Two Methods

1. Kubernetes Setup using Kubeadm

~Start - Execute the below commands in both Master/worker nodes

Login to both instances execute the below commands:

```
sudo apt-get update -y  && sudo apt-get install apt-transport-https -y
```

Change to root user

```
sudo su -
sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -

cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF

sudo apt-get update
```

#Disable swap memory for better performance

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

Enable IP tables

#We need to enable IT tables for pod to pod communication.

```
modprobe br_netfilter
sysctl -p
sudo sysctl net.bridge.bridge-nf-call-iptables=1
```

Install Docker on both Master and Worker nodes

```
apt-get install docker.io -y
```

Add ubuntu user to Docker group

```
usermod -aG docker ubuntu
systemctl restart docker
systemctl enable docker.service
```

Type exit to come out of root user.

Install Kubernetes Modules

```
sudo apt-get install -y kubelet kubeadm kubectl kubernetes-cni

sudo systemctl daemon-reload
sudo systemctl start kubelet
sudo systemctl enable kubelet.service
```

sudo systemctl status docker

#End - Execute the above commands in both Master/worker nodes##########

```
cd /etc/docker/
vi daemon.json
```

```
add this below commands:-
{
   "exec-opts": ["native.cgroupdriver=systemd"]
}
```

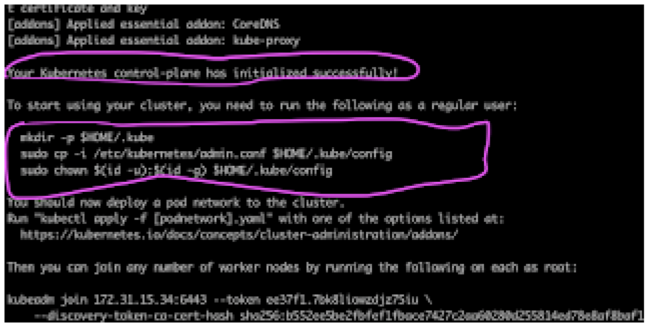
sudo systemctl daemon-reload
sudo systemctl restart docker
sudo systemctl restart kubelet

Wait for Sometime, It will take some time

Initialize Kubeadm on Master Node(only on Master Node)

#Execute the below command as root user to initialize Kubernetes Master node.

```
sudo su -
kubeadm init
```



```
t 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

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 172.31.15.34:6443 --token ee37f1.7bk8liowzdjz75iu \
    --discovery-token-ca-cert-hash sha256:b552ee5be2fbfef1fbace7427c2aa60280d255814ed78e8af8baf1
```

Make sure you see the above message to confirm master node is up.

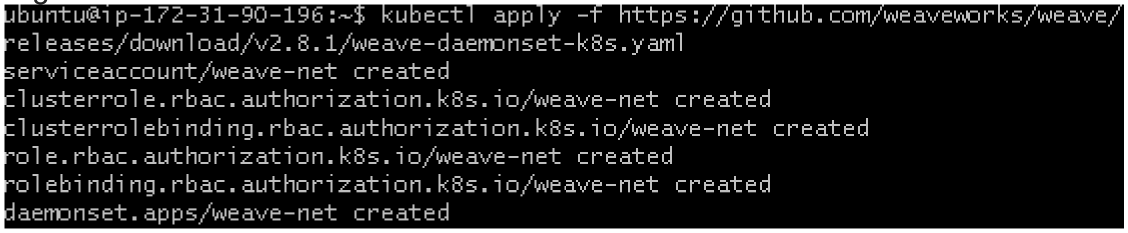
#Now type exit to exit from root user and execute below commands as normal user

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

Installing the Weave Net Add-On

kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml

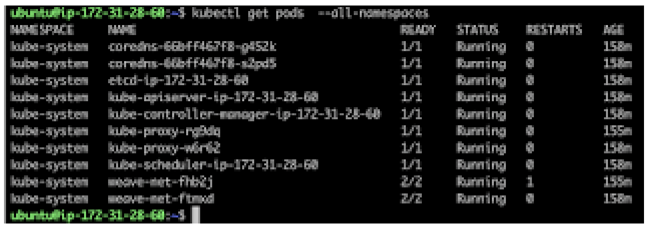
It make take a few mins to execute the above command and show show the below message.



```
ubuntu@ip-172-31-90-196:~$ kubectl apply -f https://github.com/weaveworks/weave/
releases/download/v2.8.1/weave-daemonset-k8s.yaml
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
```

Now execute the below command to see the pods.

