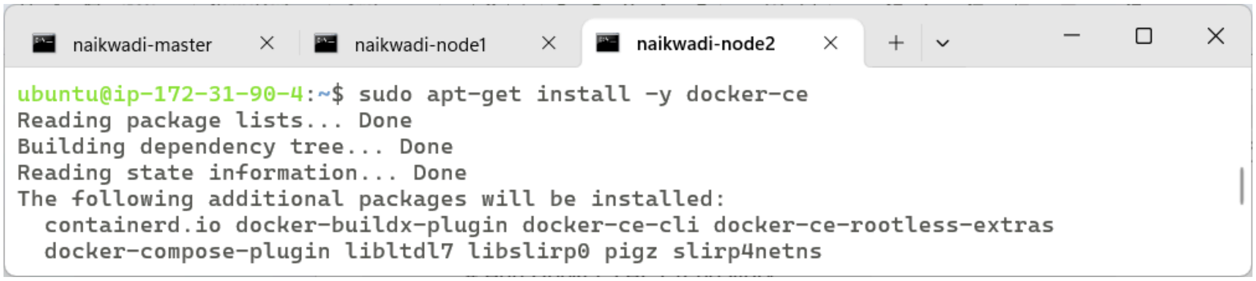
sudo apt-get update



```
ubuntu@ip-172-31-88-198:~$ sudo apt-get update
Ign:1 https://download.docker.com/linux/ubuntu noble InRelease
Ign:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:1 https://download.docker.com/linux/ubuntu noble InRelease
Ign:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Ign:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Ign:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
```

# Install Docker CE (Community Edition)

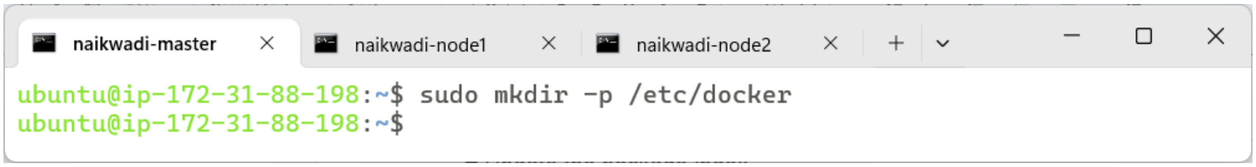
sudo apt-get install -y docker-ce



```
ubuntu@ip-172-31-90-4:~$ sudo apt-get install -y docker-ce
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
```

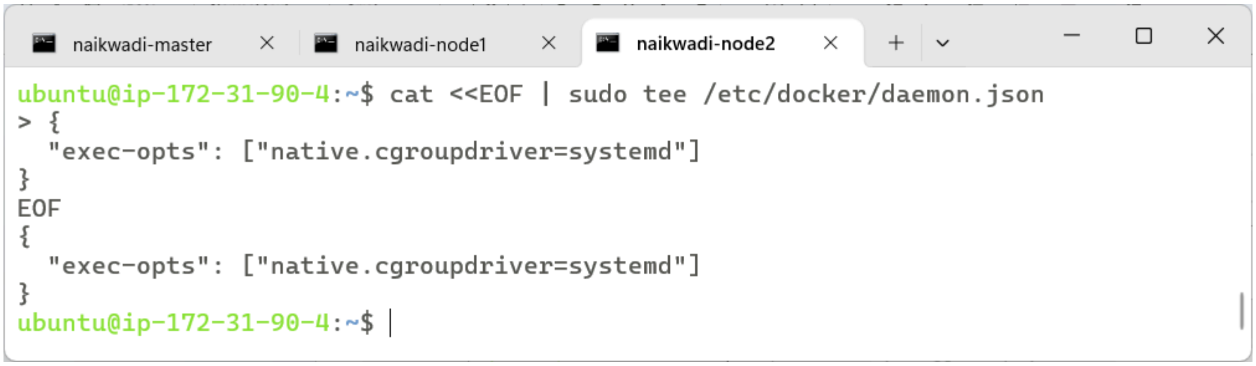
# Configure Docker to use systemd as the cgroup driver

sudo mkdir -p /etc/docker



```
ubuntu@ip-172-31-88-198:~$ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-88-198:~$
```

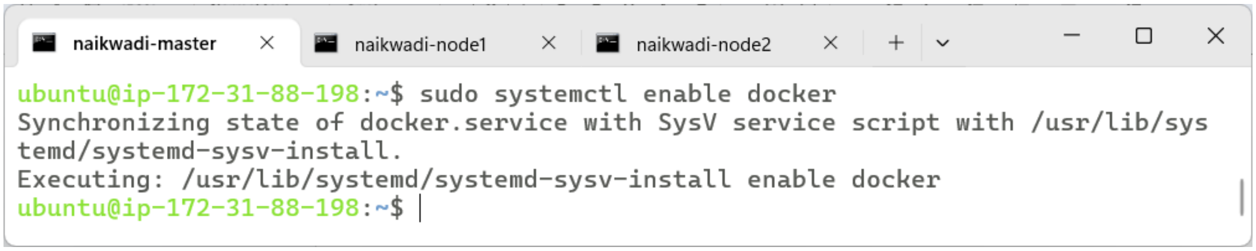
```
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
```



```
ubuntu@ip-172-31-90-4:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
> {
  "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-90-4:~$ 
```

