Advanced DevOps Lab

Experiment 3

Name: Yash Rahate
Class: D15B
Roll No.: 48

Aim:

To understand the Kubernetes Cluster Architecture, install and Spin Up a 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.

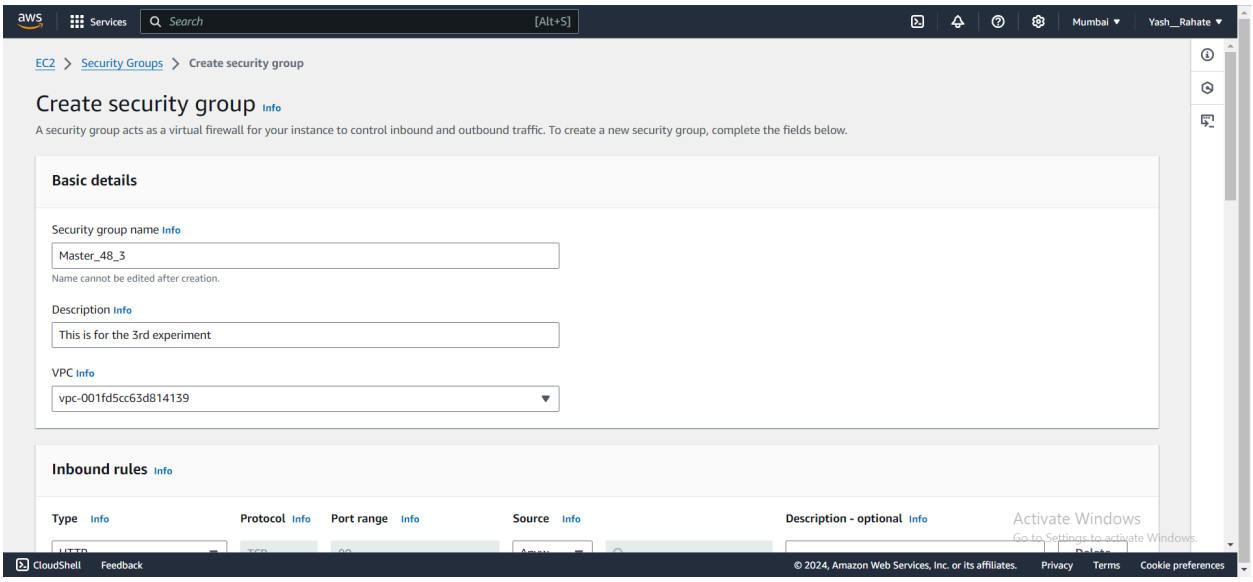
This ensured that the Kubernetes cluster had the necessary resources to function smoothly

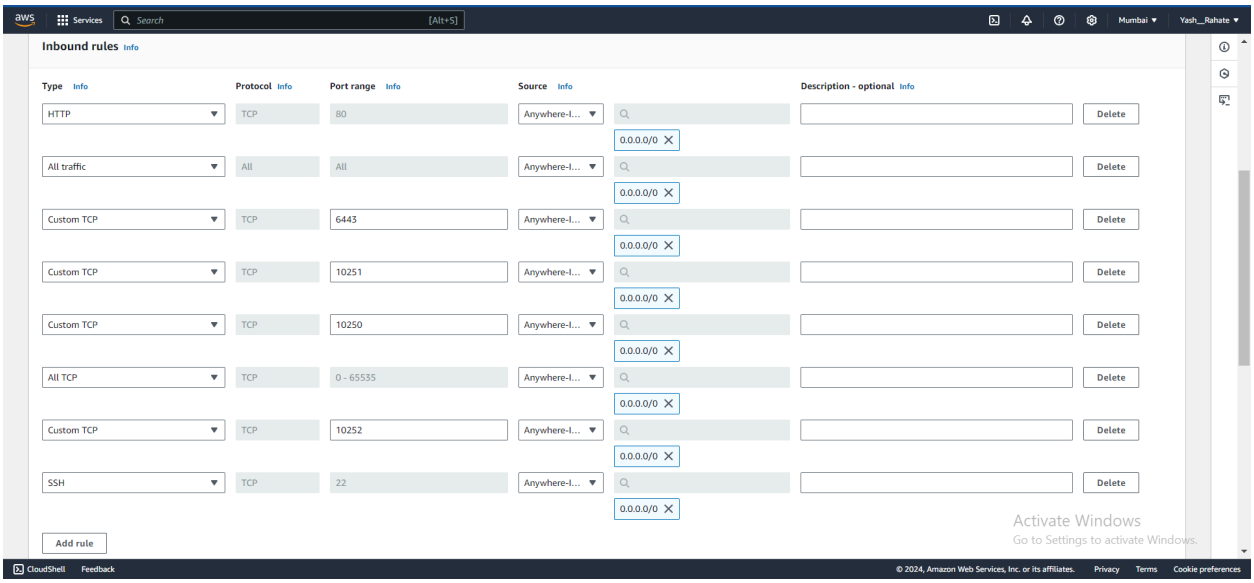
Note:

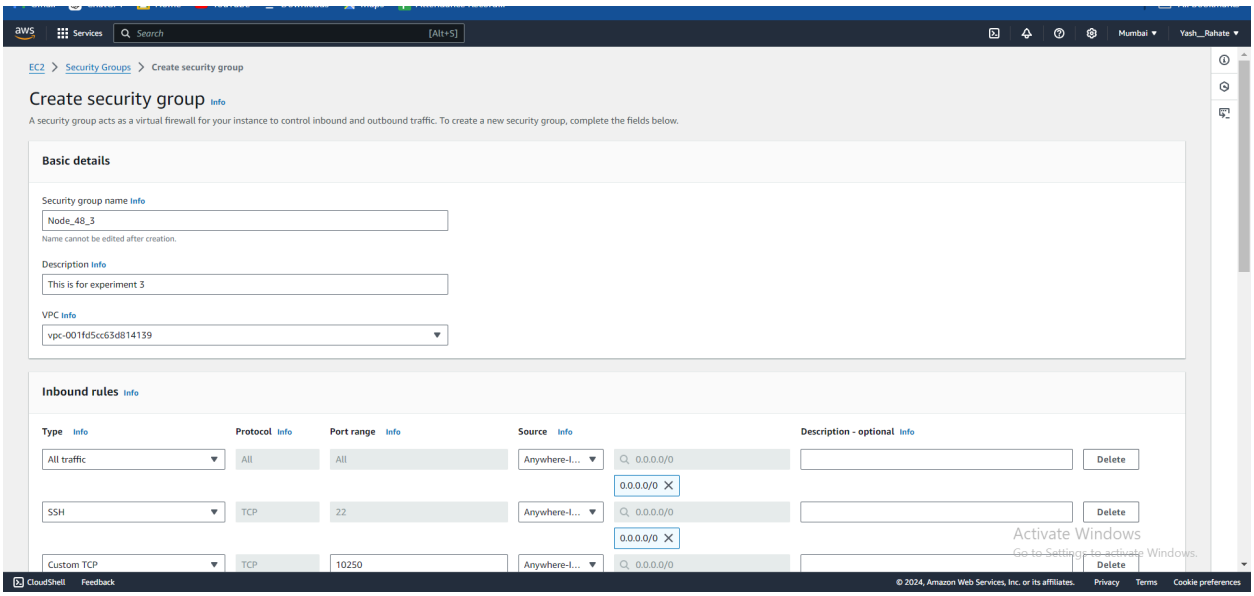
AWS Personal Account is preferred but we can also perform it on AWS Academy(adding some ignores in the command if any error occurs in below as the below experiment is performed on Personal Account .).If You are using AWS Academy Account Errors you will face in kubeadm init command so you have to add some ignores with this command.

