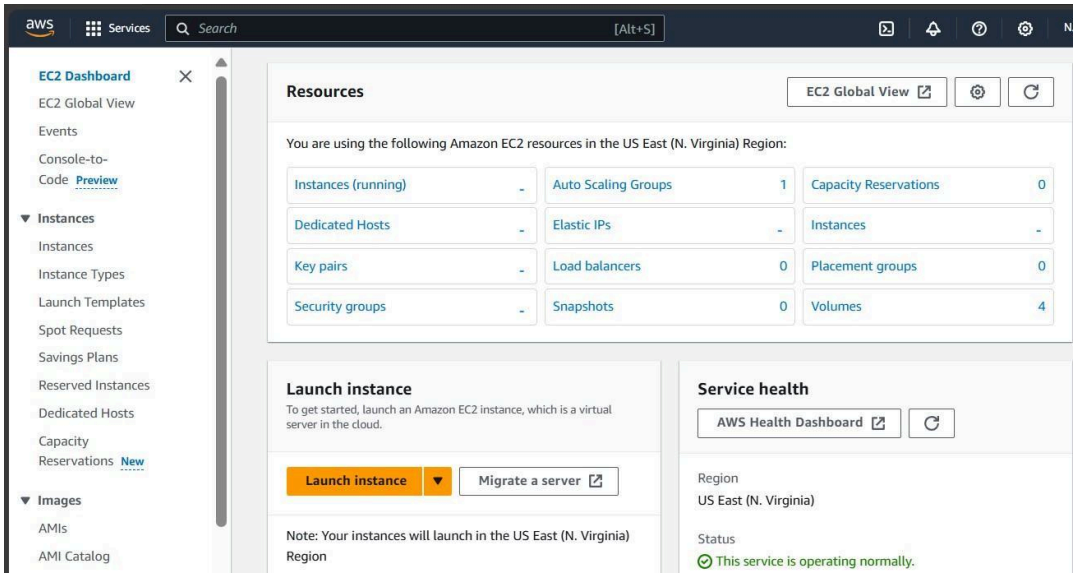
EXPERIMENT NO. 3

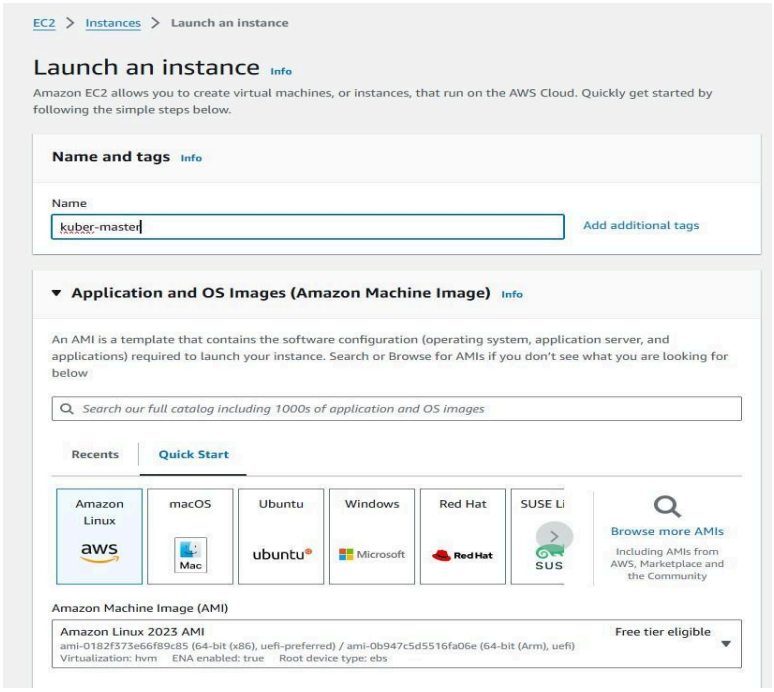
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud.

Procedure:

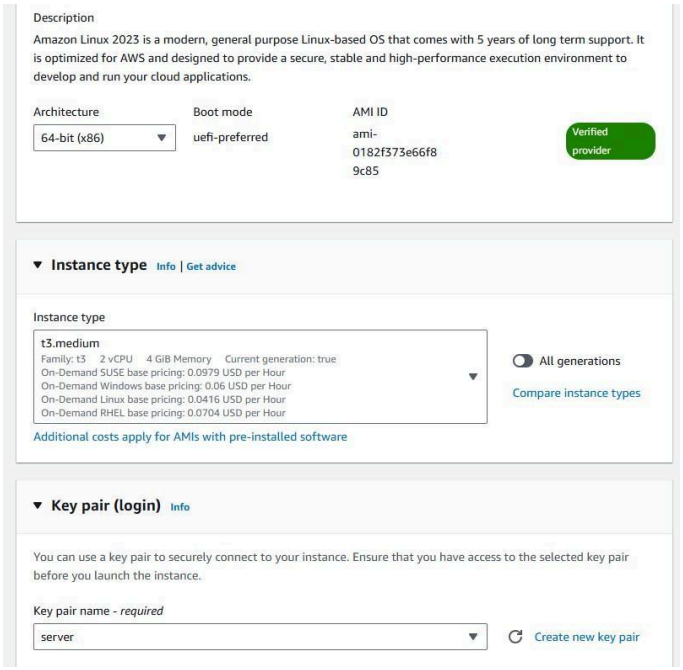
1. **Creation Of Instance**

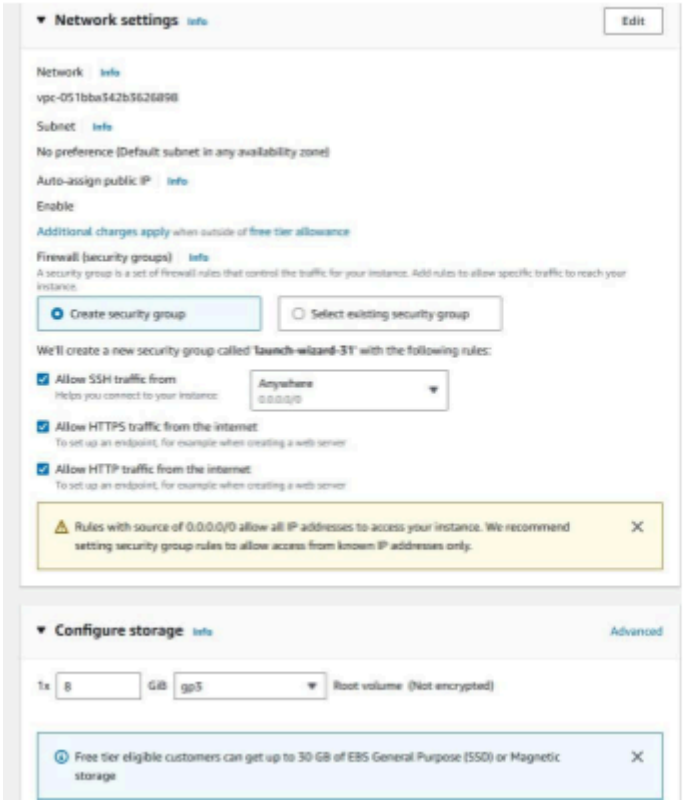


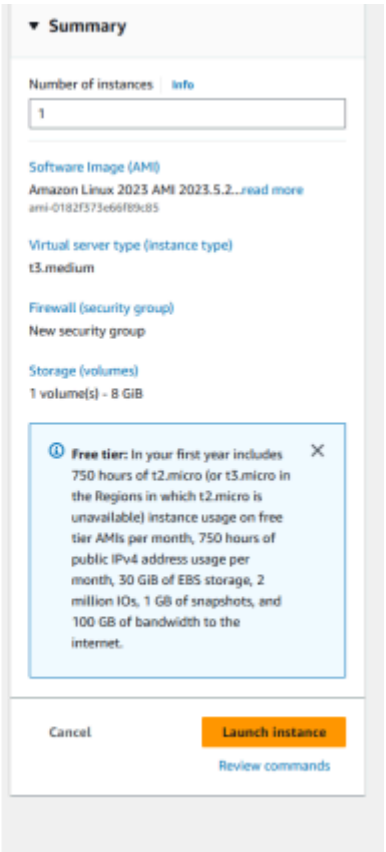
Search EC-2 instance. Then create three EC-2 instances and choose Amazon Linux as OS and also allow ssh traffic from anywhere.



NAME: KAUSHAL S GALAV D15C 11 EXP 3







To efficiently run kubernetes cluster select instance type of at least t3.medium as kubernetes recommends at least 2 vCPU to run smoothly on it.

Name	Instance ID	Instance state
kube-master	i-00aa79ac09d7462c0	Running
kube-worker1	i-0bab86cd3fbfcb40a	Running
kube-worker2	i-00dcfd302ffd80dda	Running