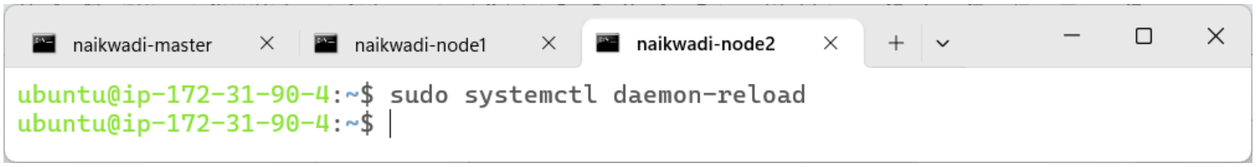
# Enable and start Docker

sudo systemctl enable docker



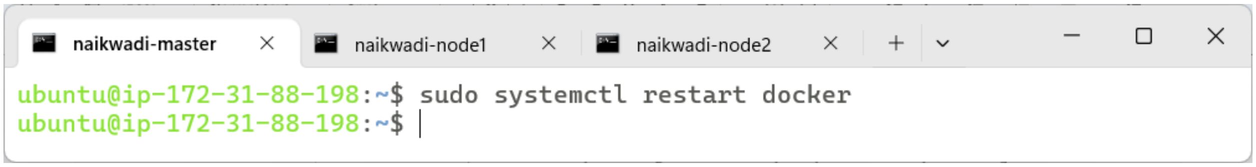
```
ubuntu@ip-172-31-88-198:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/sys
temd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-88-198:~$ 
```

sudo systemctl daemon-reload



```
ubuntu@ip-172-31-90-4:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-90-4:~$ 
```

sudo systemctl restart docker

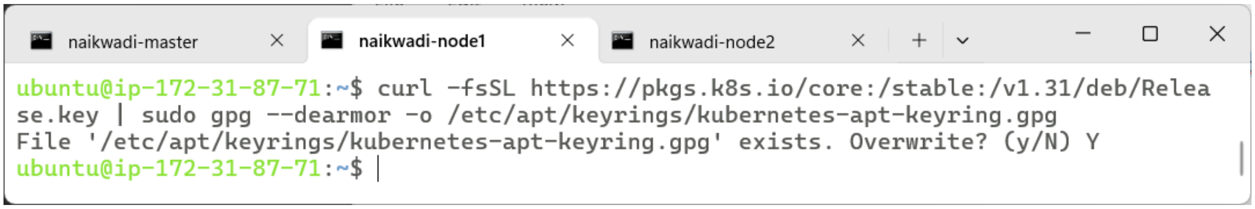


```
ubuntu@ip-172-31-88-198:~$ sudo systemctl restart docker
ubuntu@ip-172-31-88-198:~$ 
```

<u>Install Kubernetes on All Instances</u>

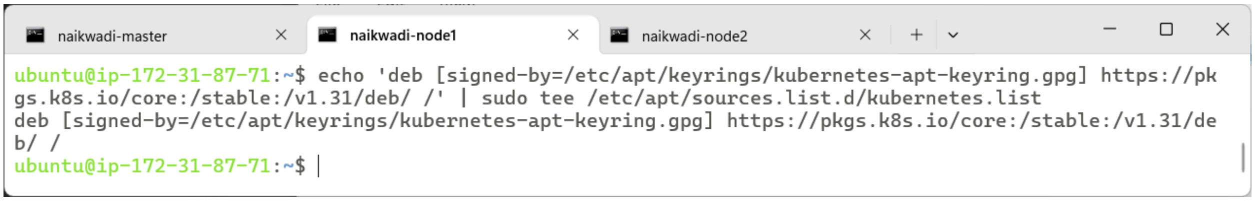
Run the below command to install Kubernets.

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg



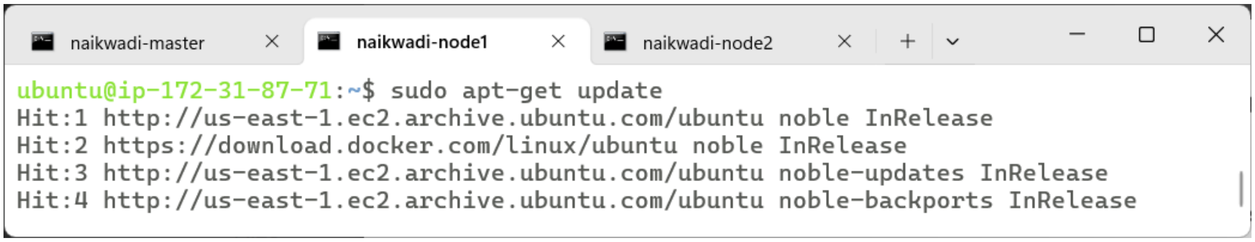
```
ubuntu@ip-172-31-87-71:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Relea
se.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
File '/etc/apt/keyrings/kubernetes-apt-keyring.gpg' exists. Overwrite? (y/N) Y
ubuntu@ip-172-31-87-71:~$ 
```

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



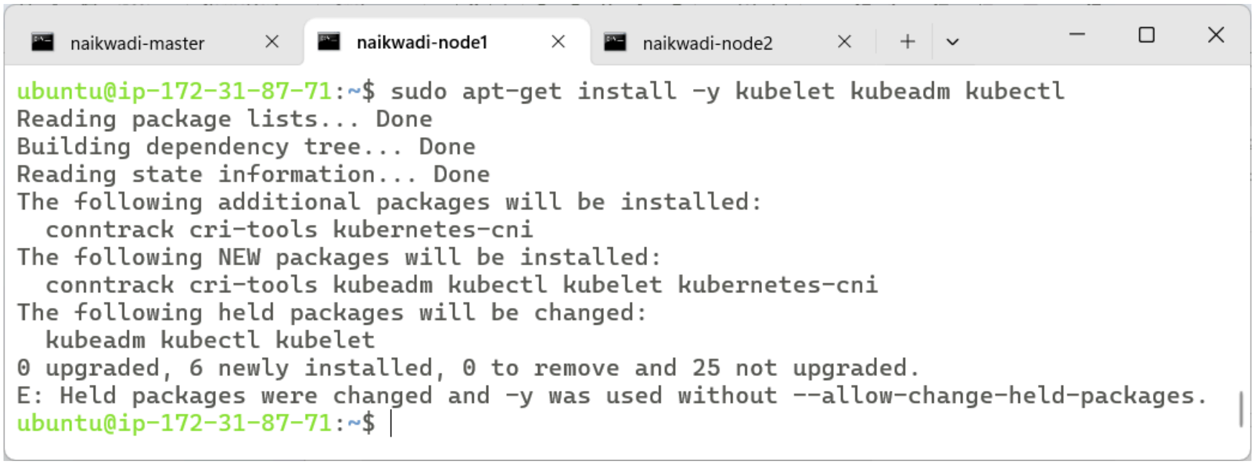
```
ubuntu@ip-172-31-87-71:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pk
gs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/de
b/ /
ubuntu@ip-172-31-87-71:~$ 
```