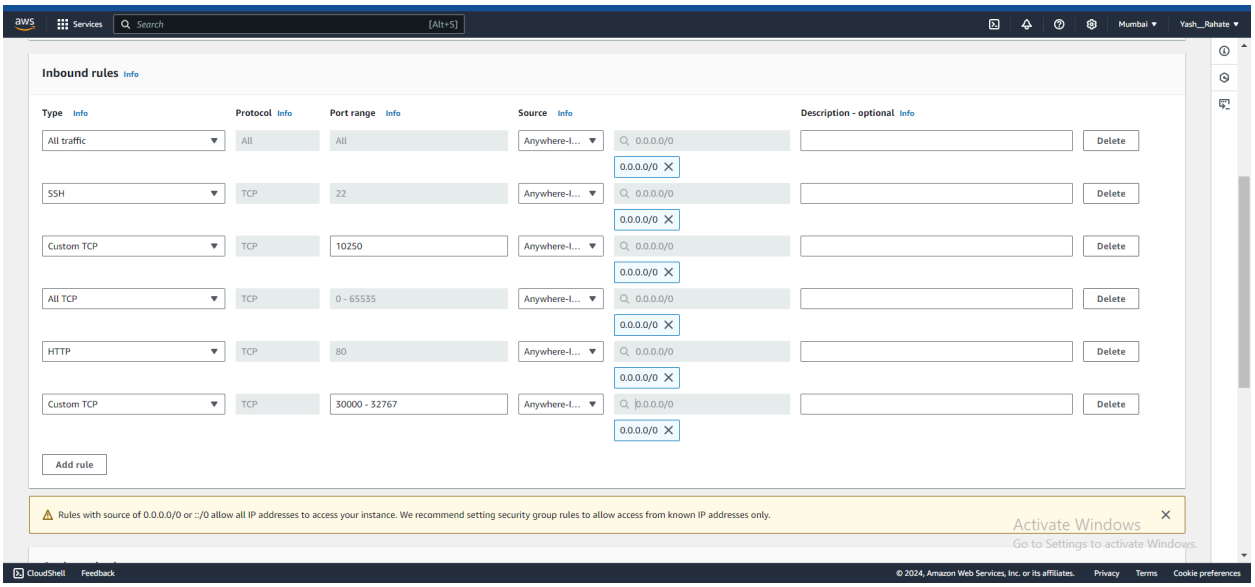
Prerequisites :

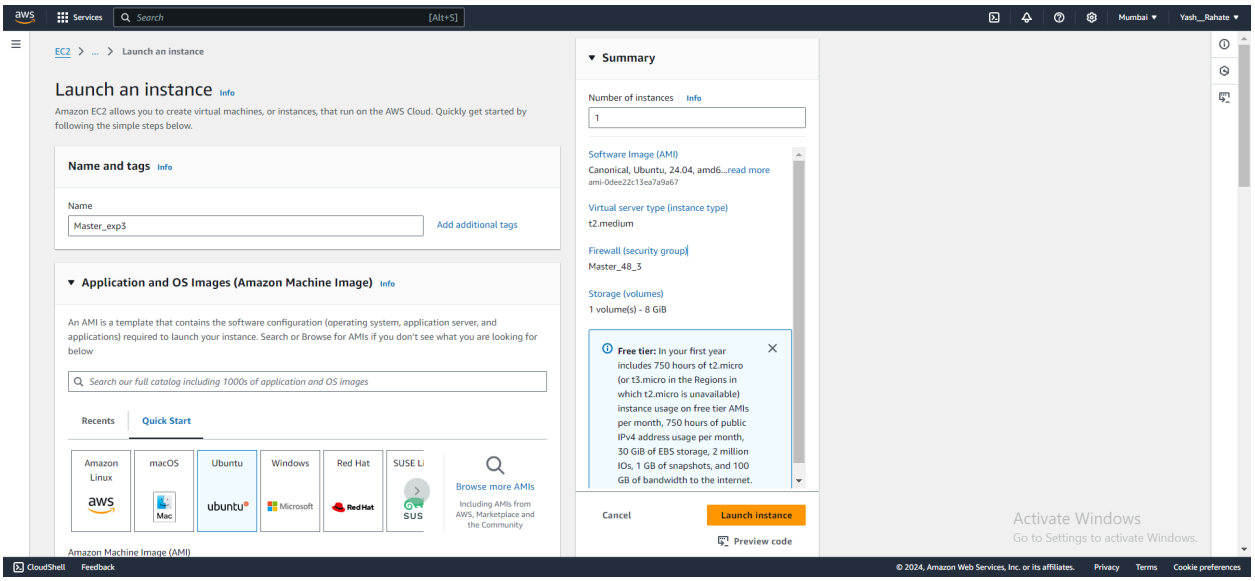
Create 2 Security Groups for Master and Nodes and add the following rules inbound rules in those Groups.

