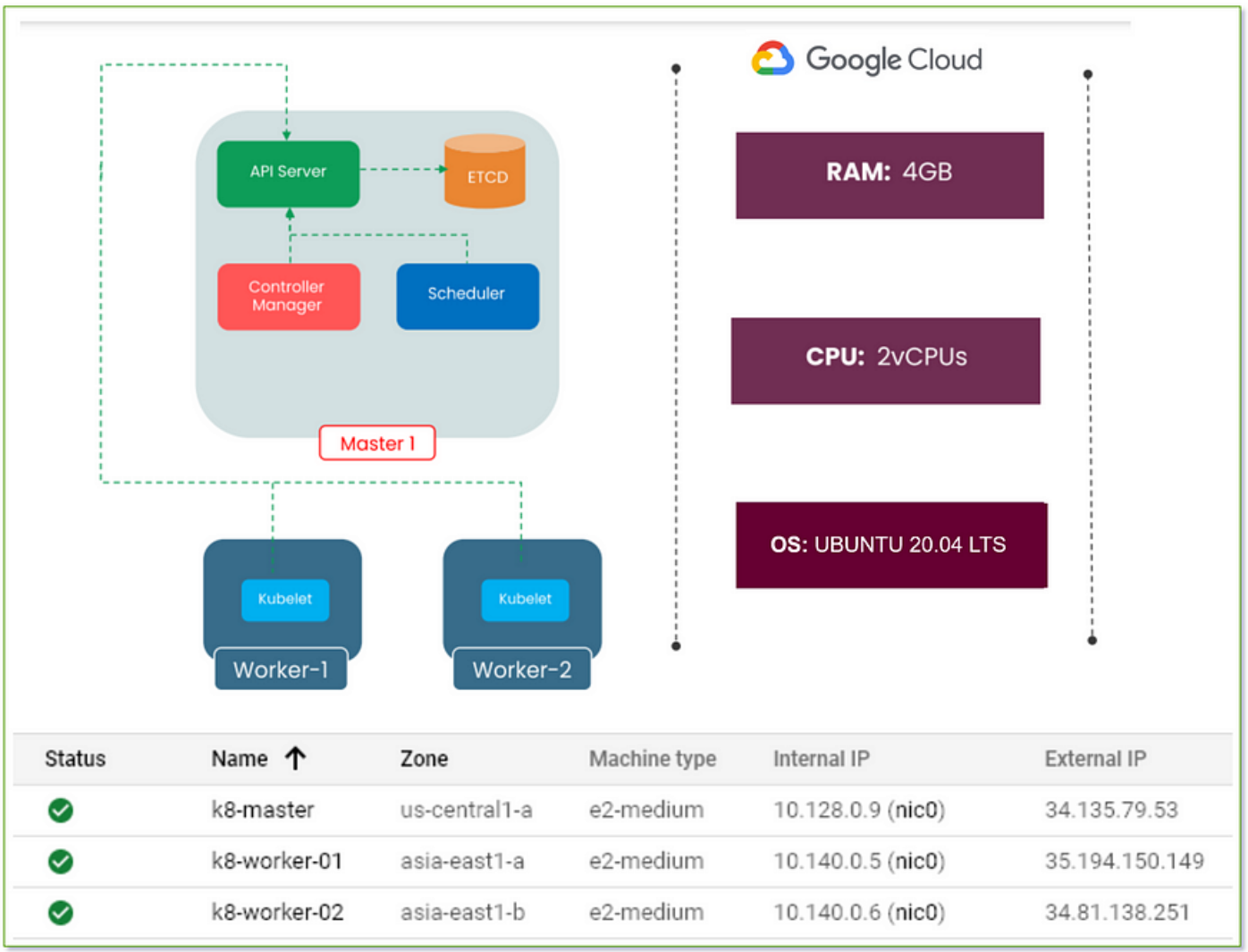
Create a Kubernetes cluster using kubeadm