sudo apt-get update



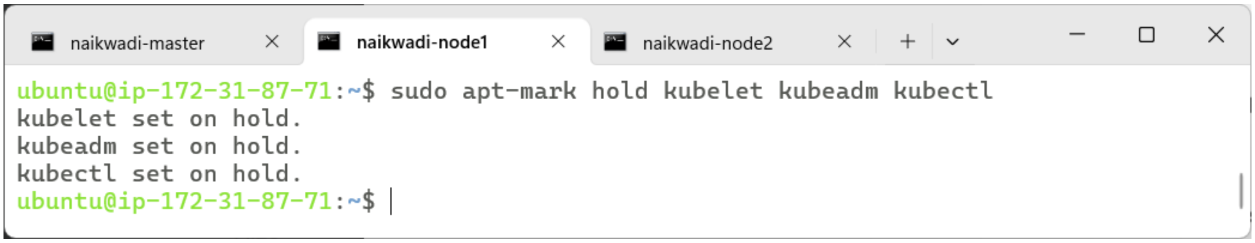
```
ubuntu@ip-172-31-87-71:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
```

sudo apt-get install -y kubelet kubeadm kubectl



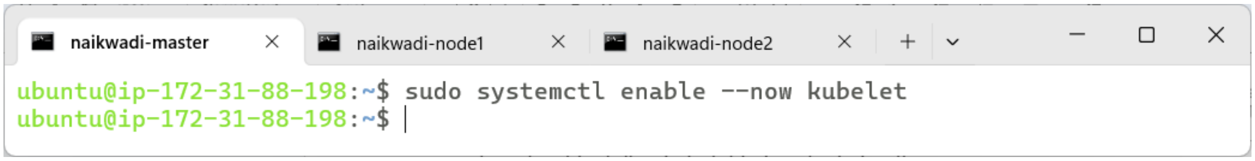
```
ubuntu@ip-172-31-87-71:~$ sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
The following held packages will be changed:
  kubeadm kubectl kubelet
0 upgraded, 6 newly installed, 0 to remove and 25 not upgraded.
E: Held packages were changed and -y was used without --allow-change-held-packages.
ubuntu@ip-172-31-87-71:~$
```

sudo apt-mark hold kubelet kubeadm kubectl



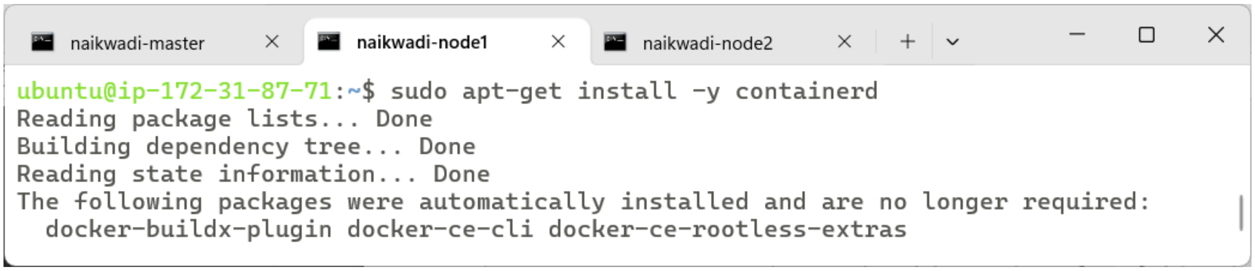
```
ubuntu@ip-172-31-87-71:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-87-71:~$
```

sudo systemctl enable --now kubelet



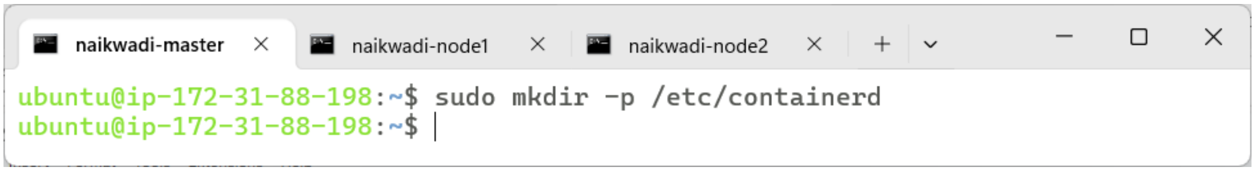
```
ubuntu@ip-172-31-88-198:~$ sudo systemctl enable --now kubelet
ubuntu@ip-172-31-88-198:~$
```

sudo apt-get install -y containerd



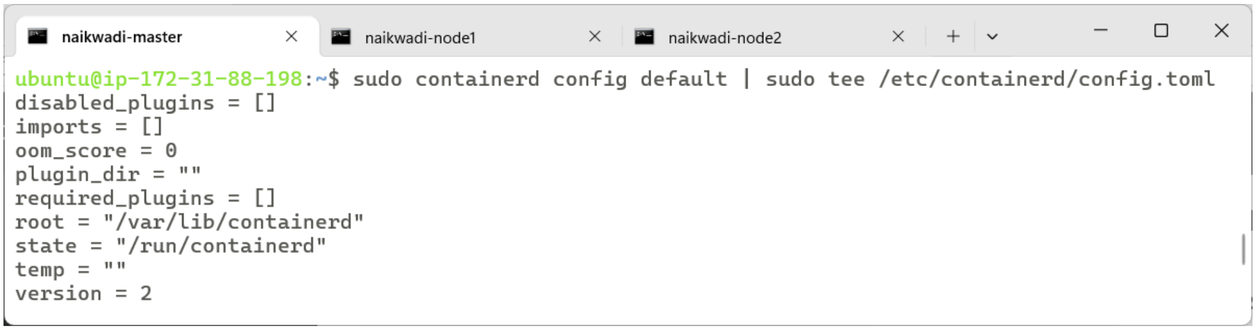
```
ubuntu@ip-172-31-87-71:~$ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras
```