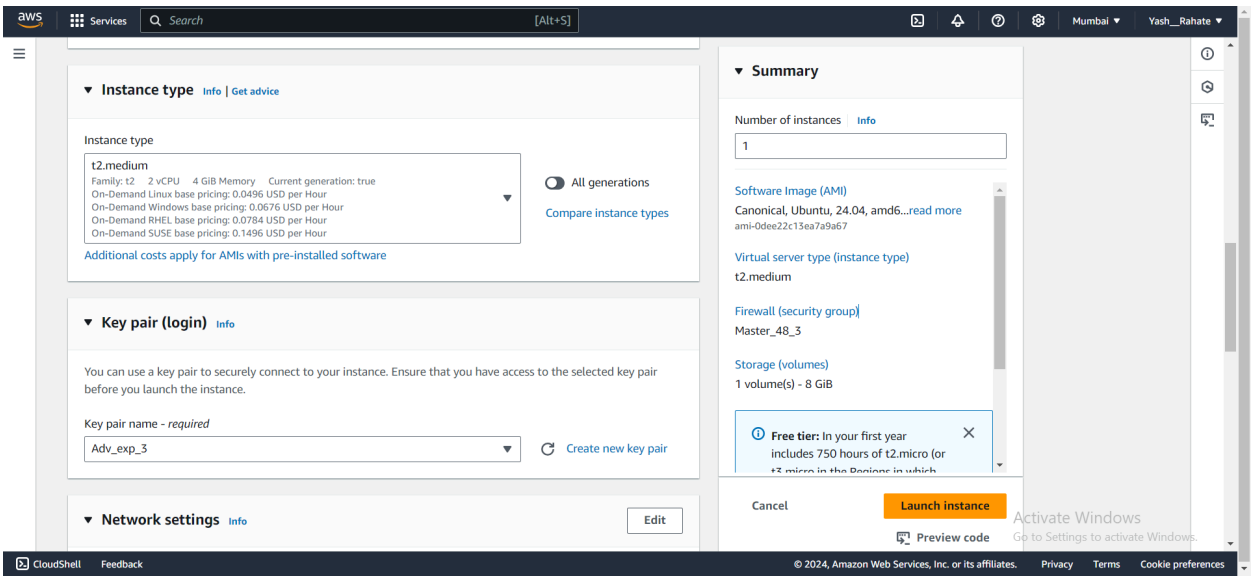

Services
Search
[Alt+S]
Mumbai
Yash_Rahate
EC2 > Security Groups > Create security group
Create security group Info
A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.
Basic details
Security group name Info
Master_48_3
Name cannot be edited after creation.
Description Info
This is for the 3rd experiment
VPC Info
vpc-001fd5cc63d814139
Inbound rules Info
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

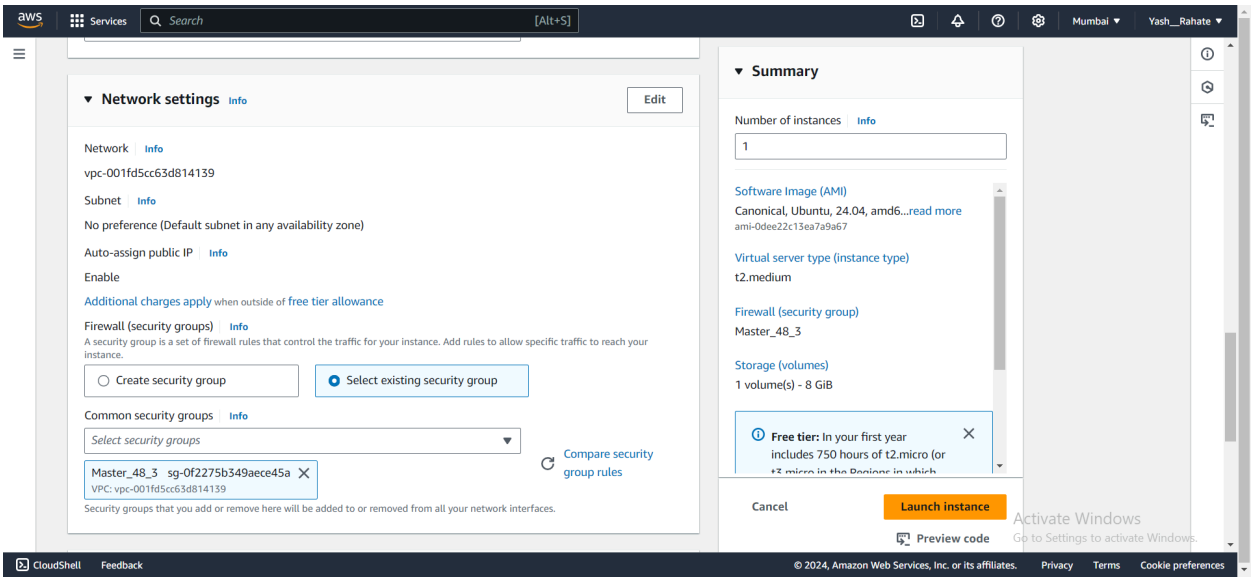

Inbound rules Info
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
HTTP
TCP
80
Anywhere-I...
0.0.0.0/0
Delete
All traffic
All
All
Custom TCP
6443
10251
10250
All TCP
0 - 65535
10252
SSH
22
Add rule
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

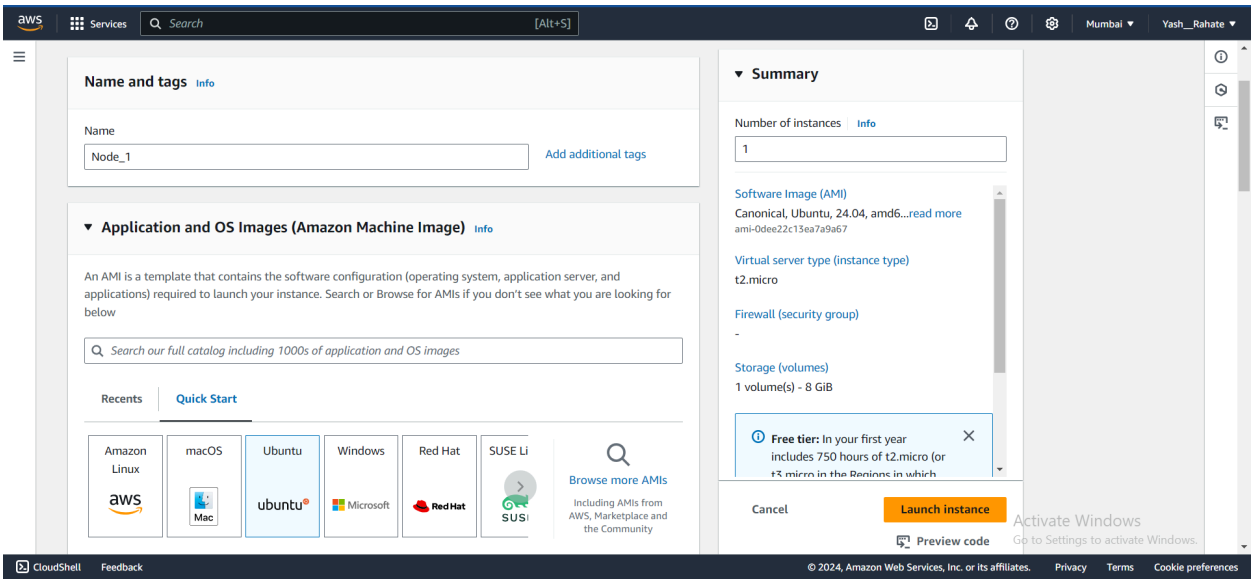

Services
Search
[Alt+S]
Mumbai
Yash_Rahate
EC2 > Security Groups > Create security group
Create security group Info
A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.
Basic details
Security group name Info
Node_48_3
Name cannot be edited after creation.
Description Info
This is for experiment 3
VPC Info
vpc-001fd5cc63d814139
Inbound rules Info
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
All traffic
All
All
Anywhere-I...
0.0.0.0/0
Delete
SSH
TCP
22
Custom TCP
10250
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