Prerequisites

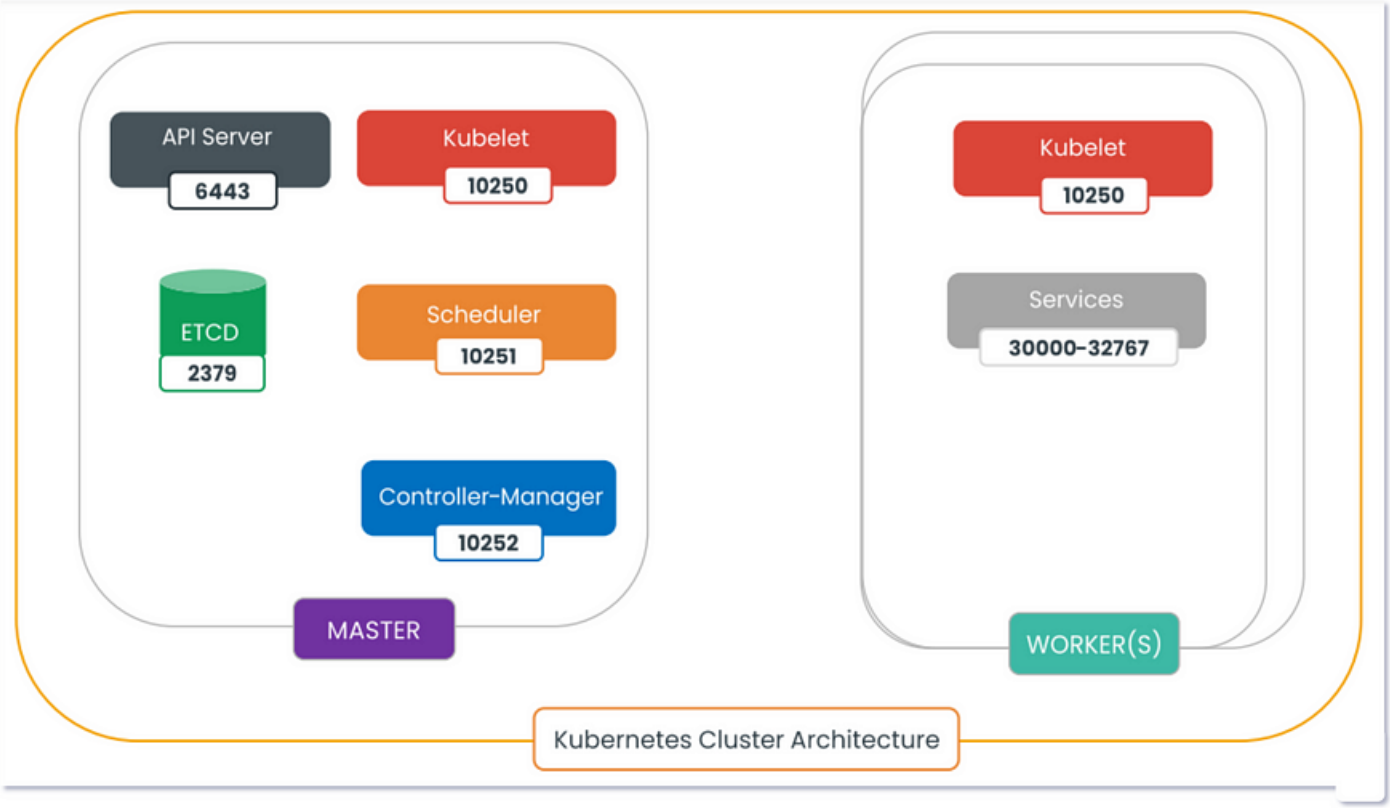
- Three Ubuntu servers 20.04 with at least 4GB RAM and 2 vCPUs each.



Status	Name ↑	Zone	Machine type	Internal IP	External IP
✓	k8-master	us-central1-a	e2-medium	10.128.0.9 (nic0)	34.135.79.53
✓	k8-worker-01	asia-east1-a	e2-medium	10.140.0.5 (nic0)	35.194.150.149
✓	k8-worker-02	asia-east1-b	e2-medium	10.140.0.6 (nic0)	34.81.138.251

