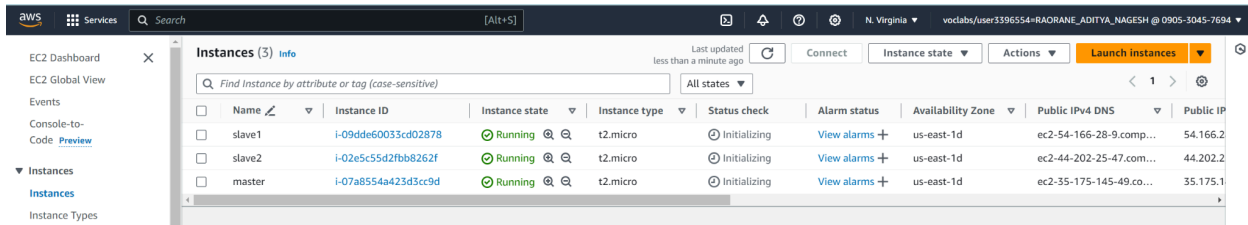


EXPERIMENT NO. 3

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

1. Create 3 EC-2 instances with all running on Amazon Linux as OS with inbound SSH allowed.

To efficient run kubernetes cluster select instance type of at least t2.micro

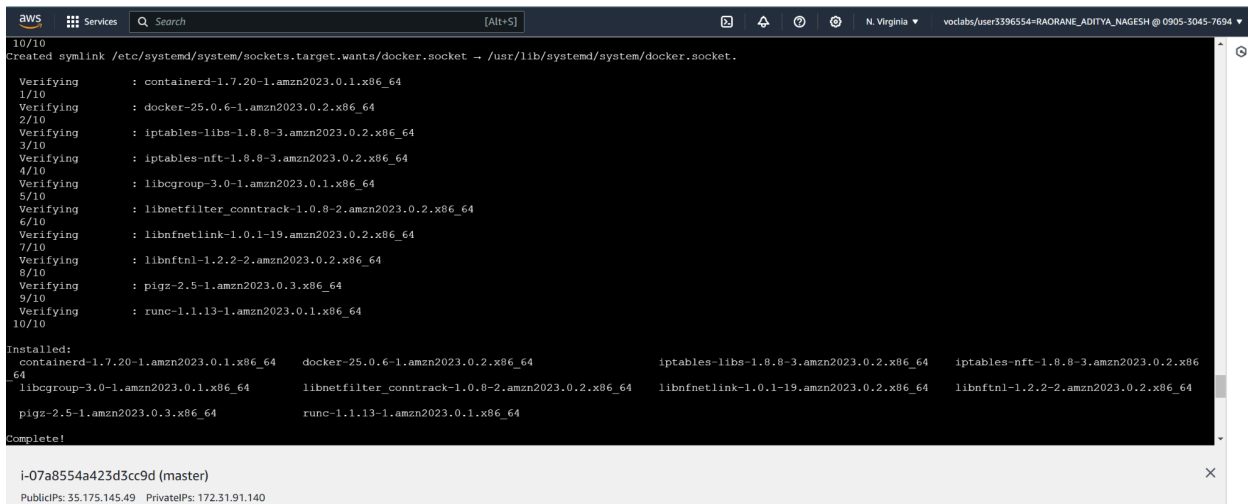


2. Perform these steps on all 3 machines.

a]Install Docker

Command : `sudo yum install docker -y`

Master:-




```

},
"storage-driver": "overlay2"
}
EOF

```

Master:-

```

[ec2-user@ip-172-31-91-140 docker]$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
[ec2-user@ip-172-31-91-140 docker]$

```

i-07a8554a423d3cc9d (master)

PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140

Slave1:-

```

AWS Services Q Search [Alt+S] N. Virginia vodabs/user3396554=RAORANE_ADITYA_NAGESH @ 0905-3045-7694
Verifying : libcgroup-3.0-1.amzn2023.0.1.x86_64 5/10
Verifying : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64 6/10
Verifying : libnftnl-1.0.1-19.amzn2023.0.2.x86_64 7/10
Verifying : libnftnl-1.2.2-2.amzn2023.0.2.x86_64 8/10
Verifying : pigz-2.5-1.amzn2023.0.3.x86_64 9/10
Verifying : runc-1.1.13-1.amzn2023.0.1.x86_64 10/10
Installed:
containerd-1.7.20-1.amzn2023.0.1.x86_64 docker-25.0.6-1.amzn2023.0.2.x86_64 iptables-libse-1.8.8-3.amzn2023.0.2.x86_64 iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
libcgroup-3.0-1.amzn2023.0.1.x86_64 libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64 libnftnl-1.0.1-19.amzn2023.0.2.x86_64 libnftnl-1.2.2-2.amzn2023.0.2.x86_64
pigz-2.5-1.amzn2023.0.3.x86_64 runc-1.1.13-1.amzn2023.0.1.x86_64
Complete!
[ec2-user@ip-172-31-90-5 ~]$ cd /etc/docker
[ec2-user@ip-172-31-90-5 docker]$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
i-09dde60033cd02878 (slave1)
PublicIPs: 54.166.28.9 PrivateIPs: 172.31.90.5

```

Slave2:-

```

aws
Services
Q Search [Alt+S]
N. Virginia voclabs/user3396554-RAORANE_ADITYA_NAGESH @ 0905-3045-7694

Verifying : libcgrouper-3.0-1.amzn2023.0.1.x86_64 5/10
Verifying : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64 6/10
Verifying : libnftnl-1.0.1-19.amzn2023.0.2.x86_64 7/10
Verifying : libnftnl-1.2.2-2.amzn2023.0.2.x86_64 8/10
Verifying : pigz-2.5-1.amzn2023.0.3.x86_64 9/10
Verifying : runc-1.1.13-1.amzn2023.0.1.x86_64 10/10

Installed:
containerd-1.7.20-1.amzn2023.0.1.x86_64 docker-25.0.6-1.amzn2023.0.2.x86_64 iptables-libs-1.8.8-3.amzn2023.0.2.x86_64 iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
libcgrouper-3.0-1.amzn2023.0.1.x86_64 libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64 libnftnl-1.0.1-19.amzn2023.0.2.x86_64 libnftnl-1.2.2-2.amzn2023.0.2.x86_64
pigz-2.5-1.amzn2023.0.3.x86_64 runc-1.1.13-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-172-31-91-65 ~]$ cd /etc/docker
[ec2-user@ip-172-31-91-65 docker]$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}

i-02e5c55d2fbb8262f (slave2)
PublicIPs: 44.202.25.47 PrivateIPs: 172.31.91.65

```

c) Run these commands to enable docker.

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

```

[ec2-user@ip-172-31-91-140 docker]$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
docker -v
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-91-140 docker]$

```

i-07a8554a423d3cc9d (master)

PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140

```

[ec2-user@ip-172-31-90-5 docker]$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
docker -v
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-90-5 docker]$

```

i-09dde60033cd02878 (slave1)

PublicIPs: 54.166.28.9 PrivateIPs: 172.31.90.5

```

[ec2-user@ip-172-31-91-65 docker]$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
docker -v
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-91-65 docker]$

```

i-02e5c55d2fbb8262f (slave2)

PublicIPs: 44.202.25.47 PrivateIPs: 172.31.91.65

3. Install Kubernetes on all 3 machines

a]SELinux needs to be disabled before configuring kubelet.