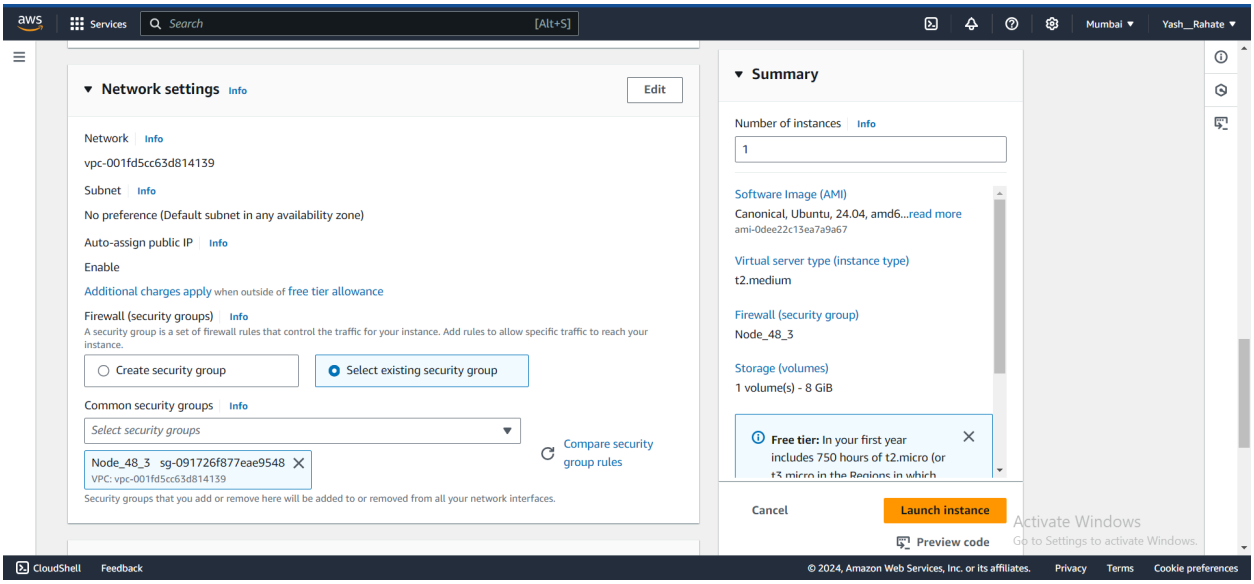

Inbound rules Info
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
All traffic
SSH
Custom TCP
10250
All TCP
0 - 65535
HTTP
80
Custom TCP
30000 - 32767
Anywhere-I...
0.0.0.0/0
Delete
Add rule
Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

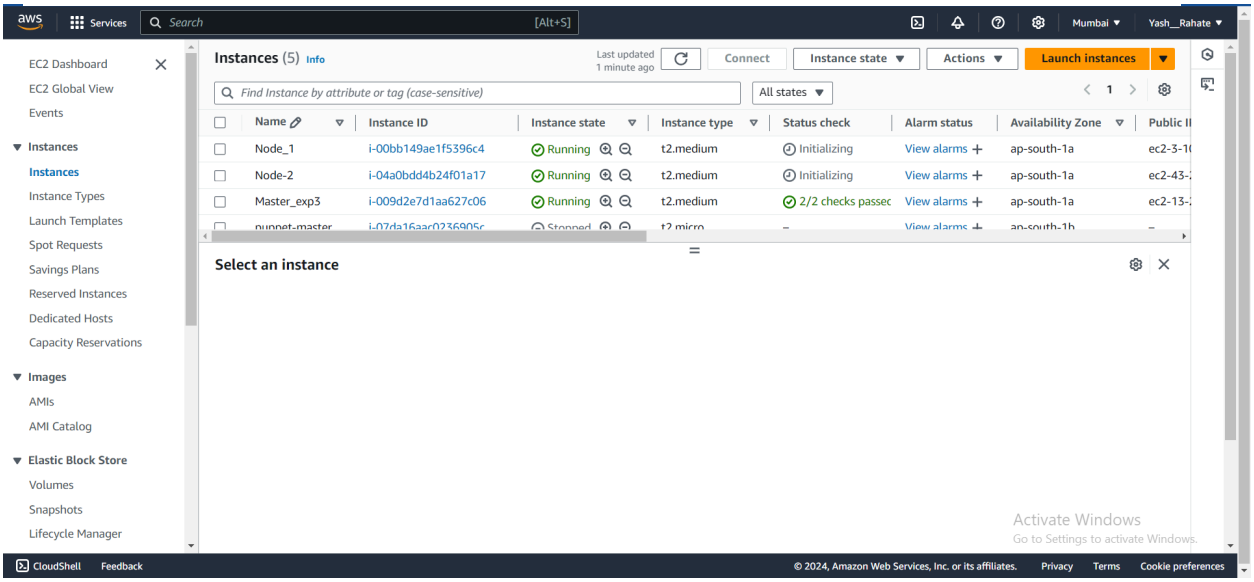

EC2 > ... > Launch an instance
Launch an instance Info
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.
Name and tags Info
Name
Master_exp3
Add additional tags
Application and OS Images (Amazon Machine 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)
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0dee22c13ea7a9a67
Virtual server type (instance type)
t2.medium
Firewall (security group)
Master_48_3
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.
Cancel
Launch instance
Preview code
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