sudo mkdir -p /etc/containerd



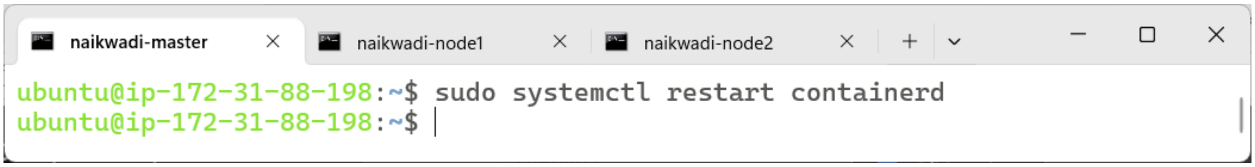
```
ubuntu@ip-172-31-88-198:~$ sudo mkdir -p /etc/containerd
ubuntu@ip-172-31-88-198:~$
```

sudo containerd config default | sudo tee /etc/containerd/config.toml



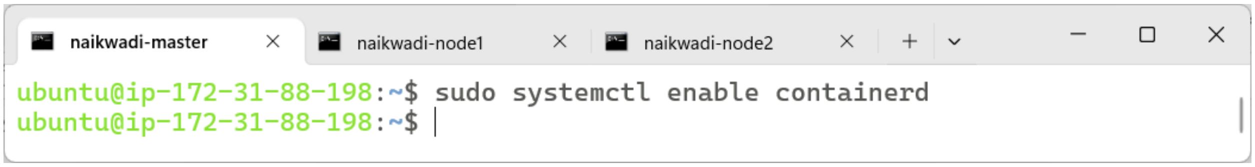
```
ubuntu@ip-172-31-88-198:~$ sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2
```

sudo systemctl restart containerd



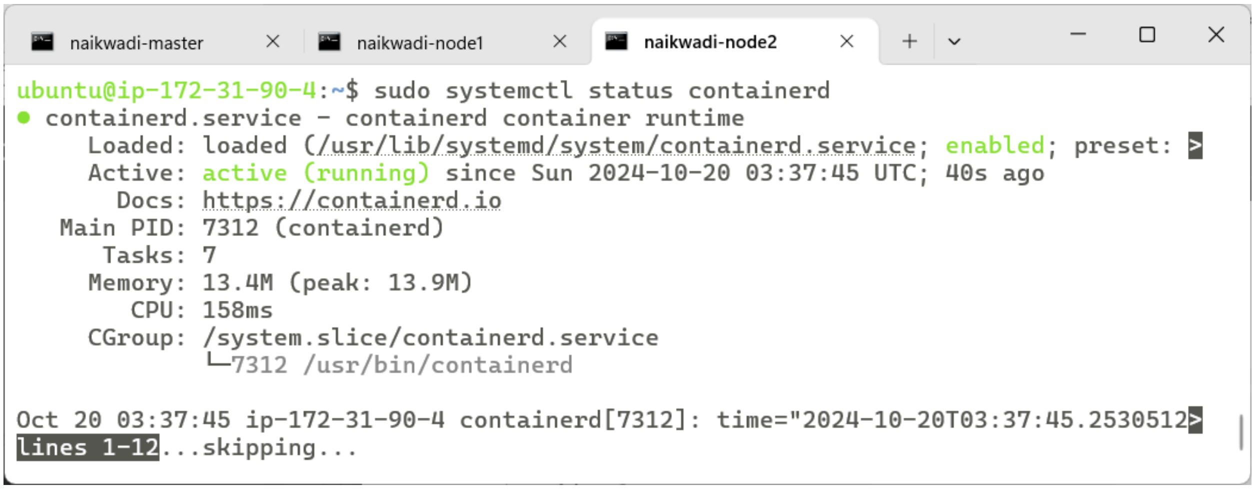
```
ubuntu@ip-172-31-88-198:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-88-198:~$
```

sudo systemctl enable containerd



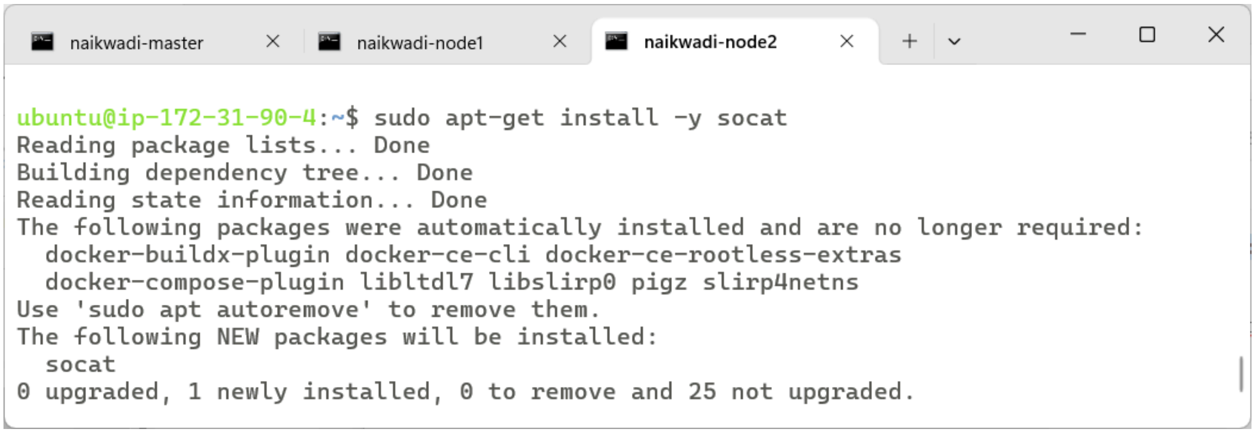
```
ubuntu@ip-172-31-88-198:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-88-198:~$
```

