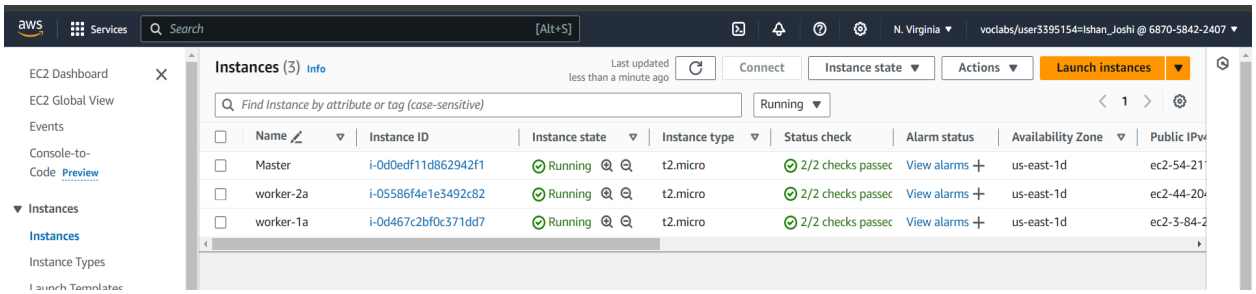
Name-Ishan Kiran Joshi Div-D15C Roll No-21 A.Y.-2024-25

Experiment 3

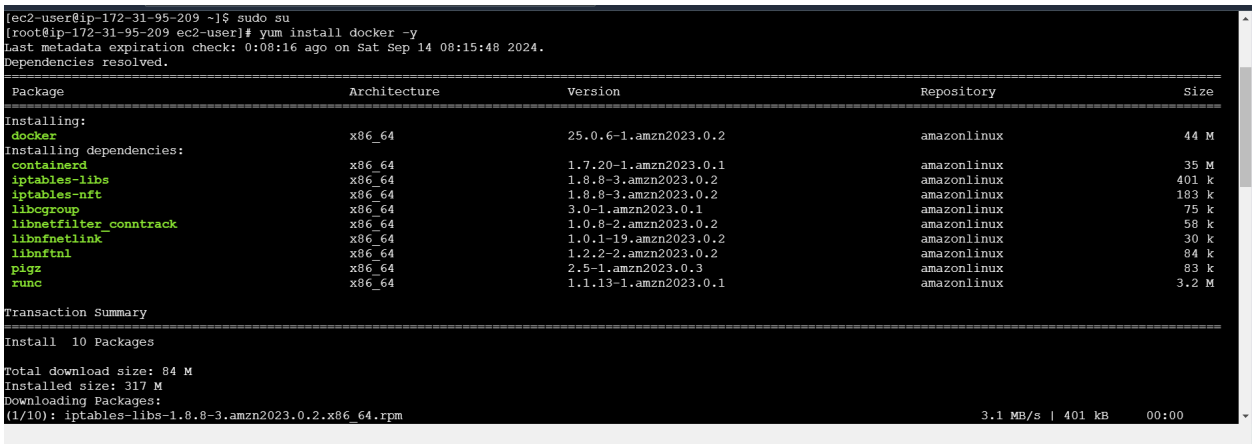
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machine

