Name: Pratik Patil

Academic Year:2024-2025

Division: D15C

Roll No: 40

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.

Name: Pratik Patil

Academic Year:2024-2025

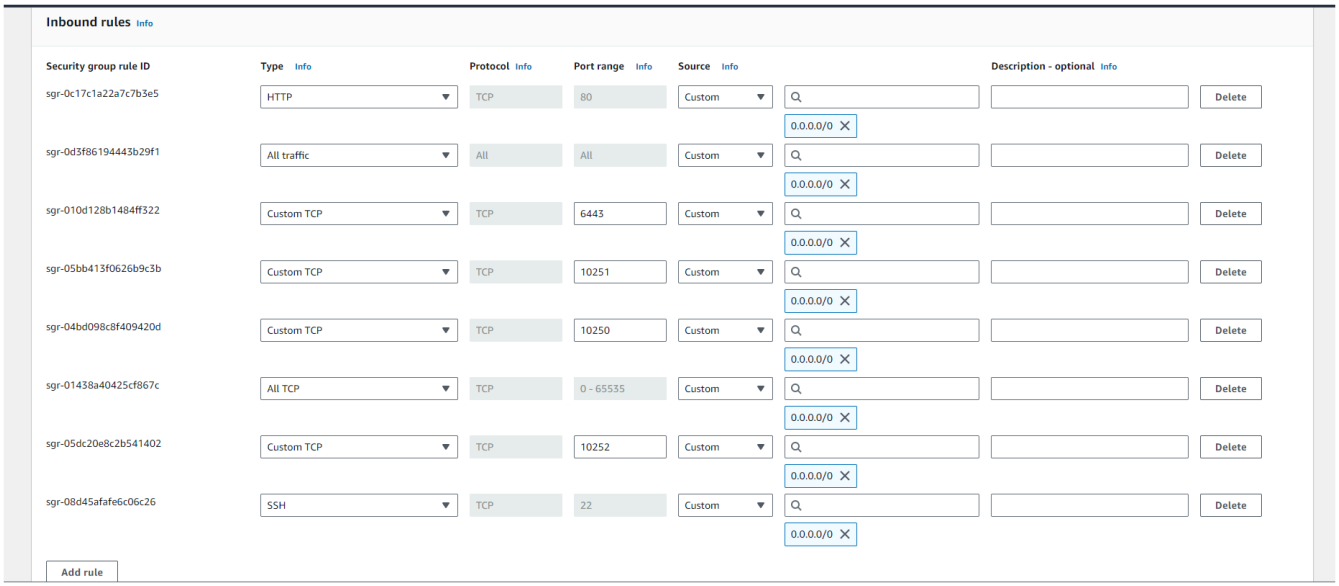
Division: D15C

Roll No: 40

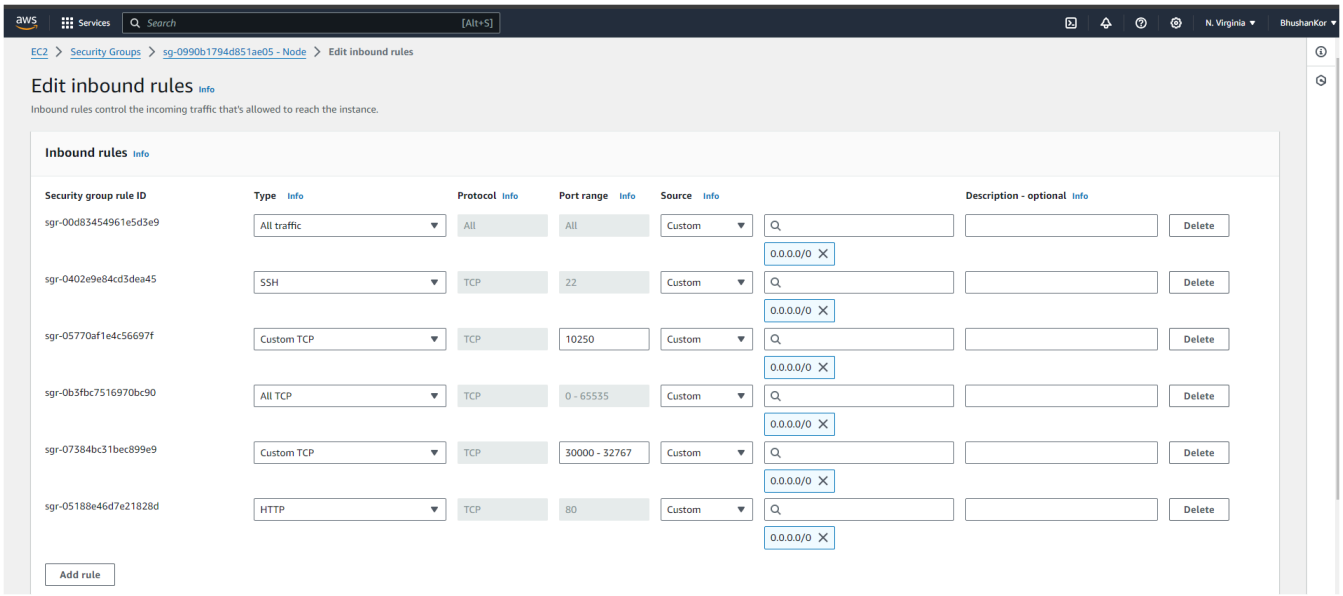
Prerequisites :