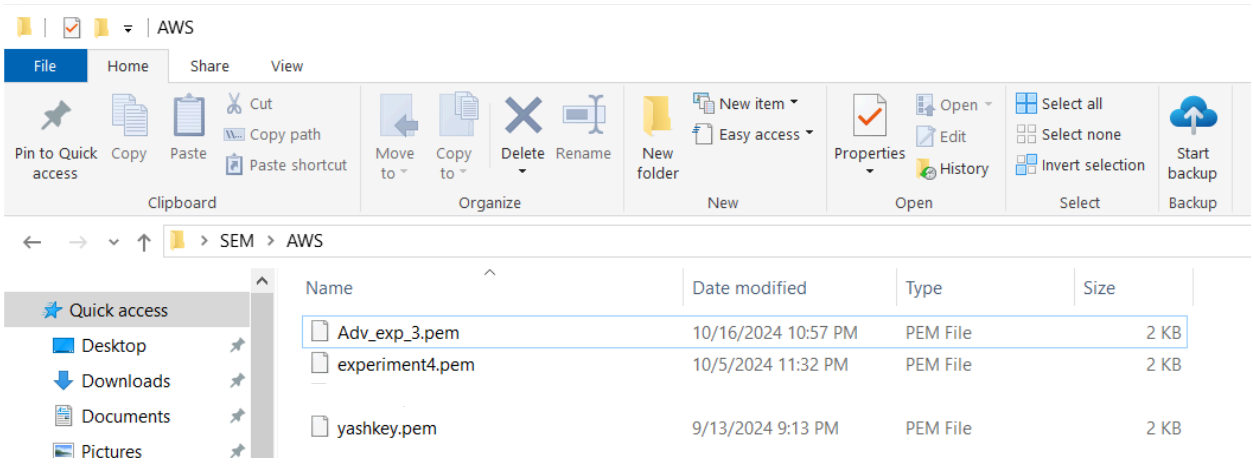

Instance type Info | Get advice
Instance type
t2.medium
Family: t2 2 vCPU 4 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0496 USD per Hour
On-Demand Windows base pricing: 0.0676 USD per Hour
On-Demand RHEL base pricing: 0.0784 USD per Hour
On-Demand SUSE base pricing: 0.1496 USD 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
Adv_exp_3
Create new key pair
Network settings Info
Edit
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0dee22c13ea7a9a67
Virtual server type (instance type)
t2.medium
Firewall (security group)
Master_48_3
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or
Cancel
Launch instance
Preview code
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

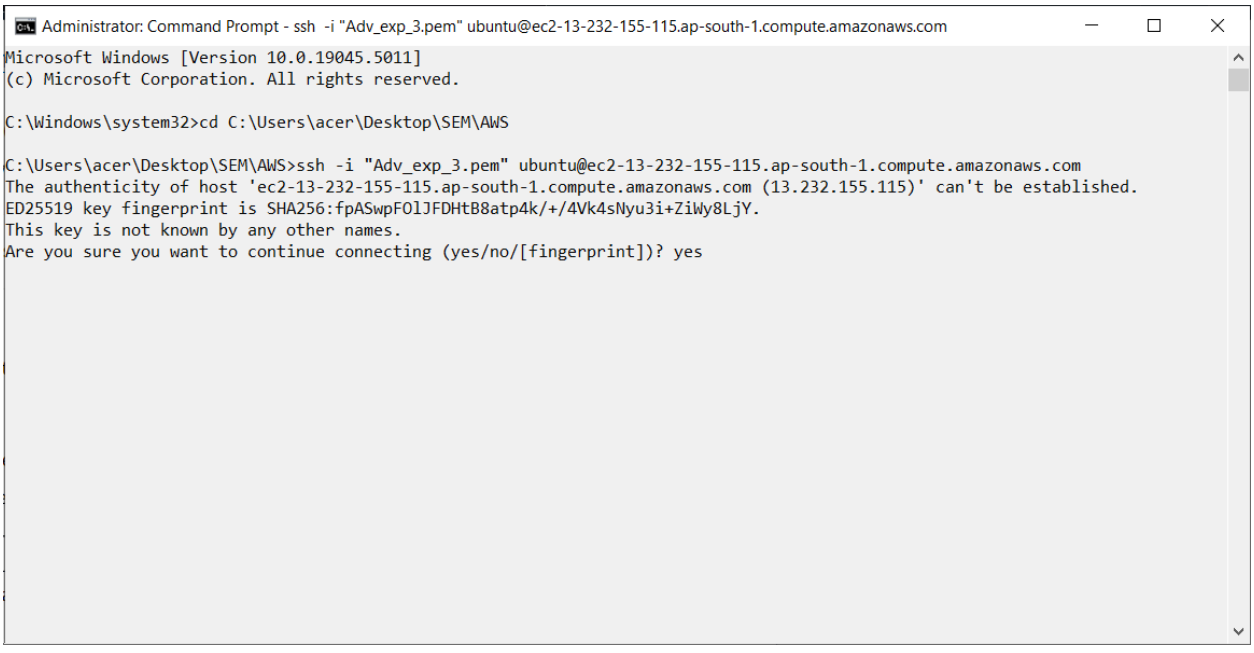

Services
Search
[Alt+S]
Mumbai
Yash_Rahate
Network settings Info
Edit
Network Info
vpc-001fd5cc63d814139
Subnet Info
No preference (Default subnet in any availability zone)
Auto-assign public IP Info
Enable
Additional charges apply when outside of free tier allowance
Firewall (security groups) Info
A 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_48_3 sg-0f2275b349aece45a
VPC: vpc-001fd5cc63d814139
Compare security group rules
Security groups that you add or remove here will be added to or removed from all your network interfaces.
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0dee22c13ea7a9a67
Virtual server type (instance type)
t2.medium
Firewall (security group)
Master_48_3
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or
Cancel
Launch instance
Preview code
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search
[Alt+S]
Mumbai
Yash_Rahate
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
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0dee22c13ea7a9a67
Virtual server type (instance type)
t2.micro
Firewall (security group)
-
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or
Cancel
Launch instance
Preview code
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