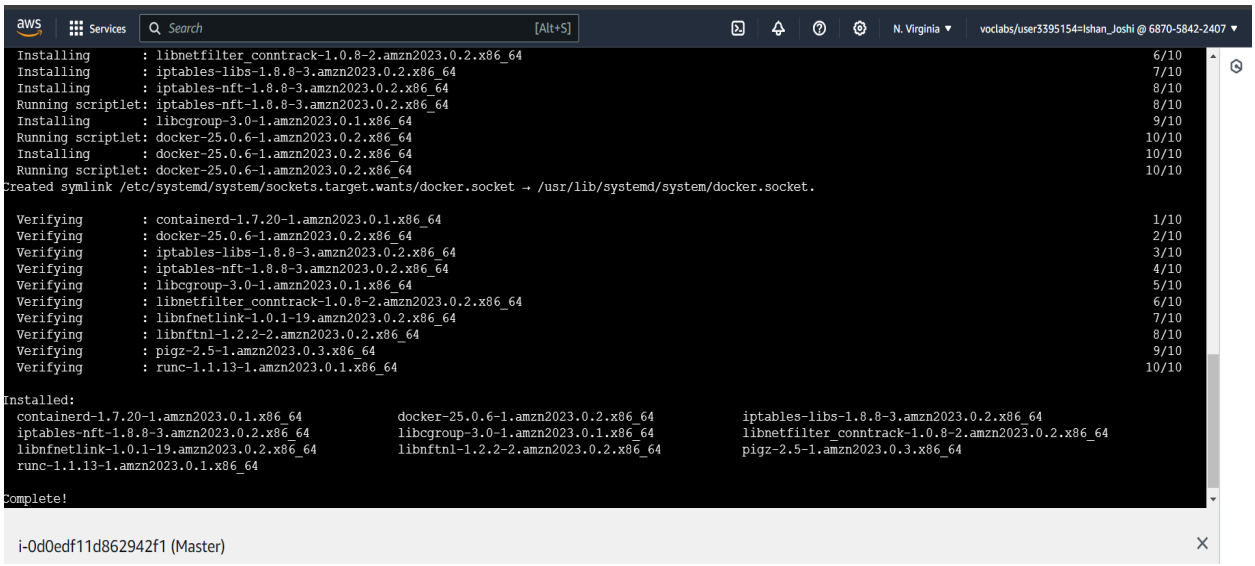
1. Create 3 ec2 instances with an OS as Amazon Linux. Select the instance type as t2.medium.



From now onwards perform the steps on all three instance unless mentioned otherwise.

2. Install docker with the command:
   sudo yum install docker -y





3. Configure cgroup in deamon.json file using the following commands.

    - cd /etc/docker
    - cd /etc/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

```
[ec2-user@ip-172-31-24-202 ~]$ cd /etc/docker
[ec2-user@ip-172-31-24-202 docker]$ cd /etc/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-172-31-24-202 docker]$
```

4. Enable docker
    - sudo systemctl enable docker
    - sudo systemctl daemon-reload
    - sudo systemctl restart docker

```
[ec2-user@ip-172-31-24-202 docker]$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-24-202 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-24-202 docker]$
```

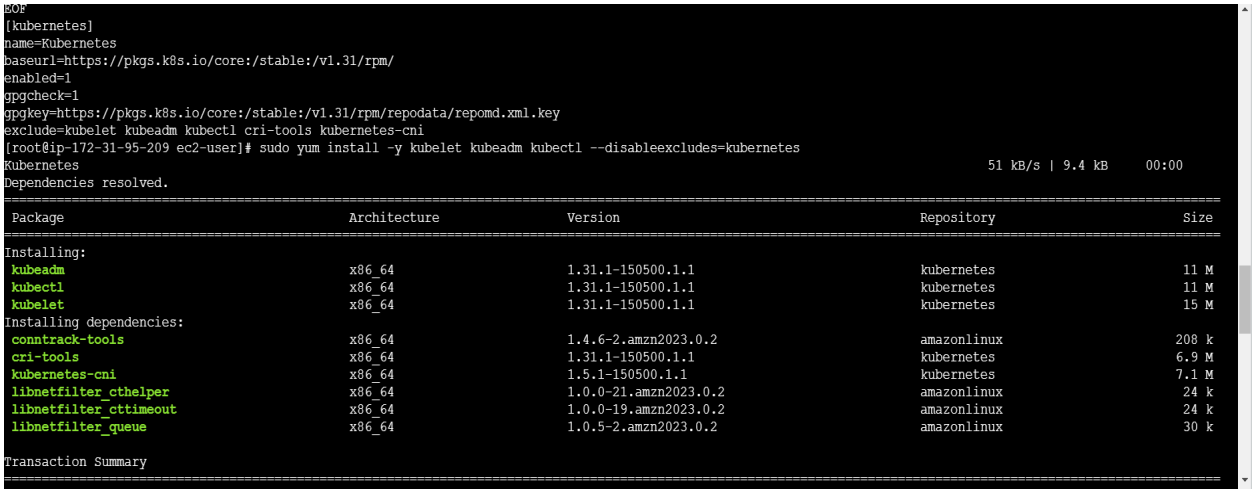
5. Install Kubernetes
    I. Disable SELinux before configuring kubelet
    sudo setenforce 0
    sudo sed -i ‘s/^SELINUX=enforcing$/SELINUX=permissive/’ /etc/selinux/config