- Then for making connection through SSH into all 3 machines each in separate terminal Use this following command:
  ssh -i <keyname>.pem ubuntu@<public_ip_address> where keyname is name of the key you created here i created key server.pem and use public IP address.(I have entered this command on git bash where i entered in downloads where server.pem is stored then as the key is not accessible hence we need to change its mode using chmod 400 “key name.pem”. Then use the given command for making connections).

```
Sadneya@DESKTOP-IEPNL3D MINGW64 ~ (master)
$ cd downloads

Sadneya@DESKTOP-IEPNL3D MINGW64 ~/downloads (master)
$ chmod 400 "server.pem"

Sadneya@DESKTOP-IEPNL3D MINGW64 ~/downloads (master)
$ ssh -i "server.pem" ec2-user@ec2-54-174-206-93.compute-1.amazonaws.com
The authenticity of host 'ec2-54-174-206-93.compute-1.amazonaws.com (54.174.206.
93)' can't be established.
ED25519 key fingerprint is SHA256:T+tsGyI15gAvUvjeAZ7GjDIWXHOaI4EPF5g5oICrkoQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-174-206-93.compute-1.amazonaws.com' (ED25519)
 to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
```

2. Installation Of Docker on three machines

- For installation of Docker into all three machines run the following command:
  sudo yum install docker -y

```
[ec2-user@ip-172-31-87-114 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:06:20 ago on Fri Sep 13 03:20:22 2024.
Dependencies resolved.
================================================================================
 Package                  Arch     Version                  Repository     Size
================================================================================
Installing:
 docker                   x86_64   25.0.6-1.amzn2023.0.2    amazonlinux    44 M
Installing dependencies:
 containerd               x86_64   1.7.20-1.amzn2023.0.1    amazonlinux    35 M
 iptables-libs            x86_64   1.8.8-3.amzn2023.0.2     amazonlinux   401 k
 iptables-nft             x86_64   1.8.8-3.amzn2023.0.2     amazonlinux   183 k
 libcgroup                x86_64   3.0-1.amzn2023.0.1       amazonlinux    75 k
 libnetfilter_conntrack   x86_64   1.0.8-2.amzn2023.0.2     amazonlinux    58 k
 libnfnetlink             x86_64   1.0.1-19.amzn2023.0.2    amazonlinux    30 k
 libnftnl                 x86_64   1.2.2-2.amzn2023.0.2     amazonlinux    84 k
 pigz                     x86_64   2.5-1.amzn2023.0.3       amazonlinux    83 k
 runc                     x86_64   1.1.13-1.amzn2023.0.1    amazonlinux   3.2 M

Transaction Summary
================================================================================
Install  10 Packages

Total download size: 84 M
Installed size: 317 M
Downloading Packages:
(1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_ 3.6 MB/s | 401 kB     00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_6 4.6 MB/s | 183 kB     00:00
```

```
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
  libcgroup-3.0-1.amzn2023.0.1.x86_64
  libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
  libnfnetlink-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!
```

- Then, configure cgroup in a daemon.json file by using following commands 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-172-31-87-114 ~]$ cd /etc/docker
[ec2-user@ip-172-31-87-114 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"
}
```

- Then after this run the following command to enable and start docker and also to load the daemon.json file.
  sudo systemctl enable docker
  sudo systemctl daemon-reload
  sudo systemctl restart docker

```
[ec2-user@ip-172-31-80-126 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.
```

- Then check the version of docker installed. docker -v

```
[ec2-user@ip-172-31-80-126 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
```

3. Installation Of Kubernetes on three machines

- SELinux needs to be disable before configuring kubelet thus run the following command
  sudo setenforce 0
  sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

```
[ec2-user@ip-172-31-80-126 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

- Here We are adding kubernetes using the repository whose command is given below. 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-172-31-80-126 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
```

- After that Run following command to make the updation and also to install kubelet ,kubeadm, kubectl: sudo yum update

```
[ec2-user@ip-172-31-80-126 docker]$ sudo yum update
Kubernetes
Dependencies resolved.
Nothing to do.
Complete!
```

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

```
[ec2-user@ip-172-31-80-126 docker]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:00:10 ago on Fri Sep 13 10:31:17 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                                                   64 MB/s |  53 MB     00:00
Kubernetes                                              20 kB/s | 1.7 kB     00:00
Importing GPG key 0x9A296436:
 Userid     : "isv:kubernetes OBS Project <isv:kubernetes@build.opensuse.org>"
 Fingerprint: DE15 B144 86CD 377B 9E87 6E1A 2346 54DA 9A29 6436
 From       : https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : kubernetes-cni-1.4.0-150500.1.1.x86_64                 1/9
  Installing       : cri-tools-1.30.1-150500.1.1.x86_64                     2/9
  Installing       : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64         3/9
  Installing       : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64    4/9
  Installing       : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64     5/9
  Installing       : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64            6/9
  Running scriptlet: conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64            6/9
  Installing       : kubelet-1.30.5-150500.1.1.x86_64                       7/9
  Running scriptlet: kubelet-1.30.5-150500.1.1.x86_64                       7/9
  Installing       : kubeadm-1.30.5-150500.1.1.x86_64                       8/9
  Installing       : kubectl-1.30.5-150500.1.1.x86_64                       9/9
  Running scriptlet: kubectl-1.30.5-150500.1.1.x86_64                       9/9
  Verifying        : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64            1/9
  Verifying        : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64     2/9
  Verifying        : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64    3/9
  Verifying        : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64         4/9
  Verifying        : cri-tools-1.30.1-150500.1.1.x86_64                     5/9
  Verifying        : kubeadm-1.30.5-150500.1.1.x86_64                       6/9
  Verifying        : kubectl-1.30.5-150500.1.1.x86_64                       7/9
  Verifying        : kubelet-1.30.5-150500.1.1.x86_64                       8/9
  Verifying        : kubernetes-cni-1.4.0-150500.1.1.x86_64                 9/9

Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64        cri-tools-1.30.1-150500.1.1.x86_64                    kubeadm-1.30.5-150500.1.1.x86_64
  kubectl-1.30.5-150500.1.1.x86_64                   kubelet-1.30.5-150500.1.1.x86_64                      kubernetes-cni-1.4.0-150500.1.1.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64 libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64   libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64

Complete!
```

- After installing Kubernetes, we need to configure internet options to allow bridging.
    1. sudo swapoff -a
    2. echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
    3. sudo sysctl -p

```
[ec2-user@ip-172-31-80-126 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
```

4. **Perform this ONLY on the Master machine**

- Initialize kubernetes by typing below command

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

```
[ec2-user@ip-172-31-80-126 docker]$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
I0913 10:32:44.629146   26680 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
```

```
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 172.31.80.126:6443 --token jhtgwo.4qv2vtxrcf6nvgpk \
        --discovery-token-ca-cert-hash sha256:766e48546942419274bcd18c370d2492f6e49dac9f98890804362194690f0f4a
```

- So after initialization you will get token at the end for joining master and worker. Like here I got this :(save this token as it is required later.Then you can join any number of worker nodes by running the following on each as root.)

  kubeadm join 172.31.80.126:6443 --token jhtgwo.4qv2vtxrcf6nvgpk\
  --discovery-token-ca-cert-hash
  sha256:766e48546942419274bcd18c370d2492f6e49dac9f98890804362194690f0f4a

- Also,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

```
[ec2-user@ip-172-31-80-126 docker]$  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

- Then, add a common networking plugin called flammel file as mentioned in the code.
  kubectl apply -f
  https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

```
[ec2-user@ip-172-31-80-126 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
```

- Now to Check the created pod use this command kubectl get pods

5. **Perform this ONLY on the worker machines**

Paste the below command on all 2 worker machines

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

Now use this
kubeadm join 172.31.80.126:6443 --token jhtgwo.4qv2vtxrcf6nvgpk\
--discovery-token-ca-cert-hash
sha256:766e48546942419274bcd18c370d2492f6e49dac9f98890804362194690f0f4a

(Optional To check the status of pods executed these commands:

Kubectl get pods -n kube-system :gives status of all pods

Kubectl get daemonstat -n kube-system: gives status of pod named daemonstat

```
[ec2-user@ip-172-31-87-114 docker]$ kubectl get pods -n kube-system
NAME                                              READY   STATUS             RESTARTS        AGE
coredns-55cb58b774-fxl2f                          1/1     Running            0               100s
coredns-55cb58b774-xnl4v                          1/1     Running            0               100s
etcd-ip-172-31-87-114.ec2.internal                1/1     Running            1 (2m45s ago)   75s
kube-apiserver-ip-172-31-87-114.ec2.internal      1/1     Running            1 (2m15s ago)   2m11s
kube-controller-manager-ip-172-31-87-114.ec2.internal   0/1     CrashLoopBackOff   1 (8s ago)      70s
kube-proxy-4dv8m                                  1/1     Running            2 (26s ago)     100s
kube-scheduler-ip-172-31-87-114.ec2.internal      1/1     Running            1 (2m45s ago)   76s
[ec2-user@ip-172-31-87-114 docker]$ kubectl get daemonset -n kube-system
NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-proxy   1         1         1       1            1           kubernetes.io/os=linux   3m
```

)

Now to see whether master and workers get connected successfully or not run **kubectl get nodes** command on master machine

```
[ec2-user@ip-172-31-87-114 docker]$ kubectl get nodes
NAME                           STATUS   ROLES           AGE     VERSION
ip-172-31-87-114.ec2.internal   Ready    control-plane   3m21s   v1.30.5
```

Conclusion: In these EC-2 instance created successfully on AWS Linux. Then I installed docker ,kuberneted and then kubelet ,kubeadm, kubectl.Then on Master machine ,I initailized the kubernetes which given me the token which will be used for connection of master and workers.then on slave I installed iproute and enabled and restarted kubelet then i enter the token which i got from master but there was an issue in joint.that is why on output i just got of only one pc mater on performing command kubectl get nodes.