Create K8s Cluster

Objective

In this checkpoint, you will create a kubernetes cluster (1 master , 2 workers) based on kubeadm.

Instructions

1. Create kubernetes cluster :
    - Using vagrant + shell : source1 , source2.
    - Using vagrant + ansible: source

==

Install Vagrant on Ubuntu 18.04

- sudo apt install virtualbox
- sudo apt update
- sudo apt install vagrant
- vagrant --version : Vagrant 2.0.2

Deploying Vagrant On Ubuntu 18.04

- sudo mkdir ~/vagrant-ubuntu
- cd ~/vagrant-ubuntu

Create **Vagrantfile** and configure 3 VMS:(in our case, we choose Ubuntu Linux 20.04 64bit as OS

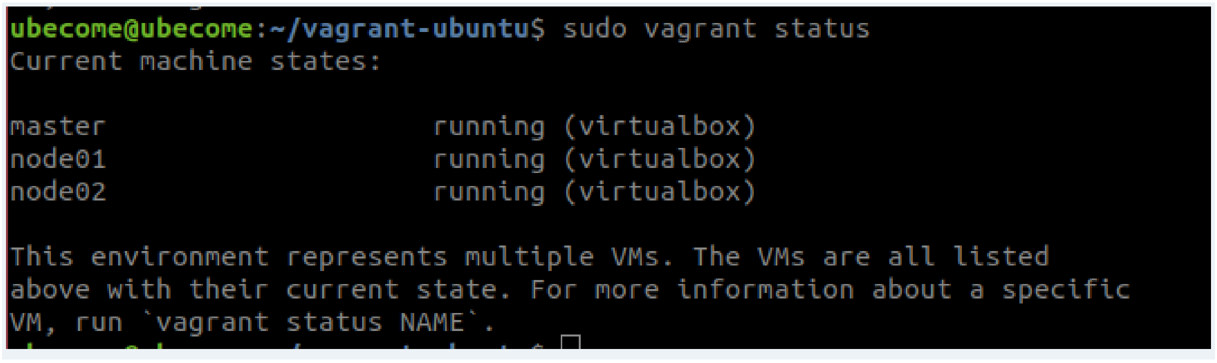
- Master
- node01
- node02

`Start Création :`

sudo vagrant up

`Lister les VMS status :`

- sudo vagrant status



```
ubecome@ubecome:~/vagrant-ubuntu$ sudo vagrant status
Current machine states:

master                    running (virtualbox)
node01                    running (virtualbox)
node02                    running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
```

Connect to each Node :

- sudo vagrant ssh master
- sudo vagrant ssh node01
- sudo vagrant ssh node02

Config Nodes : For each Node we should make this configuration :

Step 1 : SSH to Master and run the below commands

```
$ sudo su
# apt-get update
```

Step 2 : Install Docker

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

- `sudo mkdir /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
```

Restart Docker and enable on boot:

- `sudo systemctl enable docker`
- `sudo systemctl daemon-reload`
- `sudo systemctl restart docker`

Step 3 : Install kubeadm, Kubelet And Kubectl on Master.

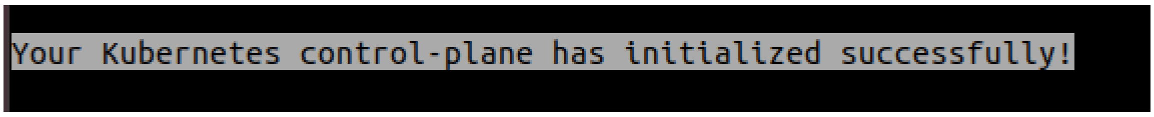
- kubeadm: the command to bootstrap the cluster.
- kubelet: the component that runs on all of the machines in your cluster and does things like starting pods and containers.
- kubectl: the command line utility to communicate with your cluster.