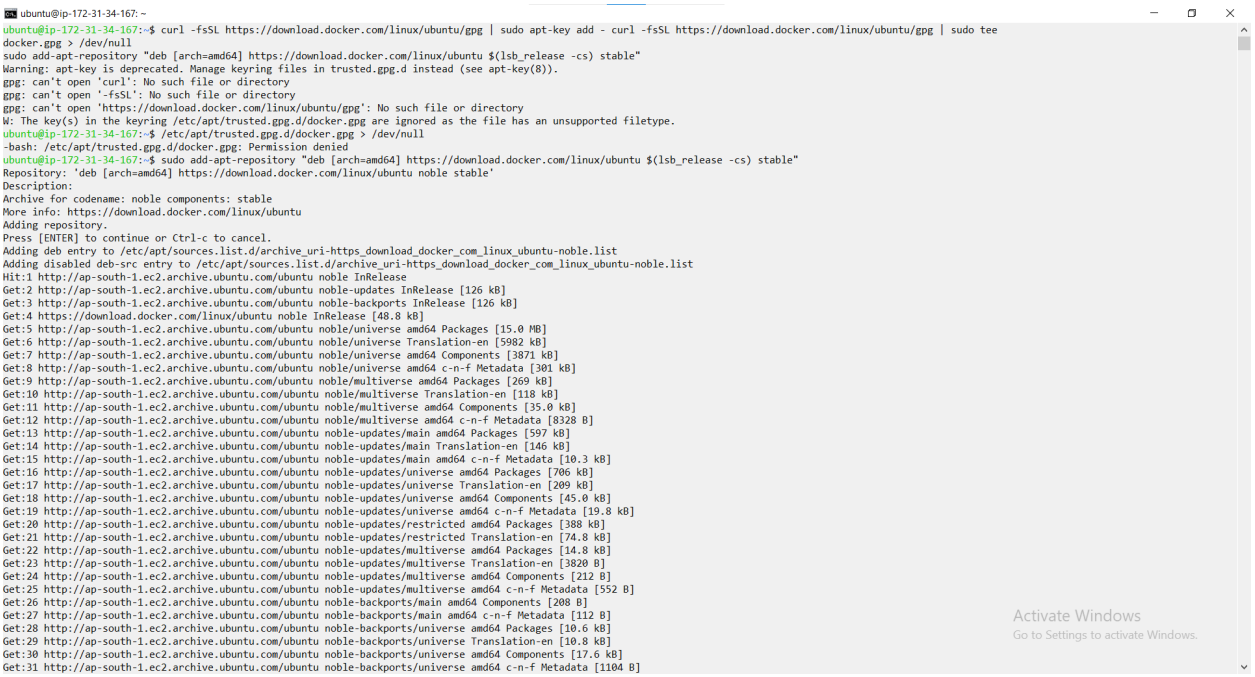

Services
Search
[Alt+S]
Mumbai
Yash_Rahate
Network settings Info
Edit
Network Info
vpc-001fd5cc63d814139
Subnet Info
No preference (Default subnet in any availability zone)
Auto-assign public IP Info
Enable
Additional charges apply when outside of free tier allowance
Firewall (security groups) Info
A 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
Node_48_3 sg-091726f877eae9548
VPC: vpc-001fd5cc63d814139
Compare security group rules
Security groups that you add or remove here will be added to or removed from all your network interfaces.
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0dee22c13ea7a9a67
Virtual server type (instance type)
t2.medium
Firewall (security group)
Node_48_3
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or
Cancel
Launch instance
Preview code
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

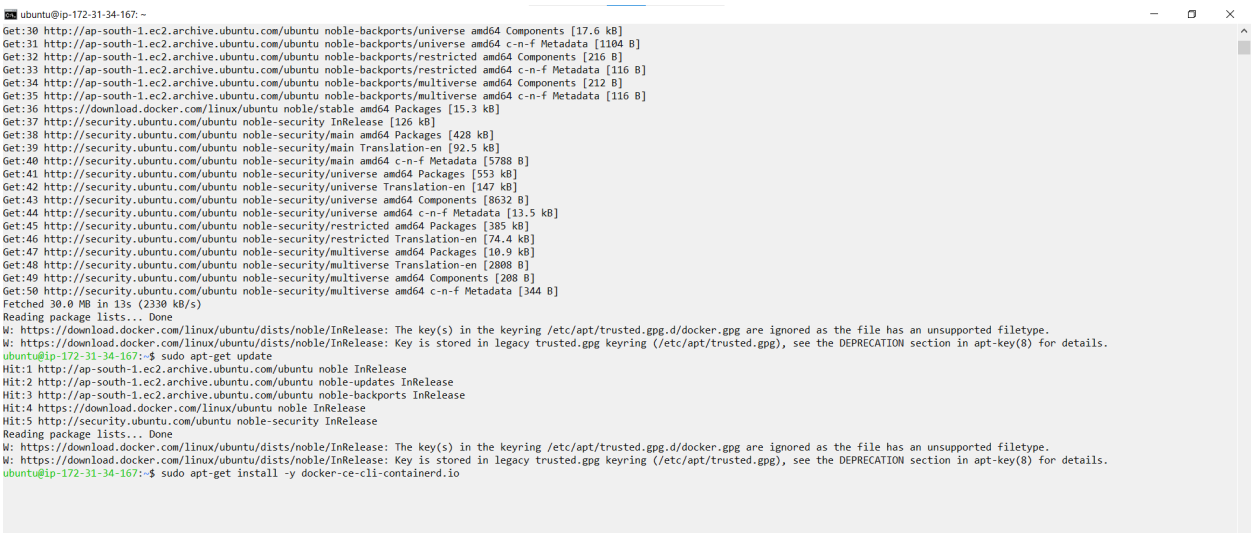

Services
Search
[Alt+S]
Mumbai
Yash_Rahate
EC2 Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Instances (5) Info
Last updated 1 minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Node_1 i-00bb149ae1f5396c4 Running t2.medium Initializing View alarms ap-south-1a
Node-2 i-04a0bdd4b24f01a17 Running t2.medium Initializing View alarms ap-south-1a
Master_exp3 i-009d2e7d1aa627c06 Running t2.medium 2/2 checks passed View alarms ap-south-1a
Select an instance
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