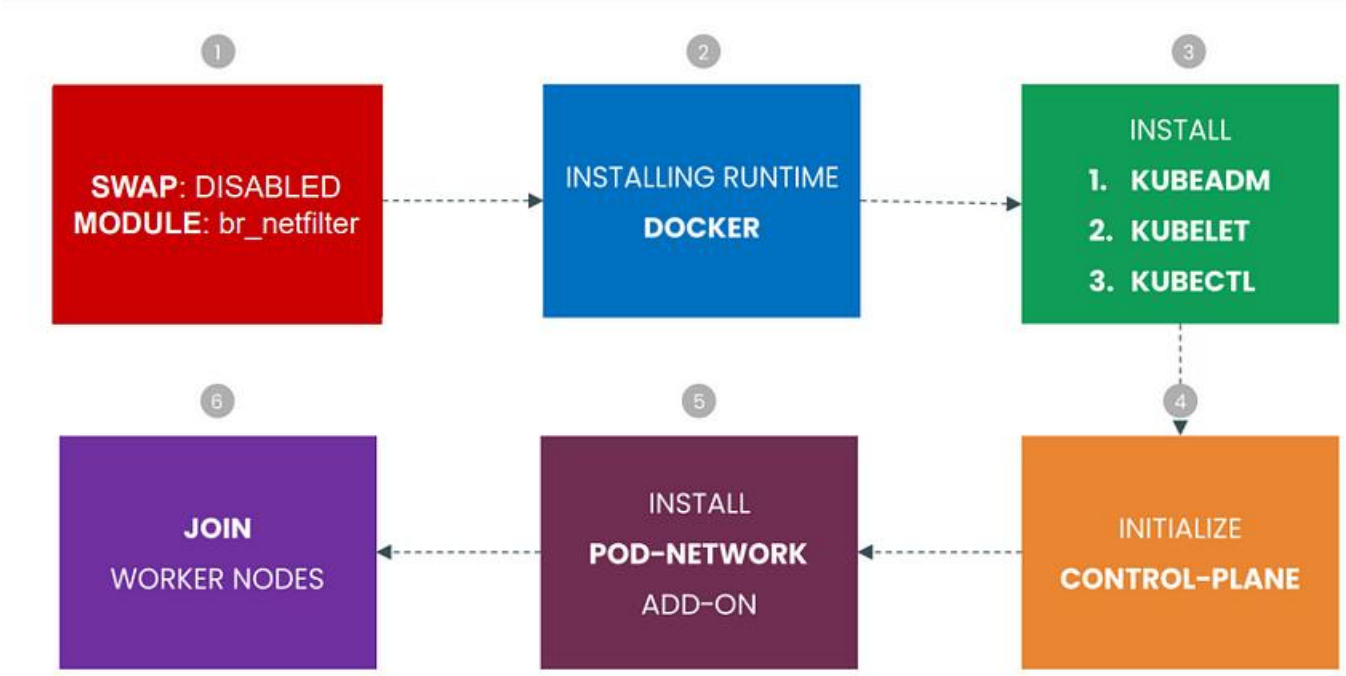
- SSH Access with sudo privileges.

- Firewall Ports/Inbound Traffic Ports should open for Kubernetes Cluster.



- Master Node Ports: 2379,6443,10250,10251,10252
- Worker Node Ports: 10250,30000–32767.
- Default port range for NodePort Services -30000–32767.

Setup Steps:



1. Disable the swap and make sure be a net filter module is installed.
2. we will need to install the container runtime interface ie. docker
3. Install kubeadm, kubelet, and kubectl: **kubeadm** is building tools that help to bootstrap the cluster,
   kubelet is an agent that runs on each node to make sure that containers are running in a Pod,
   kubectl allows you to run commands against Kubernetes clusters.
4. Initialize the Kubernetes cluster which creates certificates, pods, services, and other resources.
5. Installing wave network add-on.
6. Finally, join the worker nodes to the Kubernetes cluster.

Step1) Disable Swap (Run it on MASTER & WORKER Nodes)

```
$ swapoff -a
$ sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

```
root@k8-master:~# swapoff -a
root@k8-master:~# sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
root@k8-master:~#
```

1a) Bridge Traffic

```
$ lsmod | grep br_netfilter
$ sudo modprobe br_netfilter$ cat <<EOF | sudo tee
/etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF$ sudo sysctl --system
```

```
root@k8-master:~# lsmod | grep br_netfilter
root@k8-master:~# sudo modprobe br_netfilter
root@k8-master:~# cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
> net.bridge.bridge-nf-call-ip6tables = 1
> net.bridge.bridge-nf-call-iptables = 1
> EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
root@k8-master:~# sudo sysctl --system
* Applying /etc/sysctl.d/10-console-messages.conf ...
```

- lsmod | grep br_netfilter will load the module.
- To load it explicitly calls sudo modprobe br_netfilter.
- As a requirement for your Linux Node's iptables to correctly see bridged traffic.
- You should ensure net.bridge.bridge-nf-call-iptables is set to 1 in your sysctl configBridge Traffic.

Step2) Install Docker (Run it on MASTER & WORKER Nodes)