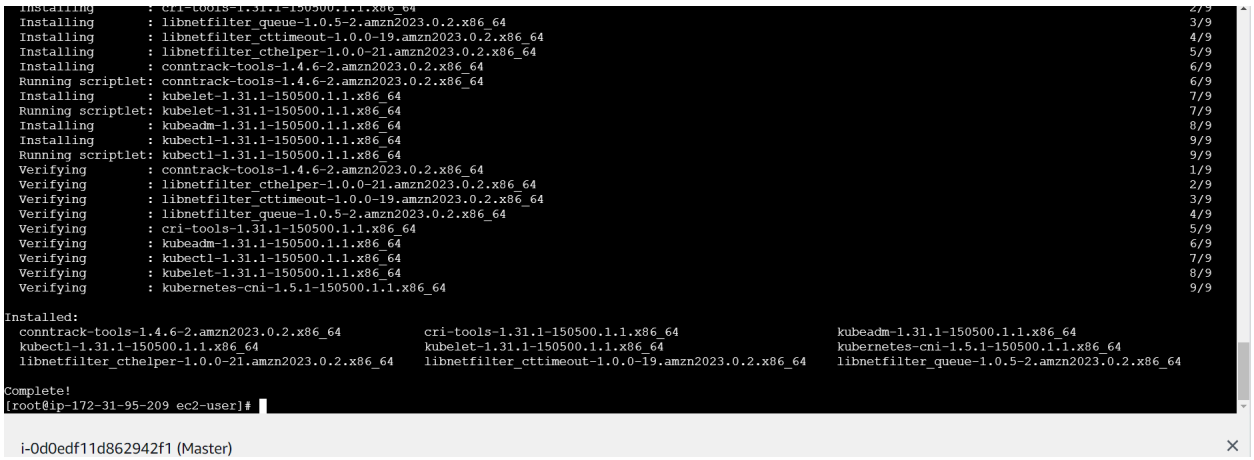
```
[ec2-user@ip-172-31-24-202 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-24-202 docker]$
```

    II. Add kubernetes repository
    cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
    [kubernetes]
    name=Kubernetes

baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF

Run the commands to install kubernetes packages
sudo yum update
sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes





III. 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

```
[root@ip-172-31-91-224 ec2-user]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-91-224 ec2-user]# 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
[root@ip-172-31-91-224 ec2-user]#
```

6. Perform the following only on master machine

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

```
[root@ip-172-31-95-209 ec2-user]# sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all
[init] Using Kubernetes version: v1.31.0
[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
[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'
W0914 08:43:49.642509   28543 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used b
y 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-172-31-95-209.ec2.internal kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.
cluster.local] and IPs [10.96.0.1 172.31.95.209]
[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-172-31-95-209.ec2.internal localhost] and IPs [172.31.95.209 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-95-209.ec2.internal localhost] and IPs [172.31.95.209 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
```

i-0d0edf11d862942f1 (Master)

PublicIPs: 54.211.84.59 PrivateIPs: 172.31.95.209


```
[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 172.31.95.209:6443 --token mj8t91.78gzp3m66y9wtqpl \
        --discovery-token-ca-cert-hash sha256:17924b14b29026a66897f3dc374726434c7dcb1e6b26b62c18d8d74b1f5f48c2
[root@ip-172-31-95-209 ec2-user]#
```

   Save the join command in notepad

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

kubeadm join 172.31.95.209:6443 --token mj8t91.78gzp3m66y9wtqpl \
        --discovery-token-ca-cert-hash sha256:17924b14b29026a66897f3dc374726434c7dcb1e6b26b62c18d8d74b1f5f48c2
[root@ip-172-31-95-209 ec2-user]#
```

Copy the commands given and run it.