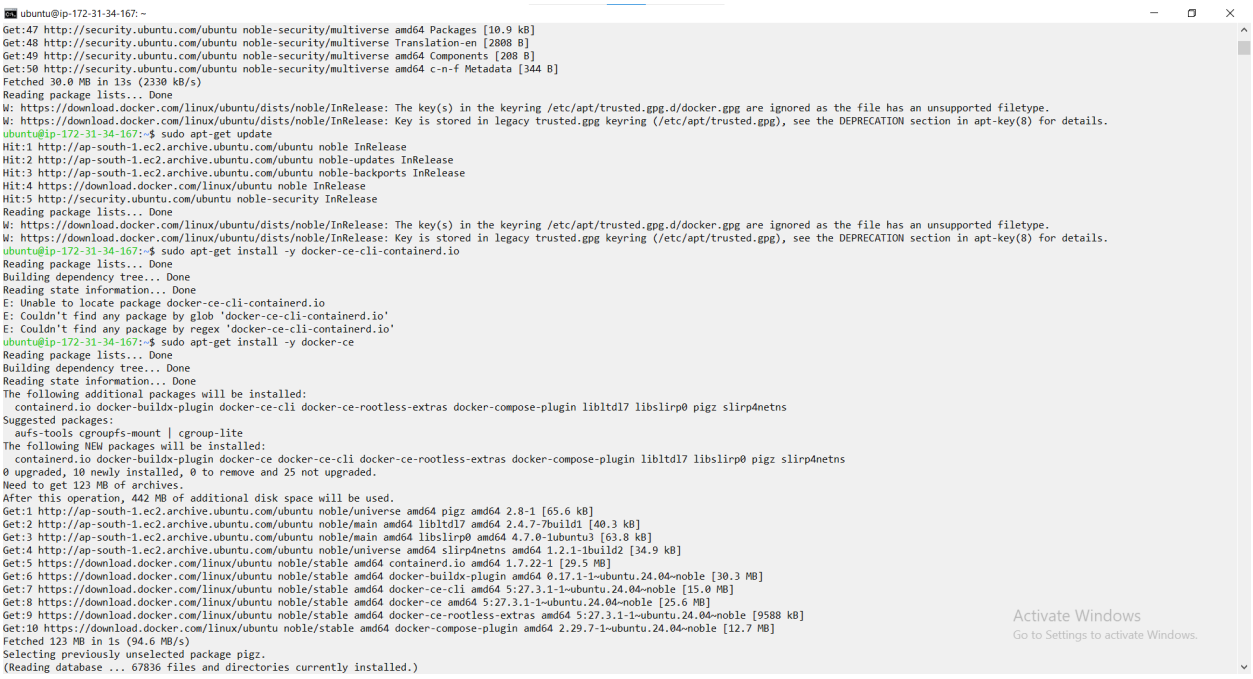

AWS
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New item
Easy access
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
Start backup
Backup
SEM > AWS
Quick access
Desktop
Downloads
Documents
Pictures
Name
Date modified
Type
Size
Adv_exp_3.pem 10/16/2024 10:57 PM PEM File 2 KB
experiment4.pem 10/5/2024 11:32 PM PEM File 2 KB
yashkey.pem 9/13/2024 9:13 PM PEM File 2 KB


Administrator: Command Prompt - ssh -i "Adv_exp_3.pem" ubuntu@ec2-13-232-155-115.ap-south-1.compute.amazonaws.com
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.
C:\Windows\system32>cd C:\Users\acer\Desktop\SEM\AWS
C:\Users\acer\Desktop\SEM\AWS>ssh -i "Adv_exp_3.pem" ubuntu@ec2-13-232-155-115.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-13-232-155-115.ap-south-1.compute.amazonaws.com (13.232.155.115)' can't be established.
ED25519 key fingerprint is SHA256:fpASwpFOlJFDHtB8atp4k/+/4Vk4sNyu3i+ZiWy8LjY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes


Administrator: Command Prompt - ssh -i "Adv_exp_3.pem" ubuntu@ec2-43-204-32-107.ap-south-1.compute.amazonaws.com
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.
C:\Windows\system32>cd C:\Users\acer\Desktop\SEM\AWS
C:\Users\acer\Desktop\SEM\AWS>ssh -i "Adv_exp_3.pem" ubuntu@ec2-43-204-32-107.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-43-204-32-107.ap-south-1.compute.amazonaws.com (43.204.32.107)' can't be established.
ED25519 key fingerprint is SHA256:vW6VFaRGXmffpuktvJU9vUzzIzapbXfE3CU48EKltec.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes


ubuntu@ip-172-31-34-167: ~
ubuntu@ip-172-31-34-167:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee
docker.gpg > /dev/null
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
gpg: can't open 'curl': No such file or directory
gpg: can't open '-fsSL': No such file or directory
gpg: can't open 'https://download.docker.com/linux/ubuntu/gpg': No such file or directory
W: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file has an unsupported filetype.
ubuntu@ip-172-31-34-167:~$ /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
-bash: /etc/apt/trusted.gpg.d/docker.gpg: Permission denied
ubuntu@ip-172-31-34-167:~$ 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.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:11 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:12 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:13 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [597 kB]
Get:14 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [146 kB]
Get:15 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [10.3 kB]
Get:16 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [706 kB]
Get:17 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [209 kB]
Get:18 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [45.0 kB]
Get:19 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [19.8 kB]
Get:20 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [388 kB]
Get:21 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [74.8 kB]
Get:22 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [14.8 kB]
Get:23 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [3820 B]
Get:24 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [212 B]
Get:25 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [552 B]
Get:26 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:27 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [112 B]
Get:28 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [10.6 kB]
Get:29 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [10.8 kB]
Get:30 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [17.6 kB]
Get:31 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1104 B]
Activate Windows
Go to Settings to activate Windows.





For node:
After this command

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"

Do
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

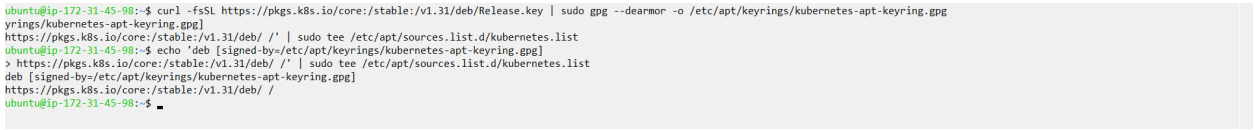
sudo apt-get update

sudo apt-get install -y docker-ce

Then
sudo mkdir -p /etc/docker

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

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



```
ubuntu@ip-172-31-34-167:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-34-167:~$ sudo systemctl enable --now kubelet
ubuntu@ip-172-31-34-167:~$ 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 25 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 53.1 MB disk space will be freed.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:2 http://ap-south-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 20s (2381 kB/s)
(Reading database ... 68159 files and directories currently installed.)
Removing docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Removing containerd.io (1.7.22-1) ...
Selecting previously unselected package runc.
(Reading database ... 68139 files and directories currently installed.)
Preparing to unpack .../runc_1.1.12-0ubuntu3.1_amd64.deb ...
Unpacking runc (1.1.12-0ubuntu3.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../containerd_1.7.12-0ubuntu4.1_amd64.deb ...
Unpacking containerd (1.7.12-0ubuntu4.1) ...
Setting up runc (1.1.12-0ubuntu3.1) ...
Setting up containerd (1.7.12-0ubuntu4.1) ...
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.
```

```
ubuntu@ip-172-31-34-167:~$ sudo mkdir -p /etc/containerd
ubuntu@ip-172-31-34-167:~$ 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 = ""
```

```
[timeouts]
  "io.containerd.timeout.bolt.open" = "0s"
  "io.containerd.timeout.metrics.shimstats" = "2s"
  "io.containerd.timeout.shim.cleanup" = "5s"
  "io.containerd.timeout.shim.load" = "5s"
  "io.containerd.timeout.shim.shutdown" = "3s"
  "io.containerd.timeout.task.state" = "2s"

[ttrpc]
  address = ""
  gid = 0
  uid = 0
ubuntu@ip-172-31-34-167:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-34-167:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-34-167:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Wed 2024-10-16 18:50:33 UTC; 20s ago
       Docs: https://containerd.io
   Main PID: 5775 (containerd)
      Tasks: 7
     Memory: 13.3M (peak: 13.8M)
        CPU: 149ms
     CGroup: /system.slice/containerd.service
             └─5775 /usr/bin/containerd

Oct 16 18:50:33 ip-172-31-34-167 containerd[5775]: time="2024-10-16T18:50:33.025724063Z" level=info msg="Start subscribing containerd event"
Oct 16 18:50:33 ip-172-31-34-167 containerd[5775]: time="2024-10-16T18:50:33.025726329Z" level=info msg="serving... address=/run/containerd/containerd.sock.ttrpc
Oct 16 18:50:33 ip-172-31-34-167 containerd[5775]: time="2024-10-16T18:50:33.025785445Z" level=info msg="Start recovering state"
Oct 16 18:50:33 ip-172-31-34-167 containerd[5775]: time="2024-10-16T18:50:33.025820256Z" level=info msg="serving... address=/run/containerd/containerd.sock
Oct 16 18:50:33 ip-172-31-34-167 containerd[5775]: time="2024-10-16T18:50:33.025864796Z" level=info msg="Start event monitor"
Oct 16 18:50:33 ip-172-31-34-167 containerd[5775]: time="2024-10-16T18:50:33.025881444Z" level=info msg="Start snapshots syncer"
Oct 16 18:50:33 ip-172-31-34-167 containerd[5775]: time="2024-10-16T18:50:33.025894133Z" level=info msg="Start cni network conf syncer for default"
Oct 16 18:50:33 ip-172-31-34-167 containerd[5775]: time="2024-10-16T18:50:33.025901025Z" level=info msg="Start streaming server"
Oct 16 18:50:33 ip-172-31-34-167 systemd[1]: Started containerd.service - containerd container runtime.
Oct 16 18:50:33 ip-172-31-34-167 containerd[5775]: time="2024-10-16T18:50:33.026718453Z" level=info msg="containerd successfully booted in 0.023924s"
ubuntu@ip-172-31-34-167:~$ sudo apt-get install -y socat
```