sudo systemctl status containerd



```
ubuntu@ip-172-31-90-4:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: >
     Active: active (running) since Sun 2024-10-20 03:37:45 UTC; 40s ago
       Docs: https://containerd.io
   Main PID: 7312 (containerd)
      Tasks: 7
     Memory: 13.4M (peak: 13.9M)
        CPU: 158ms
     CGroup: /system.slice/containerd.service
             └─7312 /usr/bin/containerd

Oct 20 03:37:45 ip-172-31-90-4 containerd[7312]: time="2024-10-20T03:37:45.2530512>
lines 1-12...skipping...
```

sudo apt-get install -y socat

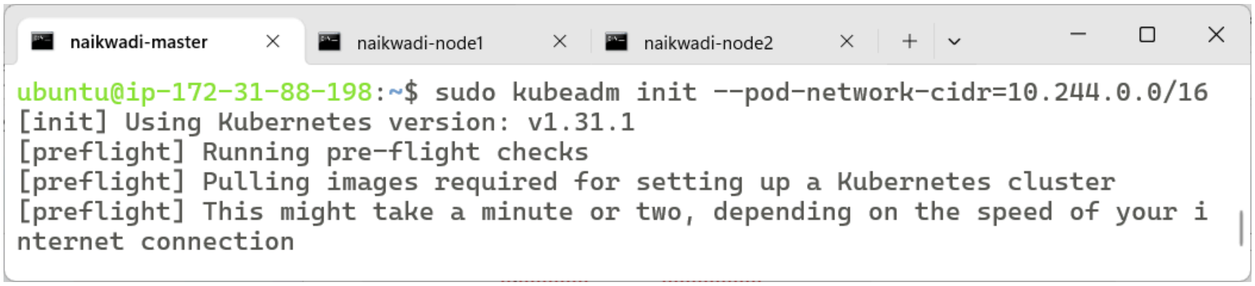


```
ubuntu@ip-172-31-90-4:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 25 not upgraded.
```

Initialize the Kubernetes Master Node

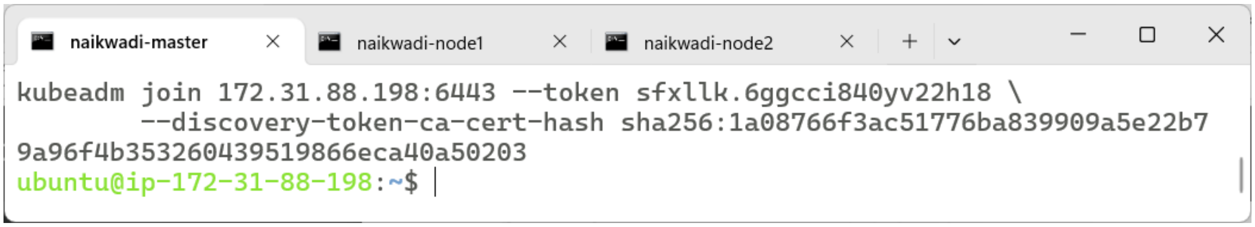
Initialize the Kubecluster .Now Perform this Command only for Master.

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



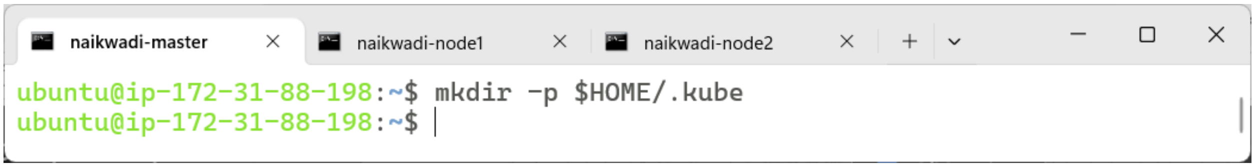
```
ubuntu@ip-172-31-88-198:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.1
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your i
nternet connection
```

Once the initialization is complete, you will see a command that looks like this:
(**Copy and save** this command for later; you will need it to join the worker nodes to the master.)



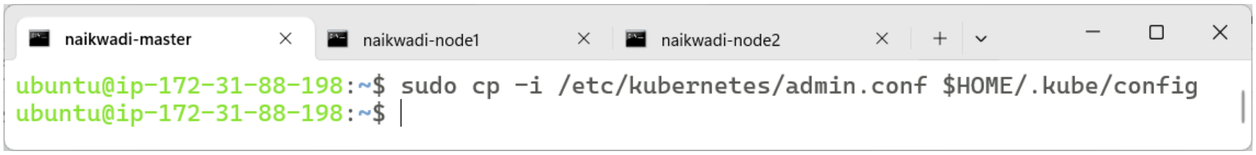
```
kubeadm join 172.31.88.198:6443 --token sfxllk.6ggci840yv22h18 \
        --discovery-token-ca-cert-hash sha256:1a08766f3ac51776ba839909a5e22b7
9a96f4b353260439519866eca40a50203
ubuntu@ip-172-31-88-198:~$
```