```
[root@ip-172-31-95-209 ec2-user]# mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown $(id -u):$(id -g) $HOME/.kube/config
cp: target '/root/.kube/config' is not a directory
[root@ip-172-31-95-209 ec2-user]# mkdir -p $HOME/.kube
[root@ip-172-31-95-209 ec2-user]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@ip-172-31-95-209 ec2-user]# sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Add the networking plugin (Flammel file) using the following command.

kubectl apply -f
https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml

```
[root@ip-172-31-95-209 ec2-user]# 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
[root@ip-172-31-95-209 ec2-user]#
```

i-0d0edf11d862942f1 (Master)

PublicIPs: 54.211.84.59 PrivateIPs: 172.31.95.209

7. Perform the following commands in worker node only.
   sudo yum install iproute-tc-y
   sudo systemctl enable kubelet
   sudo systemctl restart kubelet

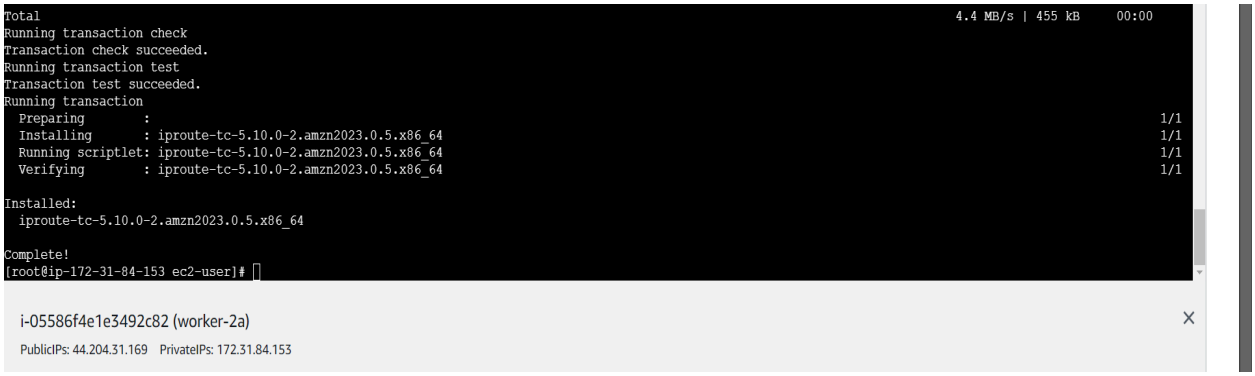
```
[root@ip-172-31-84-153 ec2-user]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-84-153 ec2-user]# sudo yum install iproute-tc -y
sudo systemctl enable kubelet
sudo systemctl restart kubelet
Last metadata expiration check: 0:15:43 ago on Sat Sep 14 08:40:20 2024.
Dependencies resolved.
================================================================================
 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          7.6 MB/s | 455 kB     00:00
--------------------------------------------------------------------------------
Total                                                4.4 MB/s | 455 kB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
```

i-05586f4e1e3492c82 (worker-2a)

PublicIPs: 44.204.31.169 PrivateIPs: 172.31.84.153



Run the join command saved in notepad previously.**(ERROR)**

The following output indicates that the command was successfully run, but its execution is not getting completed. This is due to failure of the API server.

Conclusion:

In the experiment, a Kubernetes cluster was set up using three Amazon Linux EC2 instances, with Kubernetes components installed on each one. The master node was initialized using kubeadm, and the Flannel network plugin was applied to enable pod networking. The worker nodes were connected to the cluster by running the join command generated during the initialization process. While the steps to establish a Kubernetes cluster on EC2 instances were successfully understood, there was a noticeable delay in the worker nodes joining the master node, likely due to potential network connectivity problems or configuration issues.