```
# apt-get update && apt-get install -y apt-transport-https curl
# 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 http://apt.kubernetes.io/ kubernetes-xenial main
EOF
# apt-get update
# apt-get install -y kubelet kubeadm kubectl
# apt-mark hold kubelet kubeadm kubectl
```

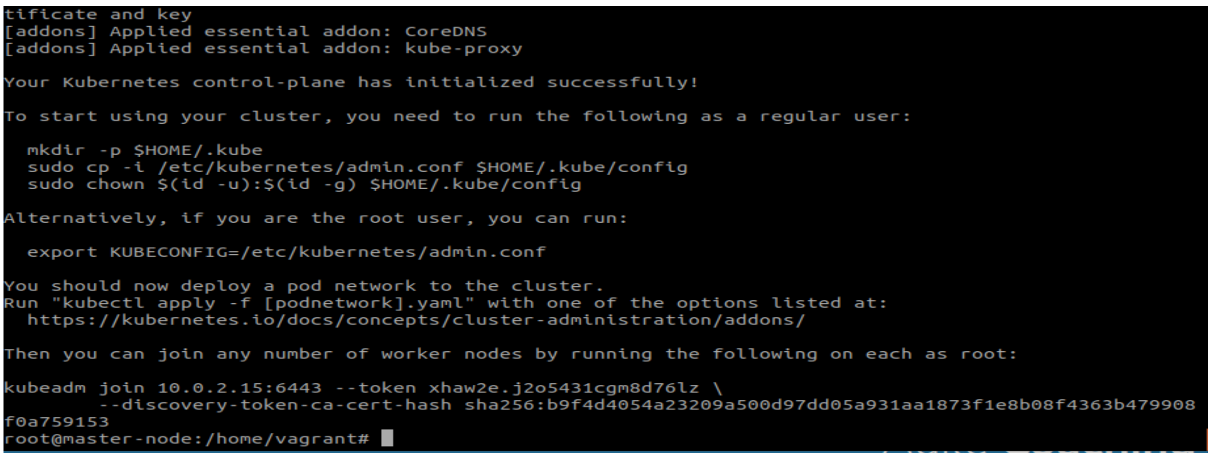
INIT CLUSTER : From the master Node :

- kubeadm init

it should be successful with generating token :



```
Your Kubernetes control-plane has initialized successfully!
```



```
tificate 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 10.0.2.15:6443 --token xhaw2e.j2o5431cgm8d76lz \
        --discovery-token-ca-cert-hash sha256:b9f4d4054a23209a500d97dd05a931aa1873f1e8b08f4363b479908
f0a759153
root@master-node:/home/vagrant# 
```

Should getting something like this:

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 10.0.2.15:6443 --token xhaw2e.j2o5431cgm8d76lz \

--discovery-token-ca-cert-hash
sha256:b9f4d4054a23209a500d97dd05a931aa1873f1e8b08f4363b479908f0a759153

Step 7 : To make kubectl work for your non-root user, run these commands.

```
# exit

$ mkdir -p $HOME/.kube

$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Step 8 : Let us now verify if kubectl is working as expected, run the below command.

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

Adding Worker Nodes to the Cluster:

For each Node (Node01 & Node02) we should Join the master cluster :

kubeadm join 10.0.2.15:6443 --token xhaw2e.j2o5431cgm8d76lz

--discovery-token-ca-cert-hash

sha256:b9f4d4054a23209a500d97dd05a931aa1873f1e8b08f4363b479908f0a759153

==

Install Vagrant on Ubuntu 18.04

To completely clean VM and start from fresh - the below worked for me - basically a combination of what others have said already.
Check VM status with vagrant locally and destroy if it exists - all done inside the vagrant folder - MAKE SURE you are in the correct folder!

- $ vagrant status
- $ vagrant destroy
- $ rm -rf .vagrant

Check VM status with vagrant globally and "destroy" if exists - can be done from anywhere

- $ vagrant global-status
- $ vagrant global-status --prune
-

Check VM status with VirtualBox's perspective and unregister VM

- $ vboxmanage list vms

Go back into appropriate vagrant folder and start VM
- sudo apt install qemu qemu-kvm libvirt-clients libvirt-daemon-system virtinst bridge-utils
- sudo apt update
- sudo apt install vagrant-libvirt
- sudo systemctl enable libvirtd
- sudo systemctl start libvirtd
- sudo apt-get install -y ebtables

Prepare Our Vagrant File :