```
$ apt-get update
$ apt install docker.io
$ systemctl start docker
```

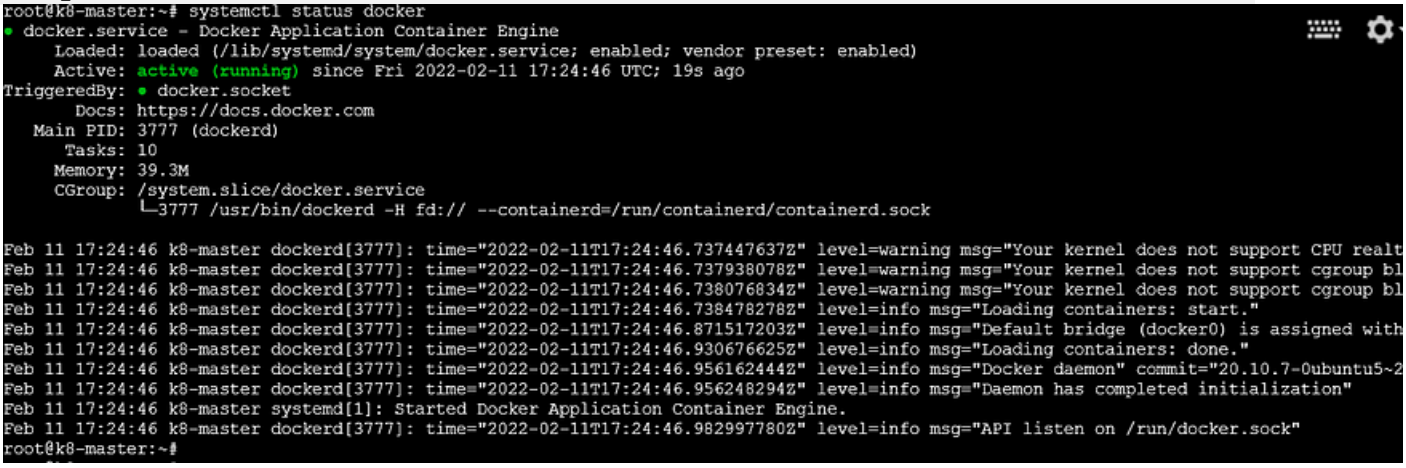
If you facing any issues, Click here to install docker.

2a) Setting up the Docker daemon

```
$ 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
```

2b) Reload, enable and restart the docker service

```
$ systemctl daemon-reload
$ systemctl enable docker
$ systemctl restart docker
$ systemctl status docker
```



Make sure the docker service is running.

Step3) Install kubeadm, kubelet, and kubectl (Run it on MASTER & WORKER Nodes)

```
$ apt-get update && sudo apt-get install -y apt-transport-https
curl
$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg |
sudo apt-key add -
$ cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
```

```
root@k8-master:~# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
OK
root@k8-master:~# cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
> deb https://apt.kubernetes.io/ kubernetes-xenial main
> EOF
deb https://apt.kubernetes.io/ kubernetes-xenial main
root@k8-master:~#
```

3a) Installing Kubeadm, Kubelet, Kubectl:

```
$ apt-get update
$ apt-get install -y kubelet kubeadm kubectl
$ apt-mark hold kubelet kubeadm kubectl
```

```
Setting up kubernetes-cni (0.8.7-00) ...
Setting up kubelet (1.23.3-00) ...
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /lib/systemd/system/kubelet.service.
Setting up kubeadm (1.23.3-00) ...
Processing triggers for man-db (2.9.1-1) ...
root@k8-master:~#
root@k8-master:~# apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@k8-master:~#
```

3b) Start and enable Kubelet

```
$ systemctl daemon-reload
$ systemctl enable kubelet
$ systemctl restart kubelet
$ systemctl status kubelet
```