Next, configure kubectl to interact with the cluster:
mkdir -p $HOME/.kube



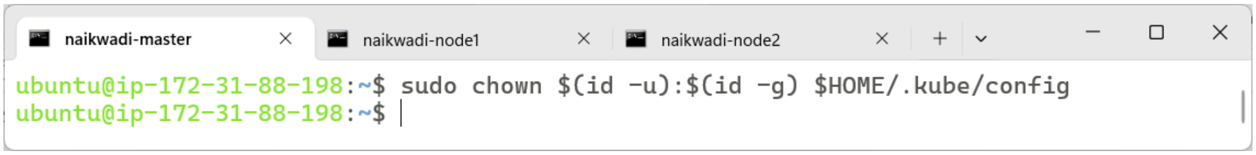
```
ubuntu@ip-172-31-88-198:~$ mkdir -p $HOME/.kube
ubuntu@ip-172-31-88-198:~$
```

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



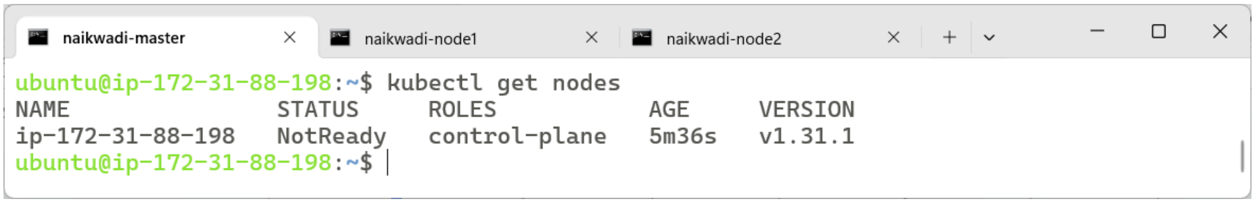
```
ubuntu@ip-172-31-88-198:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
ubuntu@ip-172-31-88-198:~$
```

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



```
ubuntu@ip-172-31-88-198:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-88-198:~$
```

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



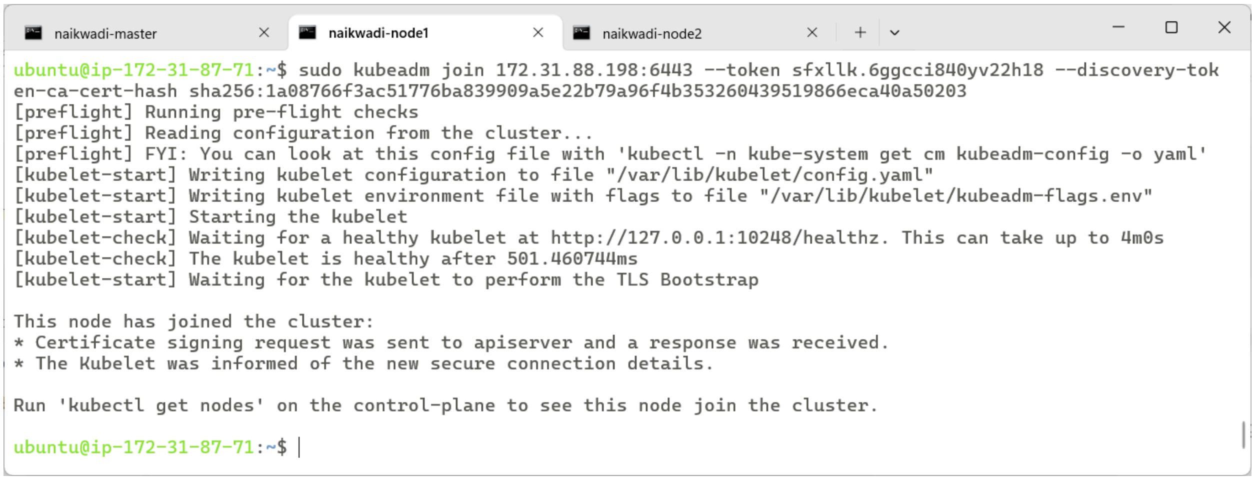
```
ubuntu@ip-172-31-88-198:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-88-198   NotReady   control-plane   5m36s   v1.31.1
ubuntu@ip-172-31-88-198:~$
```

Join Worker Nodes to the Cluster

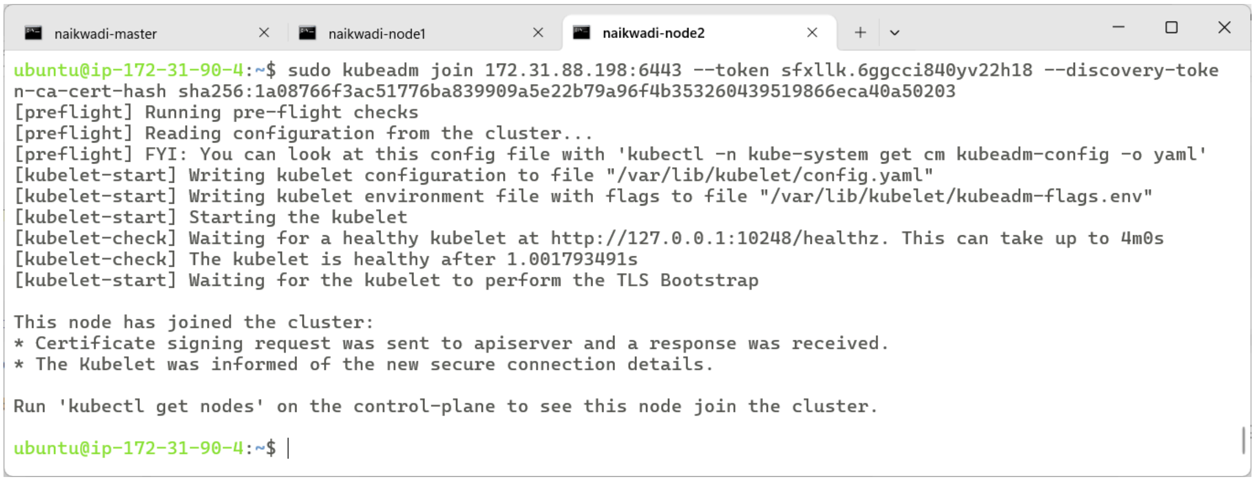
On **Node 1** and **Node 2**, run the `kubeadm join` command you saved from the master node initialization. It should look like this:

sudo kubeadm join <Master-IP>:6443 --token <token> --discovery-token-ca-cert-hash <hash>