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

Master:



Node :



Name: Pratik Patil **Academic Year:2024-2025**

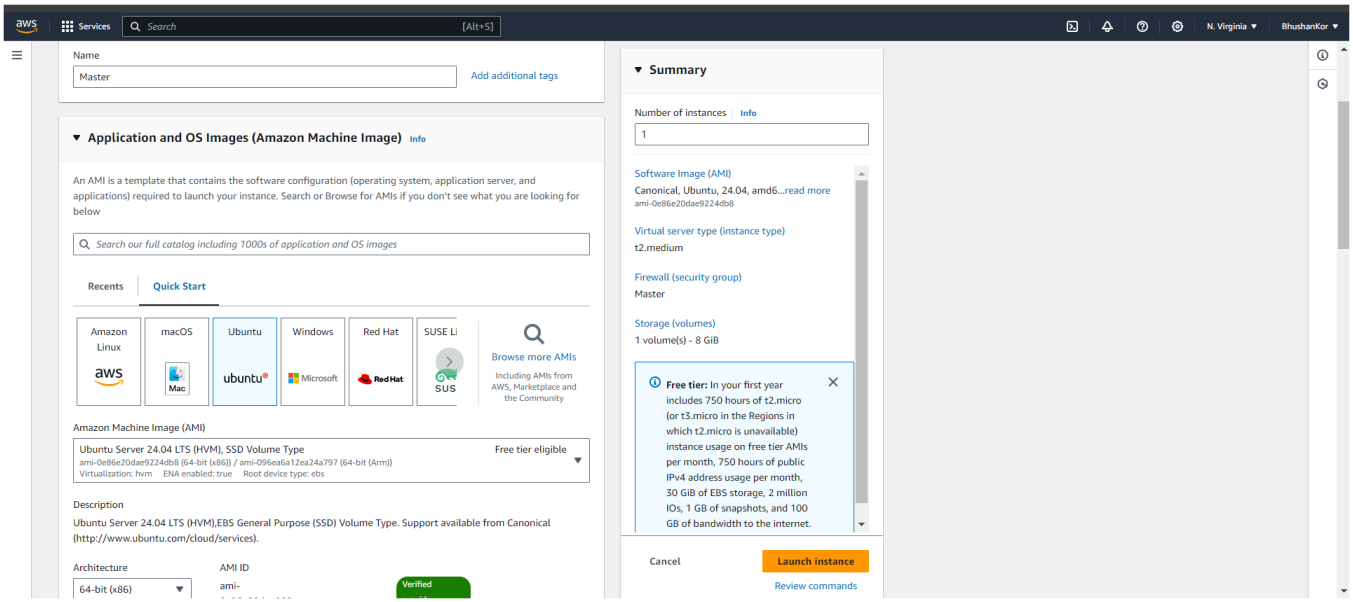
Division: D15C **Roll No: 40**

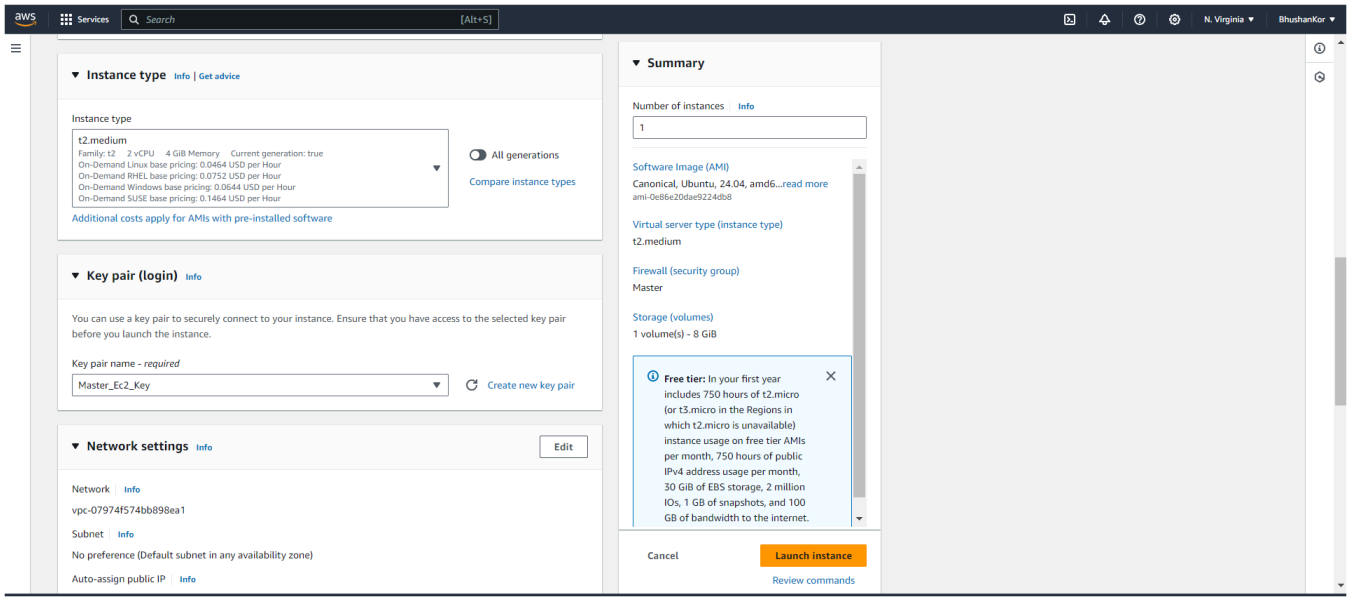
Step 1: Log in to your AWS Academy/personal account and launch 3 new Ec2 Instances. Select Ubuntu as AMI and t2.medium as Instance Type and create a key of type RSA with .pem extension and move the downloaded key to the new folder.We can use 3 Different keys or 1 common key also.

Note: A minimum of 2 CPUs are required so Please select t2.medium and do not forget to stop the instance after the experiment because it is not available in the free tier.

Also Select Security groups from existing.

Master:



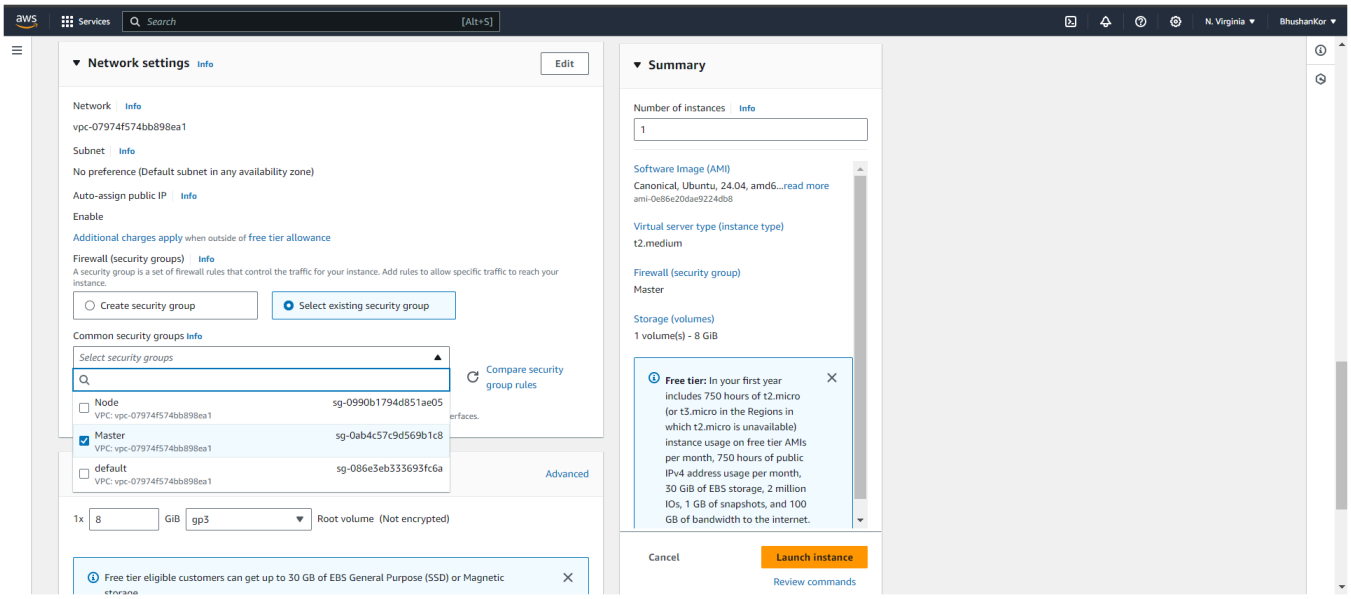


Name: Pratik Patil

Division: D15C

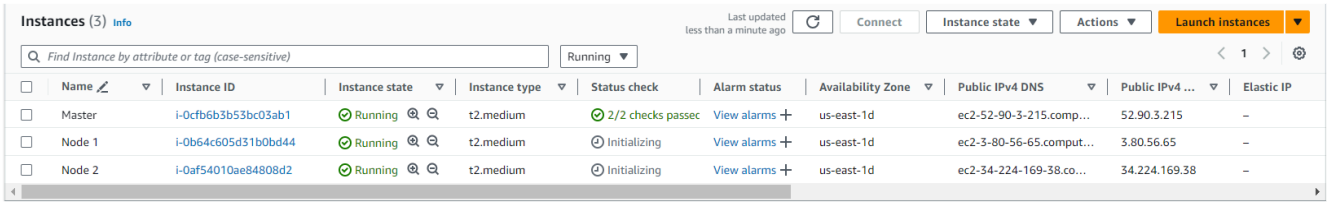
Academic Year:2024-2025

Roll No: 40

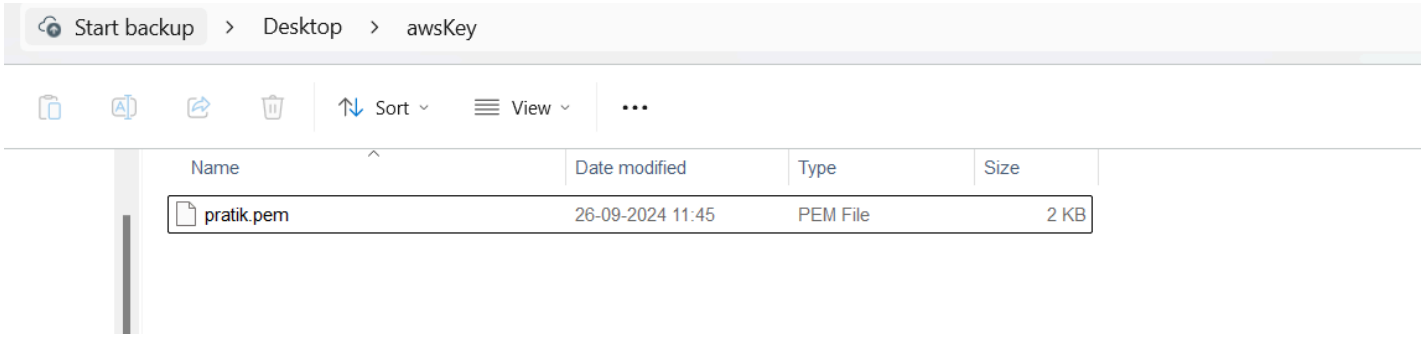


Do Same for 2 Nodes and use security groups of Node for that.

Step 2: After creating the instances click on Connect & connect all 3 instances and navigate to SSH Client.