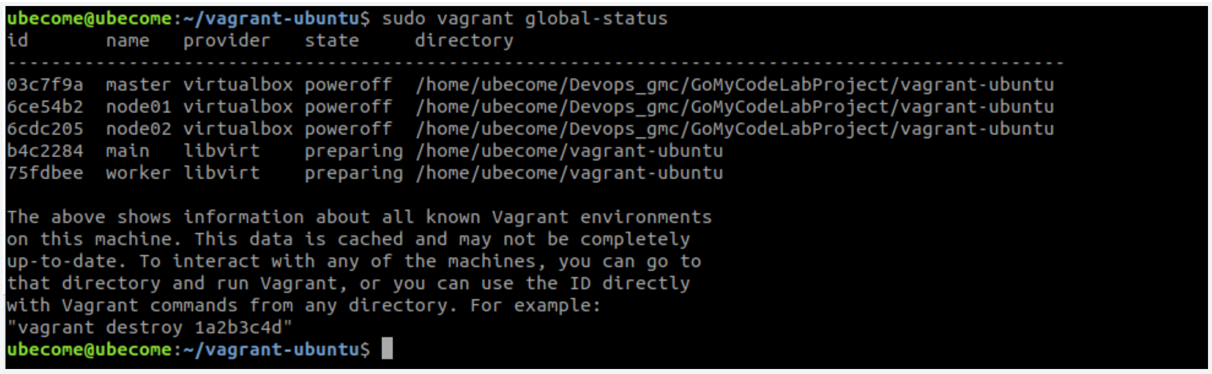
- cd ~/vagrant-ubuntu
- touch Vagrantfile

```ruby
# -- mode: ruby --
# vi: set ft=ruby:

nodes = [
    {:hostname => "main", :cpus => 2, :mem => 2048},
    {:hostname => "worker", :cpus => 2, :mem => 2048},
]

Vagrant.configure(2) do |config|
    nodes.each do |node|
        config.vm.define node[:hostname] do |vmachine|
            config.vm.box = "peru/ubuntu-20.04-server-amd64"
            config.vm.box_check_update = false
            vmachine.vm.hostname = node[:hostname]
            vmachine.vm.provider :libvirt do |domain|
                domain.memory = node[:mem]
                domain.cpus = node[:cpus]
            end
            vmachine.vm.provision :shell, path: "k8s-common.sh"
        end
    end
end
```

- $ vagrant up
- vagrant global-status

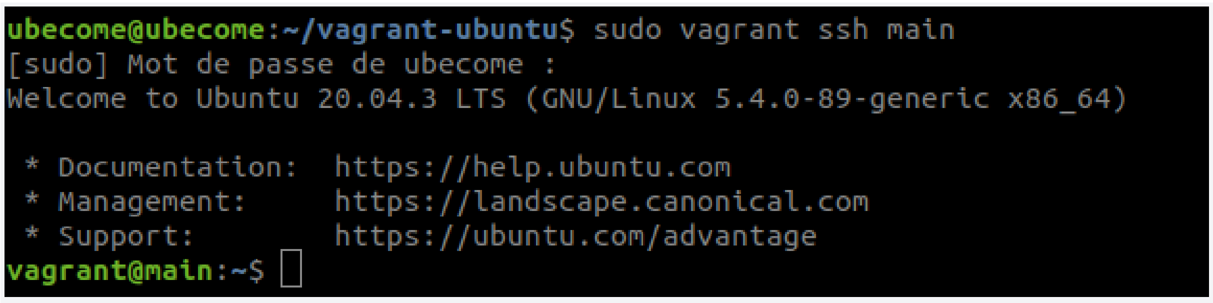


```
ubecome@ubecome:~/vagrant-ubuntu$ sudo vagrant global-status
id       name    provider   state     directory
------------------------------------------------------------------------------------------------
03c7f9a  master  virtualbox poweroff  /home/ubecome/Devops_gmc/GoMyCodeLabProject/vagrant-ubuntu
6ce54b2  node01  virtualbox poweroff  /home/ubecome/Devops_gmc/GoMyCodeLabProject/vagrant-ubuntu
6cdc205  node02  virtualbox poweroff  /home/ubecome/Devops_gmc/GoMyCodeLabProject/vagrant-ubuntu
b4c2284  main    libvirt    preparing /home/ubecome/vagrant-ubuntu
75fdbee  worker  libvirt    preparing /home/ubecome/vagrant-ubuntu

The above shows information about all known Vagrant environments
on this machine. This data is cached and may not be completely
up-to-date. To interact with any of the machines, you can go to
that directory and run Vagrant, or you can use the ID directly
with Vagrant commands from any directory. For example:
"vagrant destroy 1a2b3c4d"
ubecome@ubecome:~/vagrant-ubuntu$ 
```