kubectl get pods --all-namespaces



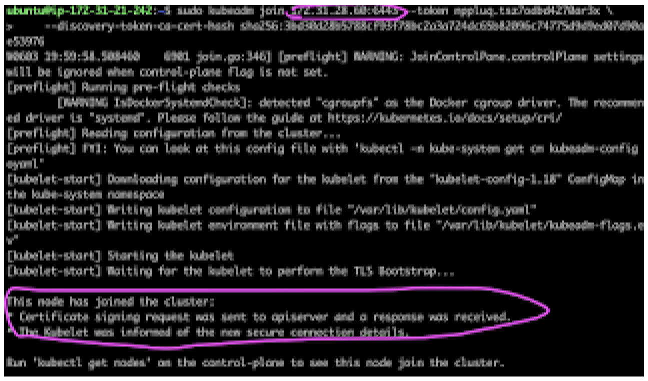
```
ubuntu@ip-172-31-28-60:~$ kubectl get pods  --all-namespaces
NAMESPACE     NAME                                      READY   STATUS    RESTARTS   AGE
kube-system   coredns-66bff467f8-g452k                  1/1     Running   0          158m
kube-system   coredns-66bff467f8-s2pd5                  1/1     Running   0          158m
kube-system   etcd-ip-172-31-28-60                      1/1     Running   0          158m
kube-system   kube-apiserver-ip-172-31-28-60            1/1     Running   0          158m
kube-system   kube-controller-manager-ip-172-31-28-60   1/1     Running   0          158m
kube-system   kube-proxy-rg9dq                          1/1     Running   0          155m
kube-system   kube-proxy-w6r62                          1/1     Running   0          158m
kube-system   kube-scheduler-ip-172-31-28-60            1/1     Running   0          158m
kube-system   weave-net-fhb2j                           2/2     Running   1          155m
kube-system   weave-net-ftmxd                           2/2     Running   0          158m
ubuntu@ip-172-31-28-60:~$
```

Now login to Worker Node

Join worker node to Master Node

The below command will join worker node to master node, execute this a normal user by putting sudo before:

sudo kubeadm join <master_node_ip>:6443 --token xrvked.s0n9771cd9x8a9oc \
    --discovery-token-ca-cert-hash sha256:288084720b5aad132787665cb73b9c530763cd1cba10e12574b4e97452137b4a

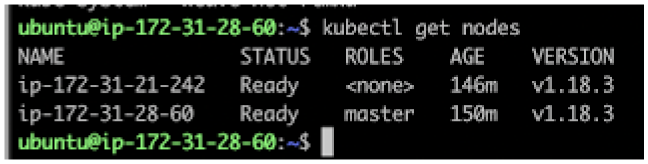


```
ubuntu@ip-172-31-21-242:~$ sudo kubeadm join 172.31.28.60:6443 --token mppluq.tsz7odbd4270ar3x \
>     --discovery-token-ca-cert-hash sha256:3b03802d05788cf93f78bc2a3a724dc65b82096c74775d9d9ed07d90a
e53976
W0603 19:59:58.508460    6981 join.go:346] [preflight] WARNING: JoinControlPane.controlPlane settings
will be ignored when control-plane flag is not set.
[preflight] Running pre-flight checks
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommen
ed driver is "systemd". Please follow the guide at https://kubernetes.io/docs/setup/cri/
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config
-oyaml'
[kubelet-start] Downloading configuration for the kubelet from the "kubelet-config-1.18" ConfigMap in
the kube-system namespace
[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/kubeadm-flags.e
nv"
[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.
```

Go to Master and type the below command

kubectl get nodes

the above command should display both Master and worker nodes.



```
ubuntu@ip-172-31-28-60:~$ kubectl get nodes
NAME              STATUS   ROLES    AGE    VERSION
ip-172-31-21-242  Ready    <none>   146m   v1.18.3
ip-172-31-28-60   Ready    master   150m   v1.18.3
ubuntu@ip-172-31-28-60:~$
```

It means Kubernetes Cluster - both Master and worker nodes are setup successfully and up and running!!!

(OR)

2. How to set up Kubernetes master-slave architecture?

Today, we will set up a complete **Kubernetes master-slave architecture using kubeadm.** According to the kubeadm source, Kubeadm is a tool built to provide kubeadm init and kubeadm join as best-practice “fast paths” for creating Kubernetes clusters.

We will consider building a Kubernetes setup with one master node and 2 worker nodes.

Here we go,

Let us assume we have three Ubuntu Linux machines named kmaster and knode

1. Installing Docker as the container runtime Interface

On all the machines do the following:

```
#update the repository
sudo apt-get update

#Install docker
sudo apt install docker.io

#Start and automate docker to start at run time
sudo systemctl start docker
sudo systemctl enable docker

#verify docker installation
docker container ls
```

Kubeadm will by default use **docker** as the container runtime interface. In case a machine has both docker and other container runtimes like **contained**, docker takes precedence.