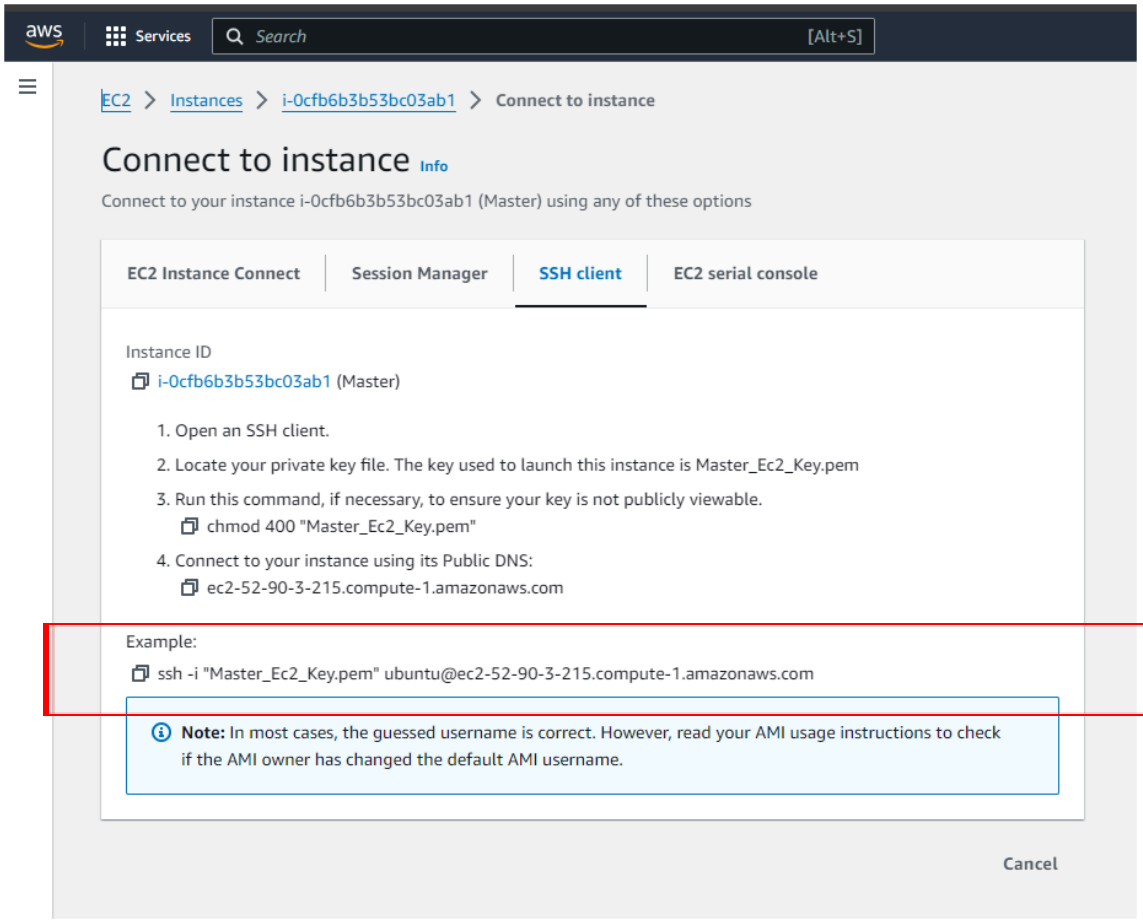


(Downloded Key



Step 3: 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 (starting with ssh -i …..) in the terminal.(ssh -i "Master_Ec2_Key.pem" ubuntu@ec2-54-196-129-215.compute-1.amazonaws.com)

Master:



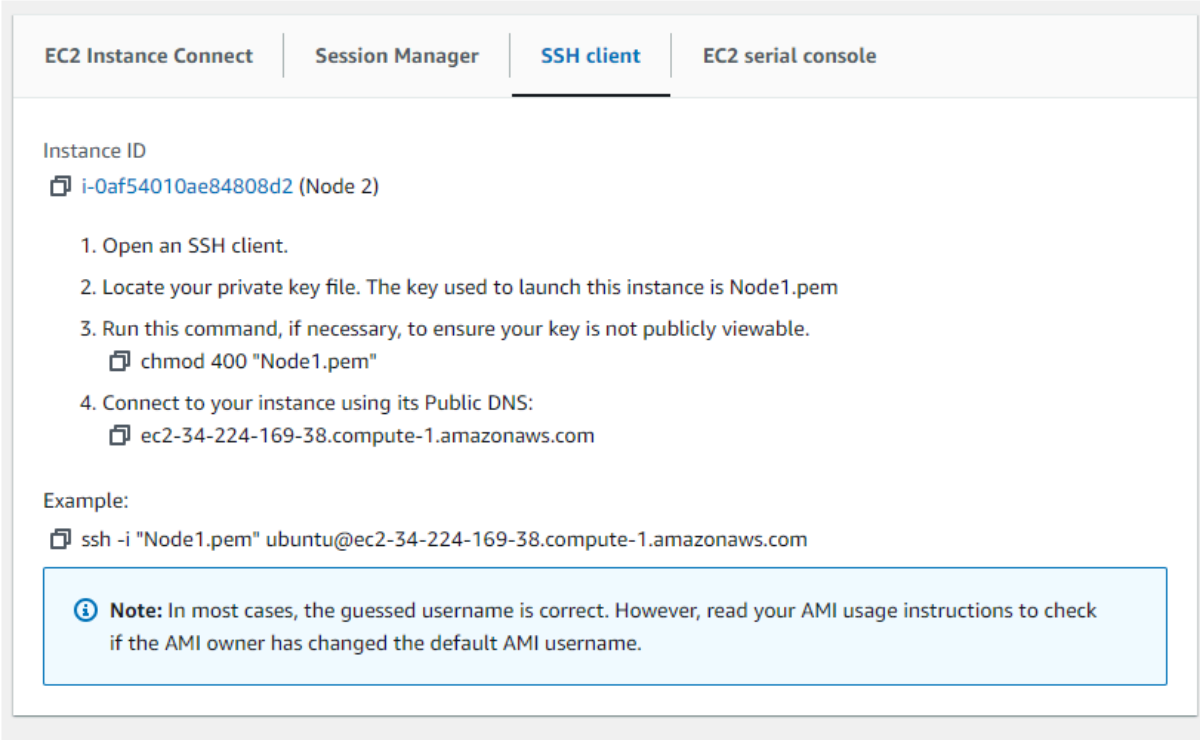
Name: Pratik Patil

Academic Year:2024-2025

Division: D15C

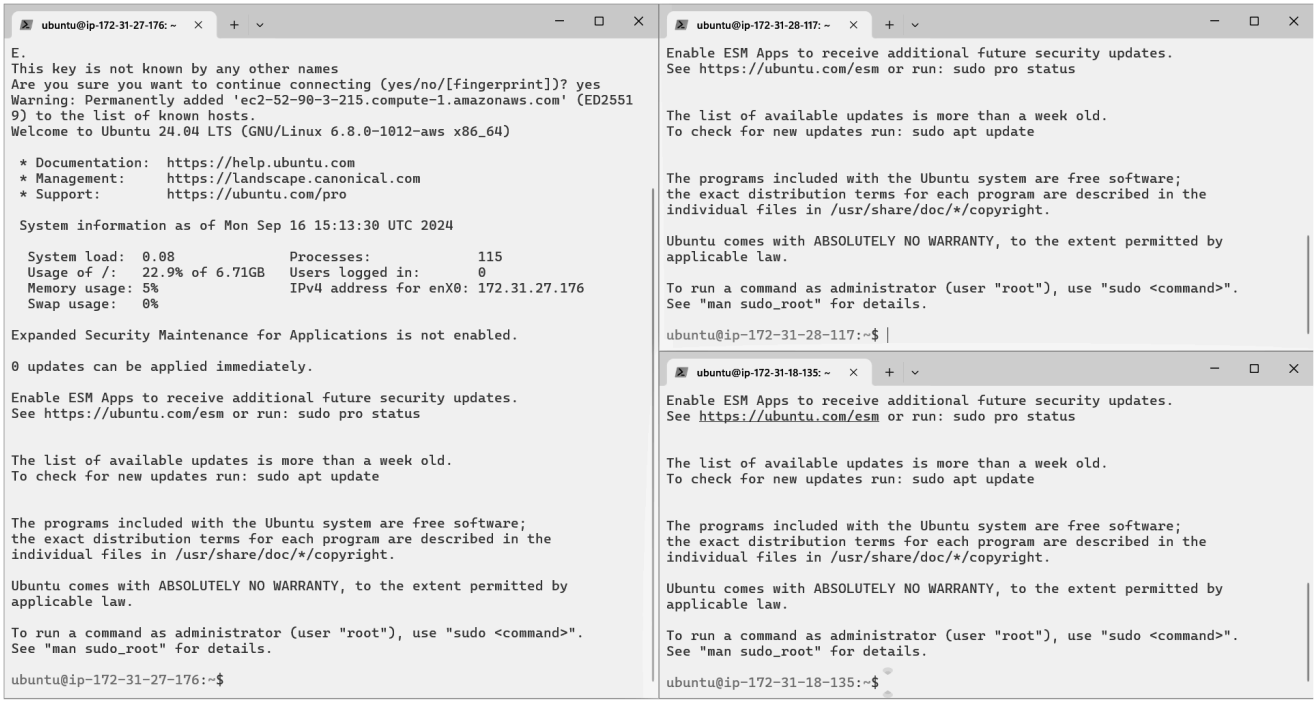
Roll No: 40

Node 2:



Here I have use 2 keys 1 for master and 1 for 2 node so I have to run open 3 terminals.In master key folder 1 terminal and 2 terminals in node 1 key folder.
If you use 1 Key only, you can open 3 terminal in one folder only.