Now only master

```
ubuntu@ip-172-31-34-167:~$ 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 internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W1016 18:57:48.326442    6254 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.
io/pause:3.10" 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-172-31-34-167 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.34.167]
[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-172-31-34-167 localhost] and IPs [172.31.34.167 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-34-167 localhost] and IPs [172.31.34.167 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
```

```
ubuntu@ip-172-31-34-167:~$ mkdir -p $HOME/.kube
ubuntu@ip-172-31-34-167:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
ubuntu@ip-172-31-34-167:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-34-167:~$
```

```
ubuntu@ip-172-31-34-167:~$ kubectl get nodes
NAME                STATUS     ROLES           AGE     VERSION
ip-172-31-34-167    NotReady   control-plane   3m50s   v1.31.1
ubuntu@ip-172-31-34-167:~$
```

See the join command

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

kubeadm join 172.31.34.167:6443 --token ewtyfl.9m5ufbfy8tc9t0hz \
        --discovery-token-ca-cert-hash sha256:4882dd53302c198ee6d6a14953f4b8f03954272f0210b9fed99179ba9b8728d8
ubuntu@ip-172-31-34-167:~$ mkdir -p $HOME/.kube
ubuntu@ip-172-31-34-167:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
ubuntu@ip-172-31-34-167:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-34-167:~$ kubectl get nodes
NAME              STATUS     ROLES           AGE     VERSION
ip-172-31-34-167   NotReady   control-plane   3m50s   v1.31.1
ubuntu@ip-172-31-34-167:~$
```

simply run the command with sudo to execute it

Node 1

```
ubuntu@ip-172-31-32-233:~$ sudo kubeadm join 172.31.34.167:6443 --token ewtyfl.9m5ufbfy8tc9t0hz \
>     --discovery-token-ca-cert-hash sha256:4882dd53302c198ee6d6a14953f4b8f03954272f0210b9fed99179ba9b8728d8
[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 504.647743ms
[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-172-31-32-233:~$
```

Node 2

```
ubuntu@ip-172-31-45-98:~$ sudo kubeadm join 172.31.34.167:6443 --token ewtyfl.9m5ufbfy8tc9t0hz \
>     --discovery-token-ca-cert-hash sha256:4882dd53302c198ee6d6a14953f4b8f03954272f0210b9fed99179ba9b8728d8
[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 502.936588ms
[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-172-31-45-98:~$
```

On master run **kubectl get nodes**

```
ubuntu@ip-172-31-34-167:~$ kubectl get nodes
NAME                STATUS     ROLES           AGE    VERSION
ip-172-31-32-233    NotReady   <none>          110s   v1.31.1
ip-172-31-34-167    NotReady   control-plane   13m    v1.31.1
ip-172-31-45-98     NotReady   <none>          106s   v1.31.1
ubuntu@ip-172-31-34-167:~$
```

```
ubuntu@ip-172-31-34-167:~$ 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
```

```
ubuntu@ip-172-31-34-167:~$  kubectl get nodes
NAME              STATUS   ROLES           AGE    VERSION
ip-172-31-32-233   Ready    <none>          4m6s   v1.31.1
ip-172-31-34-167   Ready    control-plane   16m    v1.31.1
ip-172-31-45-98    Ready    <none>          4m2s   v1.31.1
ubuntu@ip-172-31-34-167:~$  kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-32-233   Ready    <none>          4m24s   v1.31.1   172.31.32.233   <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ip-172-31-34-167   Ready    control-plane   16m     v1.31.1   172.31.34.167   <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ip-172-31-45-98    Ready    <none>          4m20s   v1.31.1   172.31.45.98    <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ubuntu@ip-172-31-34-167:~$
```

Renaming the nodes

```
ubuntu@ip-172-31-34-167:~$ kubectl label node ip-172-31-32-233 kubernetes.io/role=Node1
node/ip-172-31-32-233 labeled
ubuntu@ip-172-31-34-167:~$ kubectl label node ip-172-31-45-98 kubernetes.io/role=Node2
node/ip-172-31-45-98 labeled
ubuntu@ip-172-31-34-167:~$ kubectl get nodes -o wide
NAME               STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-32-233   Ready    Node1           8m14s   v1.31.1   172.31.32.233   <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ip-172-31-34-167   Ready    control-plane   20m     v1.31.1   172.31.34.167   <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ip-172-31-45-98    Ready    Node2           8m10s   v1.31.1   172.31.45.98    <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ubuntu@ip-172-31-34-167:~$
```

```
ubuntu@ip-172-31-34-167:~$ kubectl label node ip-172-31-34-167 kubernetes.io/role=Master
node/ip-172-31-34-167 labeled
ubuntu@ip-172-31-34-167:~$ kubectl get nodes -o wide
NAME               STATUS   ROLES                  AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-32-233   Ready    Node1                  9m20s   v1.31.1   172.31.32.233   <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ip-172-31-34-167   Ready    Master,control-plane   21m     v1.31.1   172.31.34.167   <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ip-172-31-45-98    Ready    Node2                  9m16s   v1.31.1   172.31.45.98    <none>        Ubuntu 24.04.1 LTS   6.8.0-1016-aws   containerd://1.7.12
ubuntu@ip-172-31-34-167:~$
```

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.