(In my case, command is as follows:
sudo kubeadm join 172.31.88.198:6443 --token sfxllk.6ggci840yv22h18 \
--discovery-token-ca-cert-hash
sha256:1a08766f3ac51776ba839909a5e22b79a96f4b353260439519866eca40a50203)



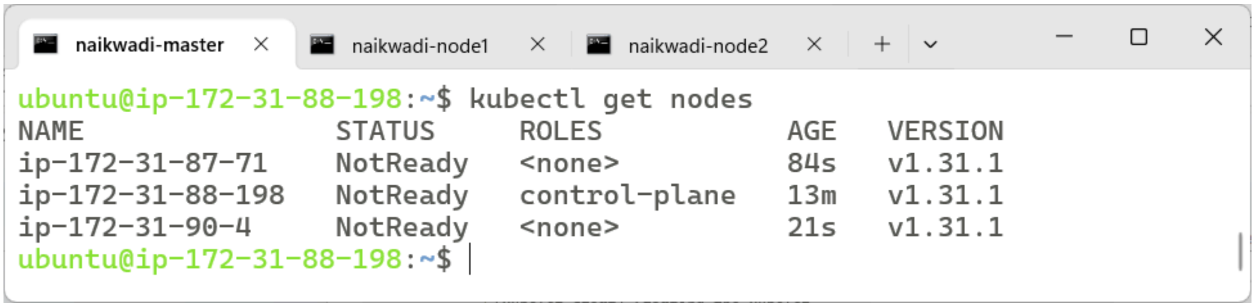
```
ubuntu@ip-172-31-87-71:~$ sudo kubeadm join 172.31.88.198:6443 --token sfxllk.6ggcci840yv22h18 --discovery-tok
en-ca-cert-hash sha256:1a08766f3ac51776ba839909a5e22b79a96f4b353260439519866eca40a50203
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] 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-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 501.460744ms
[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-172-31-87-71:~$ |
```



```
ubuntu@ip-172-31-90-4:~$ sudo kubeadm join 172.31.88.198:6443 --token sfxllk.6ggcci840yv22h18 --discovery-toke
n-ca-cert-hash sha256:1a08766f3ac51776ba839909a5e22b79a96f4b353260439519866eca40a50203
[preflight] Running pre-flight checks
[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-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.001793491s
[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-172-31-90-4:~$ |
```

Verify the Cluster

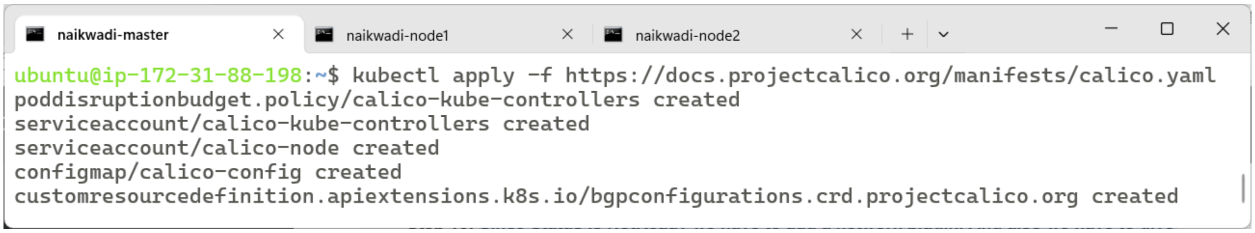
On the master node, verify that the nodes have joined the cluster:
kubectl get nodes



```
ubuntu@ip-172-31-88-198:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE   VERSION
ip-172-31-87-71    NotReady   <none>          84s   v1.31.1
ip-172-31-88-198   NotReady   control-plane   13m   v1.31.1
ip-172-31-90-4     NotReady   <none>          21s   v1.31.1
ubuntu@ip-172-31-88-198:~$ |
```

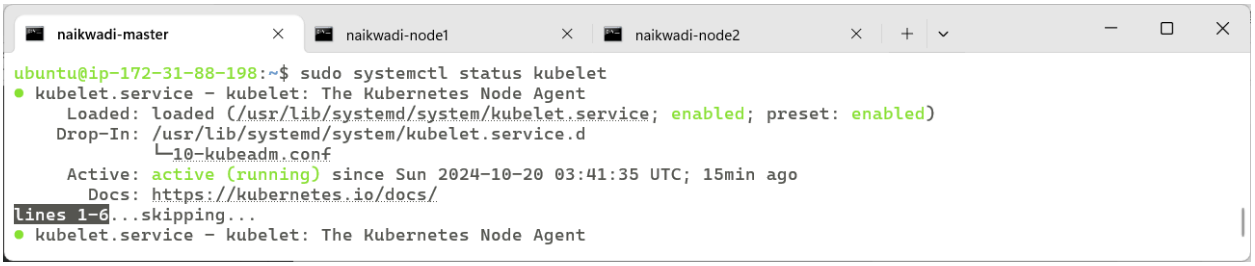
Install a Network Plugin (Calico)

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



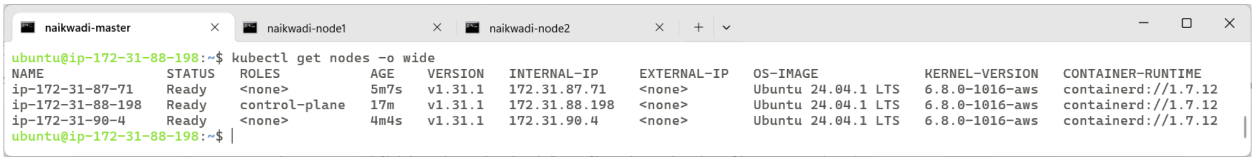
```
ubuntu@ip-172-31-88-198:~$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
```

sudo systemctl status kubelet



```
ubuntu@ip-172-31-88-198:~$ 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 Sun 2024-10-20 03:41:35 UTC; 15min ago
       Docs: https://kubernetes.io/docs/
lines 1-6...skipping...
● kubelet.service - kubelet: The Kubernetes Node Agent
```