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

```
[ec2-user@ip-172-31-91-140 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-91-140 docker]$
```

i-07a8554a423d3cc9d (master)

PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140

```
[ec2-user@ip-172-31-90-5 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-90-5 docker]$
```

i-09dde60033cd02878 (slave1)

PublicIPs: 54.166.28.9 PrivateIPs: 172.31.90.5

```
[ec2-user@ip-172-31-91-65 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-91-65 docker]$
```

i-02e5c55d2fbb8262f (slave2)

PublicIPs: 44.202.25.47 PrivateIPs: 172.31.91.65

b] Add kubernetes repository :-

```
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

```
[ec2-user@ip-172-31-91-140 docker]$ cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[ec2-user@ip-172-31-91-140 docker]$
```

i-07a8554a423d3cc9d (master)

PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140

```
[ec2-user@ip-172-31-90-5 docker]$ cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[ec2-user@ip-172-31-90-5 docker]$
```

i-09dde60033cd02878 (slave1)

PublicIPs: 54.166.28.9 PrivateIPs: 172.31.90.5

```
[ec2-user@ip-172-31-91-65 docker]$ cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[ec2-user@ip-172-31-91-65 docker]$
```

i-02e5c55d2fbb8262f (slave2)

PublicIPs: 44.202.25.47 PrivateIPs: 172.31.91.65

c] Type following commands to install set of kubernetes packages:

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

```
[ec2-user@ip-172-31-91-140 docker]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:02:56 ago on Sat Sep 14 06:37:25 2024.
Dependencies resolved.
=====
Package                               Architecture Version                               Repository
Size
=====
Installing:
kubeadm                               x86_64    1.30.5-150500.1.1                     kubernetes
10 M
kubectl                               x86_64    1.30.5-150500.1.1                     kubernetes
10 M
kubelet                                x86_64    1.30.5-150500.1.1                     kubernetes
17 M
Installing dependencies:
conntrack-tools                       x86_64    1.4.6-2.amzn2023.0.2                  amazonlinux
cri-tools                             x86_64    1.30.1-150500.1.1                     kubernetes
8.6 M
kubernetes-cni                        x86_64    1.4.0-150500.1.1                     kubernetes
6.7 M
libnetfilter_athelper                 x86_64    1.0.0-21.amzn2023.0.2                  amazonlinux
24 k
libnetfilter_athelper                 x86_64    1.0.0-19.amzn2023.0.2                  amazonlinux
24 k
libnetfilter_queue                    x86_64    1.0.5-2.amzn2023.0.2                  amazonlinux
30 k
=====

i-07a8554a423d3cc9d (master)
PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140
```

```

AWS Services Search [Alt+S] N. Virginia voclabs/user3396554=RAORANE_ADITYA_NAGESH @ 0905-3045-7694
Complete!
[ec2-user@ip-172-31-90-5 docker]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:02:58 ago on Sat Sep 14 06:37:32 2024.
Dependencies resolved.
=====
Package                                Architecture      Version           Repository        Size
-----
Installing:
kubeadm                                x86_64            1.30.5-150500.1.1  kubernetes        10 M
kubectl                                x86_64            1.30.5-150500.1.1  kubernetes        10 M
kubelet                                 x86_64            1.30.5-150500.1.1  kubernetes        17 M
Installing dependencies:
conntrack-tools                        x86_64            1.4.6-2.amzn2023.0.2  amazonlinux        208 k
cri-tools                              x86_64            1.30.1-150500.1.1    kubernetes         8.6 M
kubernetes-cni                         x86_64            1.4.0-150500.1.1    kubernetes         6.7 M
libnetfilter_cthelper                  x86_64            1.0.0-21.amzn2023.0.2  amazonlinux         24 k
libnetfilter_cttimeout                 x86_64            1.0.0-19.amzn2023.0.2  amazonlinux         24 k
libnetfilter_queue                     x86_64            1.0.5-2.amzn2023.0.2  amazonlinux         30 k
Transaction Summary
-----
Install 9 Packages
Total download size: 53 M
Installed size: 292 M
Downloading Packages:
(1/9): libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64.rpm 344 kB/s | 24 kB 00:00
(2/9): libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64.rpm 328 kB/s | 24 kB 00:00
(3/9): conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64.rpm 2.3 MB/s | 208 kB 00:00
(4/9): libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64.rpm 1.1 MB/s | 30 kB 00:00
(5/9): cri-tools-1.30.1-150500.1.1.x86_64.rpm 33 MB/s | 8.6 MB 00:00
(6/9): kubeadm-1.30.5-150500.1.1.x86_64.rpm 31 MB/s | 10 MB 00:00