2. Installing kubeadm tool

```
#add the required repository for kubeadm
sudo 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

#update the repository
$ sudo apt-get update

#installing kubelet, kubeadm and kubectl
sudo apt-get install -y kubelet kubeadm kubectl

#setting apt-mark
sudo apt-mark hold kubelet kubeadm kubectl
```

apt-mark will change whether a package has been marked as being automatically installed. Hold is used to mark a package as held back, which will prevent the package from being automatically installed, upgraded, or removed.

Restart the kubelet if required

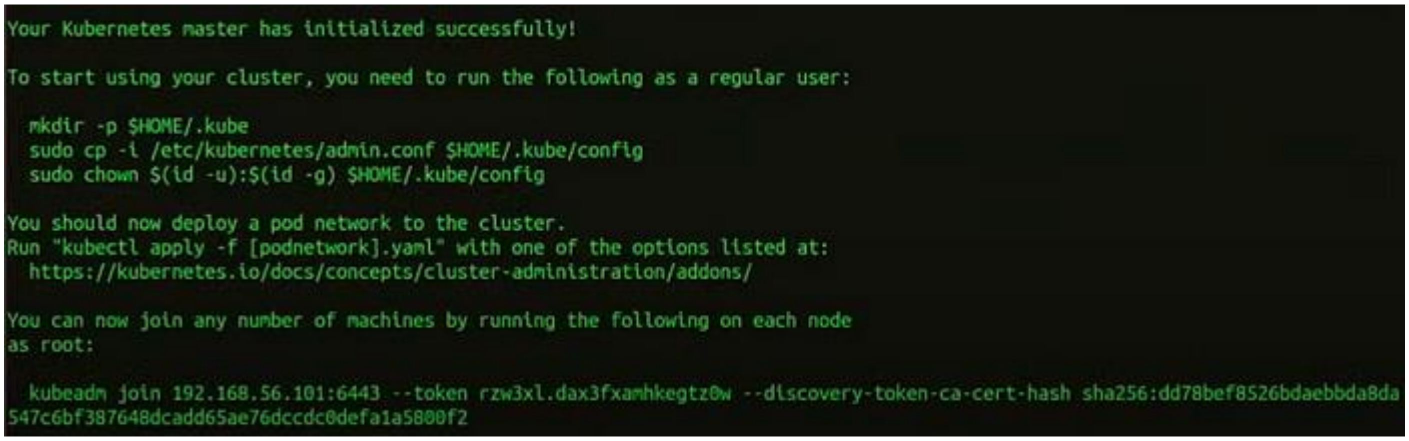
```
systemctl daemon-reload
systemctl restart kubelet
```

3. Initializing the control plane or making the node as master(on master node)

`kubeadm init` will initialize this machine to make it a master.

Kubernetes assigns each node a range of IP addresses, a CIDR(Classless Inter-Domain Routing) block so that each Pod can have a unique IP address. We will specify the private CIDR for the pods to be created.

```
kubeadm init — apiserver-advertise-address=192.168.56.101 --pod-network-
cidr=192.168.0.0/16
```



```
Your Kubernetes master 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

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/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join 192.168.56.101:6443 --token rzw3xl.dax3fxamhkegtz0w --discovery-token-ca-cert-hash sha256:dd78bef8526bdaebbda8da
547c6bf387648dcadd65ae76dccdc0defa1a5800f2
```

Now as seen in the output above, we need to run the below commands as a normal user to use the kubectl from terminal.

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

Now the machine is initialized as master.

4. Joining Cluster

To join Kubernetes cluster simply copy that token at the end and paste it in worker node’s terminal in super user mode.

```
#something like this
kubeadm join <control-plane-host>:<control-plane-port> --token <token> --
discovery-token-ca-cert-hash sha256:<hash>
```

That’s it!

To see all running pods,

```
kubectl get pods -o wide --all-namespaces
```

You would see all pods are running except DNS one. To resolve that problem, enable C**alico network.**

```
# on master node
kubectl apply -f
https://docs.projectcalico.org/v3.14/manifests/calico.yaml
```

Elaboration on Calico related issue:

You must deploy a Container Network Interface (CNI) based Pod network add-on so that your Pods can communicate with each other. Cluster DNS (CoreDNS) will not start up before a network is installed. We will use Calico as our CNI tool. Calico is a networking and network policy provider. Calico supports a flexible set of networking options so you can choose the most efficient option for your situation.