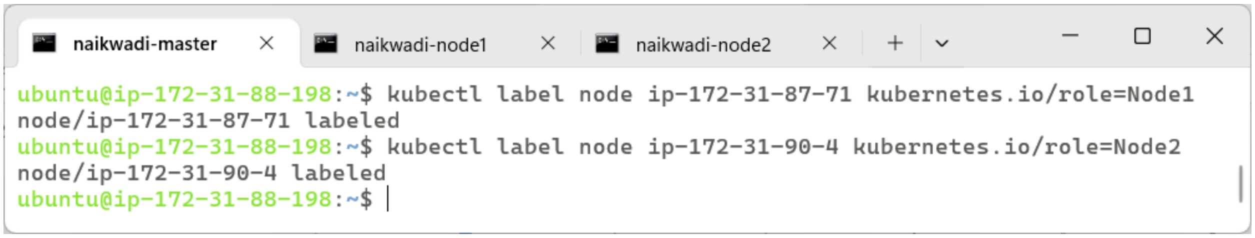
Now Run command (we can see Status is ready).
kubectl get nodes -o wide



```
ubuntu@ip-172-31-88-198:~$ kubectl get nodes -o wide
NAME               STATUS   ROLES           AGE    VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-87-71    Ready    <none>          5m7s   v1.31.1   172.31.87.71    <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ip-172-31-88-198   Ready    control-plane   17m    v1.31.1   172.31.88.198   <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ip-172-31-90-4     Ready    <none>          4m4s   v1.31.1   172.31.90.4     <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ubuntu@ip-172-31-88-198:~$ |
```

Rename/Label the Nodes

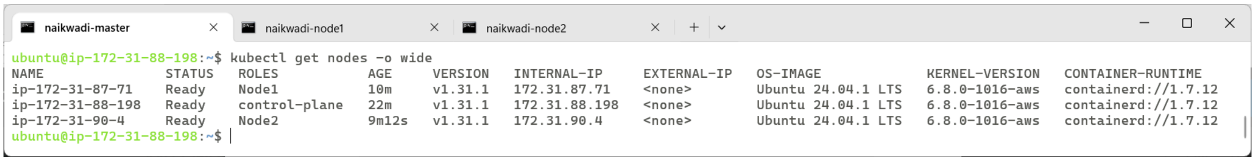
kubectl label node ip-172-31-87-71 kubernetes.io/role=Node1
kubectl label node ip-172-31-90-4 kubernetes.io/role=Node2



```
ubuntu@ip-172-31-88-198:~$ kubectl label node ip-172-31-87-71 kubernetes.io/role=Node1
node/ip-172-31-87-71 labeled
ubuntu@ip-172-31-88-198:~$ kubectl label node ip-172-31-90-4 kubernetes.io/role=Node2
node/ip-172-31-90-4 labeled
ubuntu@ip-172-31-88-198:~$ |
```

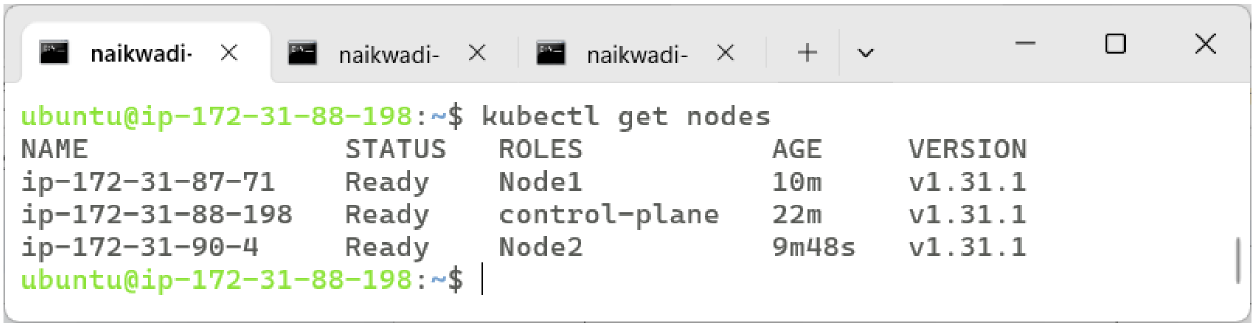
Final Check

Run (again to confirm all nodes are in the Ready state and properly labeled).
kubectl get nodes



```
ubuntu@ip-172-31-88-198:~$ kubectl get nodes -o wide
NAME               STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-87-71    Ready    Node1           10m     v1.31.1   172.31.87.71    <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ip-172-31-88-198   Ready    control-plane   22m     v1.31.1   172.31.88.198   <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ip-172-31-90-4     Ready    Node2           9m12s   v1.31.1   172.31.90.4     <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ubuntu@ip-172-31-88-198:~$ |
```

Or run
kubectl get nodes



```
ubuntu@ip-172-31-88-198:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-87-71    Ready    Node1           10m     v1.31.1
ip-172-31-88-198   Ready    control-plane   22m     v1.31.1
ip-172-31-90-4     Ready    Node2           9m48s   v1.31.1
ubuntu@ip-172-31-88-198:~$ |
```

Conclusion :

In this experiment, we successfully set up a Kubernetes cluster on AWS EC2 instances, understanding how Kubernetes manages and orchestrates containerized applications across multiple nodes. This hands-on experience demonstrated Kubernetes' ability to automate deployment, scaling, and resource management in a cloud environment.