```
root@k8-control-plane:~# systemctl enable kubelet
root@k8-control-plane:~# systemctl restart kubelet
root@k8-control-plane:~# systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
     Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
    Drop-In: /etc/systemd/system/kubelet.service.d
             └─10-kubeadm.conf
     Active: active (running) since Tue 2022-02-08 18:56:02 UTC; 14ms ago
       Docs: https://kubernetes.io/docs/home/
   Main PID: 7771 (kubelet)
      Tasks: 6 (limit: 4698)
     Memory: 1.9M
     CGroup: /system.slice/kubelet.service
             └─7771 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/

Feb 08 18:56:02 k8-control-plane systemd[1]: Started kubelet: The Kubernetes Node Agent.
```

Step4) Initializing CONTROL-PLANE (Run it on MASTER Node only)

```
$ kubeadm init --pod-network-cidr 10.0.0.0/16
```

```
root@k8-master:~# kubeadm init --pod-network-cidr 10.0.0.0/16
[init] Using Kubernetes version: v1.23.3
[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 in beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[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 10.128.0.9:6443 --token 21dg74.rkaqfcksuut150xm \
        --discovery-token-ca-cert-hash sha256:69cffbab4446f21047711bd074074747daa4211508c973931c0c7f177db4f108
root@k8-master:~#
```

As the above output mentioned copy the token in your notepad, we will need to join worker/slave to the master node.

4a) Create new '.kube' configuration directory and copy the configuration 'admin.conf' from '/etc/kubernetes' directory.

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

```
root@k8-master:~# mkdir -p $HOME/.kube
root@k8-master:~#   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@k8-master:~#   sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Step5) Installing POD-NETWORK add-on (Run it on MASTER Node only)

```
$ kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
```

```
root@k8-master:~# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
root@k8-master:~#
```

Step6) Next Join two worker nodes to master (Run it on both worker nodes)

Paste the Join command from the above kubeadm init output

```
$kubeadm join 10.128.0.9:6443 --token 21dg74.rkaqfcksuut150xm \
>         --discovery-token-ca-cert-hash
sha256:69cffbab4446f21047711bd074074747daa4211508c973931c0c7f177db4
f108
```

```
root@k8-worker-02:~# kubeadm join 10.128.0.9:6443 --token 21dg74.rkaqfcksuut150xm \
>         --discovery-token-ca-cert-hash sha256:69cffbab4446f21047711bd074074747daa4211508
c973931c0c7f177db4f108
[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 kube
adm-config -o yaml'
W0211 17:46:12.146571   17185 utils.go:69] The recommended value for "resolvConf" in "Kube
letConfiguration" is: /run/systemd/resolve/resolv.conf; the provided value is: /run/system
d/resolve/resolv.conf
[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/kube
adm-flags.env"
[kubelet-start] Starting the kubelet
[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.
```

```
root@k8-worker-01:~# kubeadm join 10.128.0.9:6443 --token 21dg74.rka
qfcksuut150xm \
>         --discovery-token-ca-cert-hash sha256:69cffbab4446f2104771
1bd074074747daa4211508c973931c0c7f177db4f108
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n k
ube-system get cm kubeadm-config -o yaml'
W0211 17:39:11.658514   17122 utils.go:69] The recommended value for
 "resolvConf" in "KubeletConfiguration" is: /run/systemd/resolve/res
olv.conf; the provided value is: /run/systemd/resolve/resolv.conf
[kubelet-start] Writing kubelet configuration to file "/var/lib/kube
let/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[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 w
as received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join t
he cluster.

root@k8-worker-01:~#
```

6a) Run this command IF you do not have the above join command.

```
$ kubeadm token create — print-join-command
```

6b) Check the joined nodes

```
$ kubectl get nodes -o wide
```

```
root@k8-master:~# kubectl get nodes -o wide
NAME          STATUS   ROLES                  AGE     VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION     CONTAINER-RUNTIME
k8-master     Ready    control-plane,master   18m     v1.23.3   10.128.0.9    <none>        Ubuntu 20.04.3 LTS   5.11.0-1029-gcp    docker://20.10.7
k8-worker-01  Ready    <none>                 8m24s   v1.23.3   10.140.0.5    <none>        Ubuntu 20.04.3 LTS   5.11.0-1029-gcp    docker://20.10.7
k8-worker-02  Ready    <none>                 83s     v1.23.3   10.140.0.6    <none>        Ubuntu 20.04.3 LTS   5.11.0-1029-gcp    docker://20.10.7
root@k8-master:~#
```

Also, check