Successful Connection:



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"

Name: Pratik Patil

Division: D15C

Academic Year:2024-2025

Roll No: 40

```
cted amd64 c-n-f Metadata [116 B]
Get:51 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiv
erse amd64 Components [212 B]
Get:52 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiv
erse amd64 c-n-f Metadata [116 B]
Fetched 28.9 MB in 4s (6976 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s
) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file h
as an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is st
ored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATI
ON section in apt-key(8) for details.
ubuntu@ip-172-31-27-176:~$
```

Name: Pratik Patil

Academic Year:2024-2025

Division: D15C

Roll No: 40

```
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service →
 /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /us
r/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
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-172-31-27-176:~$
```

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

```
ubuntu@ip-172-31-27-176:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-27-176:~$
```

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

```
ubuntu@ip-172-31-89-148:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart dockerSynchronizing state of docker.service with SysV ser
vice script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
```

Step 5: 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

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-172-31-89-148:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/de
b/Release.key | sudo gpg --dearmor -o
tes.listgpg: missing argument for option "-o"
ubuntu@ip-172-31-89-148:~$ /etc/apt/keyrings/kubernetes-apt-keyring.gpg
-bash: /etc/apt/keyrings/kubernetes-apt-keyring.gpg: Permission denied
ubuntu@ip-172-31-89-148:~$ 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.li
st.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

```
ubuntu@ip-172-31-89-148:~$ 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 142 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 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:/stabl
e:/v1.31/deb  cri-tools 1.31.1-1.1 [15.7 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stabl
e:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stabl
e:/v1.31/deb  kubectl 1.31.1-1.1 [11.2 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stabl
e:/v1.31/deb  kubernetes-cni 1.5.1-1.1 [33.9 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stabl
e:/v1.31/deb  kubelet 1.31.1-1.1 [15.2 MB]
Fetched 87.4 MB in 1s (89.9 MB/s)
```

```
ubuntu@ip-172-31-89-148:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
```

sudo systemctl enable --now kubelet
sudo apt-get install -y containerd

```
ubuntu@ip-172-31-89-148:~$ sudo systemctl enable --now kubelet
ubuntu@ip-172-31-89-148:~$ 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 142 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 ru
nc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 co
ntainerd amd64 1.7.12-0ubuntu4.1 [38.6 MB]
Fetched 47.2 MB in 1s (81.3 MB/s)
```

Name: Pratik Patil

Academic Year:2024-2025

Division: D15C

Roll No: 40

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

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-89-148:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
ubuntu@ip-172-31-89-148:~$ sudo mkdir -p /etc/containerd
ubuntu@ip-172-31-89-148:~$ sudo containerd config default | sudo tee /etc/contai
nerd/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
sudo systemctl enable containerd sudo systemctl status containerd

```
ubuntu@ip-172-31-89-148:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-89-148:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-89-148:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; prese>
     Active: active (running) since Thu 2024-09-26 08:32:08 UTC; 17s ago
       Docs: https://containerd.io
   Main PID: 2605 (containerd)
      Tasks: 7
     Memory: 13.5M (peak: 13.8M)
        CPU: 134ms
     CGroup: /system.slice/containerd.service
             └─2605 /usr/bin/containerd

Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
```

Name: Pratik Patil

Academic Year:2024-2025

Division: D15C

Roll No: 40

sudo apt-get install -y socat

```
ubuntu@ip-172-31-89-148:~$ 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 142 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 amd6
4 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (15.6 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.
```

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-172-31-89-148:~$ kubeadm join 172.31.84.180:6443 --token i0v7v8.m228gg
uv7d2qavun \
>         --discovery-token-ca-cert-hash sha256:4288044fe587af6e8bc218c177eef151
58edb9ea12287885006bb474f2f264d3
[preflight] Running pre-flight checks
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR IsPrivilegedUser]: user is not running as root
[preflight] If you know what you are doing, you can make a check non-fatal with
`--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
ubuntu@ip-172-31-89-148:~$ kubeadm join 172.31.84.180:6443 --token i0v7v8.m228gg
```

```
ubuntu@ip-172-31-89-148:~$ sudo kubeadm join 172.31.84.180:6443 --token i0v7v8.m
228gguv7d2qavun \
>     --discovery-token-ca-cert-hash sha256:4288044fe587af6e8bc218c177eef15158ed
b9ea12287885006bb474f2f264d3
[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 g
et cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.y
aml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/ku
belet/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 501.011949ms
[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.
```

Run this command on master and also copy and save the Join command from above. 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-172-31-27-176:~$ mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-27-176:~$
```