i-09dde60033cd02878 (slave1)
PublicIPs: 54.166.28.9 PrivateIPs: 172.31.90.5

```

```

AWS Services Search [Alt+S] N. Virginia voclabs/user3396554=RAORANE_ADITYA_NAGESH @ 0905-3045-7694
Complete!
[ec2-user@ip-172-31-91-65 docker]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:02:59 ago on Sat Sep 14 06:37:37 2024.
Dependencies resolved.
=====
Package                                Architecture      Version           Repository        Size
-----
Installing:
kubeadm                                x86_64            1.30.5-150500.1.1  kubernetes        10 M
kubectl                                x86_64            1.30.5-150500.1.1  kubernetes        10 M
kubelet                                 x86_64            1.30.5-150500.1.1  kubernetes        17 M
Installing dependencies:
conntrack-tools                        x86_64            1.4.6-2.amzn2023.0.2  amazonlinux        208 k
cri-tools                              x86_64            1.30.1-150500.1.1    kubernetes         8.6 M
kubernetes-cni                         x86_64            1.4.0-150500.1.1    kubernetes         6.7 M
libnetfilter_cthelper                  x86_64            1.0.0-21.amzn2023.0.2  amazonlinux         24 k
libnetfilter_cttimeout                 x86_64            1.0.0-19.amzn2023.0.2  amazonlinux         24 k
libnetfilter_queue                     x86_64            1.0.5-2.amzn2023.0.2  amazonlinux         30 k
Transaction Summary
-----
Install 9 Packages
Total download size: 53 M
Installed size: 292 M
Downloading Packages:
(1/9): libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64.rpm 399 kB/s | 24 kB 00:00
(2/9): libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64.rpm 374 kB/s | 24 kB 00:00
(3/9): conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64.rpm 2.9 MB/s | 208 kB 00:00
(4/9): libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64.rpm 1.2 MB/s | 30 kB 00:00
(5/9): cri-tools-1.30.1-150500.1.1.x86_64.rpm 22 MB/s | 8.6 MB 00:00
(6/9): kubeadm-1.30.5-150500.1.1.x86_64.rpm 23 MB/s | 10 MB 00:00

i-02e5c55d2fbb8262f (slave2)
PublicIPs: 44.202.25.47 PrivateIPs: 172.31.91.65

```

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

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


```
[ec2-user@ip-172-31-91-140 docker]$ sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-91-140 docker]$
```

i-07a8554a423d3cc9d (master)

PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140

```
[ec2-user@ip-172-31-90-5 docker]$ sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-90-5 docker]$
```

i-09dde60033cd02878 (slave1)

PublicIPs: 54.166.28.9 PrivateIPs: 172.31.90.5

```
[ec2-user@ip-172-31-91-65 docker]$ sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-91-65 docker]$
```

i-02e5c55d2fbb8262f (slave2)

PublicIPs: 44.202.25.47 PrivateIPs: 172.31.91.65

4. Perform this ONLY on the Master machine

a] Initialize kubernetes by typing below command

- ❖ `sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all`

b] Copy the mkdir and chown commands from the top and execute them.

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