Name: Pratik Patil

Academic Year:2024-2025

Division: D15C

Roll No: 40

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

```
ubuntu@ip-172-31-84-180:~$ kubectlghet nodes
kubectlghet: command not found
ubuntu@ip-172-31-84-180:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE    VERSION
ip-172-31-84-180    Ready    control-plane   123m   v1.31.1
ip-172-31-89-148    Ready    <none>          2m3s   v1.31.1
ubuntu@ip-172-31-84-180:~$ sudo hostnamectl set-hostname node1
```

Step 8: Now Run the following command on Node 1 and Node 2 to Join to master.
**sudo kubeadm join 172.31.27.176:6443 --token ttay2x.n0sqeukjai8sgfg3 **
--discovery-token-ca-cert-hash
sha256:d6fc5fb7e984c83e2807780047fec6c4f2acfe9da9184ecc028d77157608fbb6

Node 1:

```
ubuntu@ip-172-31-27-176:~$ 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 2024-09-16 15:40:01 UTC; 11min ago
       Docs: https://kubernetes.io/docs/
   Main PID: 5989 (kubelet)
      Tasks: 10 (limit: 4676)
     Memory: 32.6M (peak: 33.2M)
        CPU: 10.705s
     CGroup: /system.slice/kubelet.service
             └─5989 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf --config=/var/

Sep 16 15:51:29 ip-172-31-27-176 kubelet[5989]: I0916 15:51:29.497458    5989 reconciler_common.go:245] "operationExecutor.VerifyControllerAttachedVolume s
Sep 16 15:51:29 ip-172-31-27-176 kubelet[5989]: I0916 15:51:29.497516    5989 reconciler_common.go:245] "operationExecutor.VerifyControllerAttachedVolume s
Sep 16 15:51:29 ip-172-31-27-176 kubelet[5989]: I0916 15:51:29.497569    5989 reconciler_common.go:245] "operationExecutor.VerifyControllerAttachedVolume s
Sep 16 15:51:29 ip-172-31-27-176 kubelet[5989]: I0916 15:51:29.497620    5989 reconciler_common.go:245] "operationExecutor.VerifyControllerAttachedVolume s
Sep 16 15:51:29 ip-172-31-27-176 kubelet[5989]: I0916 15:51:29.497669    5989 reconciler_common.go:245] "operationExecutor.VerifyControllerAttachedVolume s
Sep 16 15:51:29 ip-172-31-27-176 kubelet[5989]: I0916 15:51:29.497719    5989 reconciler_common.go:245] "operationExecutor.VerifyControllerAttachedVolume s
Sep 16 15:51:31 ip-172-31-27-176 kubelet[5989]: E0916 15:51:31.605091    5989 kubelet.go:2902] "Container runtime network not ready" networkReady="NetworkR
Sep 16 15:51:32 ip-172-31-27-176 kubelet[5989]: I0916 15:51:32.366237    5989 scope.go:117] "RemoveContainer" containerID="f44f06967c5b3e567e07841a7b4352ae
Sep 16 15:51:36 ip-172-31-27-176 kubelet[5989]: E0916 15:51:36.606675    5989 kubelet.go:2902] "Container runtime network not ready" networkReady="NetworkR
Sep 16 15:51:41 ip-172-31-27-176 kubelet[5989]: E0916 15:51:41.608404    5989 kubelet.go:2902] "Container runtime network not ready" networkReady="NetworkR
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-172-31-84-180:~$ kubectlghet nodes
kubectlghet: command not found
ubuntu@ip-172-31-84-180:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE    VERSION
ip-172-31-84-180   Ready    control-plane   123m   v1.31.1
ip-172-31-89-148   Ready    <none>          2m3s   v1.31.1
ubuntu@ip-172-31-84-180:~$ sudo hostnamectl set-hostname node1
ubuntu@ip-172-31-84-180:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE    VERSION
ip-172-31-84-180   Ready    control-plane   124m   v1.31.1
ip-172-31-89-148   Ready    <none>          3m8s   v1.31.1
ubuntu@ip-172-31-84-180:~$
```

Conclusion :-

Kubernetes is a powerful tool for managing containerized applications at scale, providing essential capabilities for orchestrating and automating deployment, scaling, and maintenance of these applications. By abstracting the complexity of the underlying infrastructure, Kubernetes enables development teams to focus on building and deploying applications without worrying about the operational details.

In this experiment, launching a **t2.medium EC2 instance** (with at least 2 CPUs and sufficient memory) ensured that the Kubernetes cluster could handle the resource demands effectively. With Kubernetes, tasks like load balancing, resource management, self-healing of applications, and scaling across multiple nodes are automated, making it a crucial technology for modern micr