```
aws Services Search [Alt+S] N. Virginia voclabs/user3396554-RAORANE_ADITYA_NAGESH @ 0905-3045-7694
[ec2-user@ip-172-31-91-140 docker]$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all
[0914 06:45:25.366258] 28511 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.4
[preflight] Running pre-flight checks
[WARNING NumCPU]: the number of available CPUs 1 is less than the required 2
[WARNING Mem]: the system RAM (949 MB) is less than the minimum 1700 MB
[WARNING FileExisting-socat]: socat not found in system path
[WARNING FileExisting-tc]: tc not found in system path
[WARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable kubelet.service'
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[0914 06:45:25.613053] 28511 checks.go:844] 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:3.9" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-91-140.ec2.internal kubernet.es default kubernet.es default.svc kubernet.es default.svc.cluster.local] and IPs [10.96.0.1 172.31.91.140]
[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-172-31-91-140.ec2.internal localhost] and IPs [172.31.91.140 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-91-140.ec2.internal localhost] and IPs [172.31.91.140 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "ca" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file

i-07a8554a423d3cc9d (master)
PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140
```

```
aws Services Search [Alt+S] N. Virginia voclabs/user3396554-RAORANE_ADITYA_NAGESH @ 0905-3045-7694
[bootstrap-token] Using token: 16d2xj.bm6gcn6bct3o1rh6
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.91.140:6443 --token 16d2xj.bm6gcn6bct3o1rh6 \
  --discovery-token-ca-cert-hash sha256:5aa079e1e33b46536187519e5496ba9a2290627b56b511607f2078b90f0c6607
[ec2-user@ip-172-31-91-140 docker]$

i-07a8554a423d3cc9d (master)
PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140
```

c] Use this link to join the other nodes :-

`kubeadm join 172.31.91.140:6443 --token 16d2xj.bm6gcn6bct3o1rh6 \`

`--discovery-token-ca-cert-hash`

`sha256:5aa079e1e33b46536187519e5496ba9a2290627b56b511607f2078b90f0c6607`

d] Then, add a common networking plugin called flannel file as mentioned in the code.

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

```
[ec2-user@ip-172-31-91-140 docker]$ mkdir -p $HOME/.kube
[sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-91-140 docker]$ ^C
[ec2-user@ip-172-31-91-140 docker]$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
[ec2-user@ip-172-31-91-140 docker]$ kubectl get pods
No resources found in default namespace.
[ec2-user@ip-172-31-91-140 docker]$
```

i-07a8554a423d3cc9d (master)

PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140

e] Check the created pod using this command

- ❖ `kubectl get pods`

5. Perform this ONLY on the slave machines

a] Paste the below command on all 2 slave machines

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

```

aws
Services Search [Alt+S] N. Virginia vodlabs/user3396554-RAORANE_ADITYA_NAGESH @ 0905-3045-7694

Package Architecture Version Repository Size
Installing:
iproute-tc x86_64 5.10.0-2.amzn2023.0.5 amazonlinux 455 k

Transaction Summary
-----
Install 1 Package

Total download size: 455 k
Installed size: 928 k
Downloading Packages:
iproute-tc-5.10.0-2.amzn2023.0.5.x86_64.rpm 4.2 MB/s | 455 kB 00:00
-----
Total 2.6 MB/s | 455 kB 00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :
  Installing      : iproute-tc-5.10.0-2.amzn2023.0.5.x86_64 1/1
  Running scriptlet: iproute-tc-5.10.0-2.amzn2023.0.5.x86_64 1/1
  Verifying      : iproute-tc-5.10.0-2.amzn2023.0.5.x86_64 1/1

Installed:
iproute-tc-5.10.0-2.amzn2023.0.5.x86_64

Complete!
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service -> /usr/lib/systemd/system/kubelet.service.
[ec2-user@ip-172-31-90-5 docker]$

i-09dde60033cd02878 (slave1)
PublicIPs: 54.166.28.9 PrivateIPs: 172.31.90.5

```

```

aws
Services Search [Alt+S] N. Virginia vodlabs/user3396554-RAORANE_ADITYA_NAGESH @ 0905-3045-7694

Package Architecture Version Repository Size
Installing:
iproute-tc x86_64 5.10.0-2.amzn2023.0.5 amazonlinux 455 k

Transaction Summary
-----
Install 1 Package

Total download size: 455 k
Installed size: 928 k
Downloading Packages:
iproute-tc-5.10.0-2.amzn2023.0.5.x86_64.rpm 6.4 MB/s | 455 kB 00:00
-----
Total 4.1 MB/s | 455 kB 00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :
  Installing      : iproute-tc-5.10.0-2.amzn2023.0.5.x86_64 1/1
  Running scriptlet: iproute-tc-5.10.0-2.amzn2023.0.5.x86_64 1/1
  Verifying      : iproute-tc-5.10.0-2.amzn2023.0.5.x86_64 1/1

Installed:
iproute-tc-5.10.0-2.amzn2023.0.5.x86_64

Complete!
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service -> /usr/lib/systemd/system/kubelet.service.
[ec2-user@ip-172-31-91-65 docker]$

i-02e5c55d2fbb8262f (slave2)
PublicIPs: 44.202.25.47 PrivateIPs: 172.31.91.65

```

b) Run this command to join the master machine.

❖ `kubeadm join 172.31.91.140:6443 --token 16d2xj.bm6gcn6bct3o1rh6 \`
`--discovery-token-ca-cert-hash`
`sha256:5aa079e1e33b46536187519e5496ba9a2290627b56b511607f2078`
`b90f0c6607`

```

[ec2-user@ip-172-31-90-5 docker]$ sudo kubeadm join 172.31.91.140:6443 --token s1kpn9.1m1gky28v0ublrer \
--discovery-token-ca-cert-hash sha256:5aa079e1e33b46536187519e5496ba9a2290627b56b511607f2078b90f0c6607
[preflight] Running pre-flight checks
[WARNING FileExisting-socat]: socat not found in system path
error execution phase preflight: couldn't validate the identity of the API Server: failed to request the cluster-info ConfigMap: Get "https://172.31.91.140:6443/api/v1/namespaces/kube-public/configmaps/cluster-info?timeout=10s": context deadline exceeded
To see the stack trace of this error execute with --v=5 or higher
[ec2-user@ip-172-31-90-5 docker]$

i-09dde60033cd02878 (slave1)
PublicIPs: 54.166.28.9 PrivateIPs: 172.31.90.5

```

```
[ec2-user@ip-172-31-91-65 docker]$ sudo kubeadm join 172.31.91.140:6443 --token zlkpn9.1mlgky28v0ublrer \
--discovery-token-ca-cert-hash sha256:5aa079e1e33b46536187519e5496ba9a2290627b56b511607f2078b90f0c6607
[preflight] Running pre-flight checks
[WARNING FileExisting-socat]: socat not found in system path
error execution phase preflight: couldn't validate the identity of the API Server: failed to request the cluster-info ConfigMap: client rate limiter Wait returned an error: rate: Wa
it(n=1) would exceed context deadline
To see the stack trace of this error execute with --v=5 or higher
[ec2-user@ip-172-31-91-65 docker]$
```

i-02e5c55d2fbb8262f (slave2)
PublicIPs: 44.202.25.47 PrivateIPs: 172.31.91.65

c] Now we can see in the master/control node of kubernetes that worker nodes are connected by typing **watch kubectl get nodes** in the **master node instance**.

Note:-

The "deadline exceeded" error likely occurred due to connectivity issues, firewall restrictions, expired tokens, or misconfigurations preventing the node from properly communicating with the Kubernetes control plane.

Conclusion:

We explored Kubernetes Cluster Architecture by setting up and configuring a Kubernetes cluster on AWS EC2 Linux instances. After launching EC2 machines, we installed Docker as the container runtime and deployed Kubernetes using **kubeadm**, **kubelet**, **kubectl**. During the node-joining process, we encountered common challenges such as network connectivity issues and token expiration. Overcoming these highlighted the importance of proper network configuration, firewall settings, and token management, ensuring smooth communication between nodes and the control plane in the cluster.