Connect To Our Nodes :

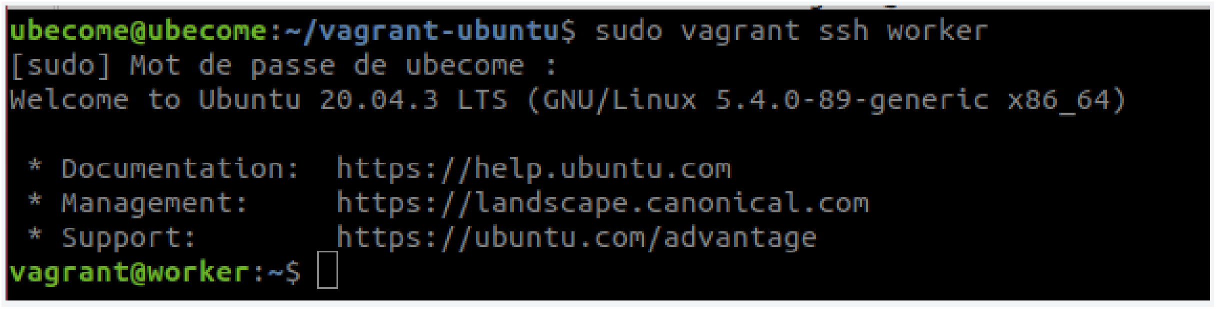
- sudo vagrant ssh main



```
ubecome@ubecome:~/vagrant-ubuntu$ sudo vagrant ssh main
[sudo] Mot de passe de ubecome :
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-89-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
vagrant@main:~$ 
```

- sudo vagrant ssh worker

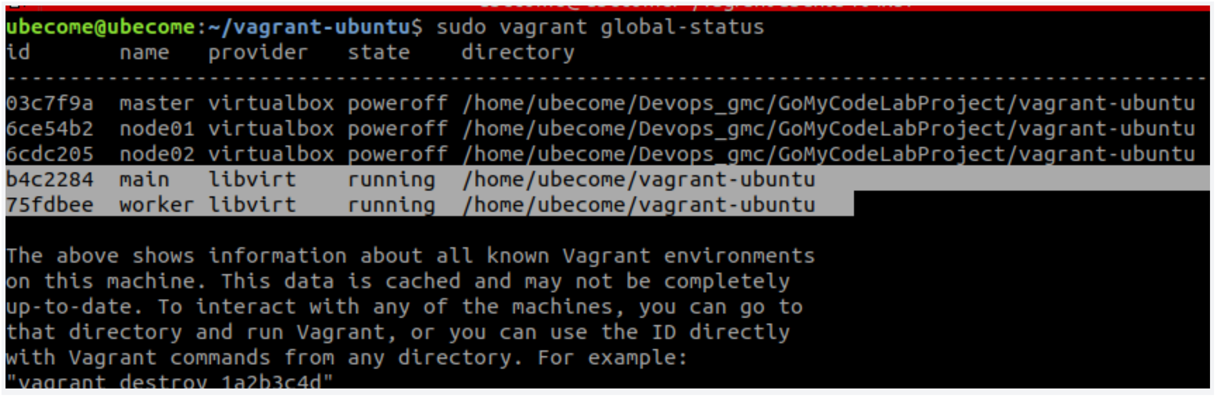


```
ubecome@ubecome:~/vagrant-ubuntu$ sudo vagrant ssh worker
[sudo] Mot de passe de ubecome :
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-89-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
vagrant@worker:~$
```

Now Both Nodes main & worker are connected :

- vagrant global-status



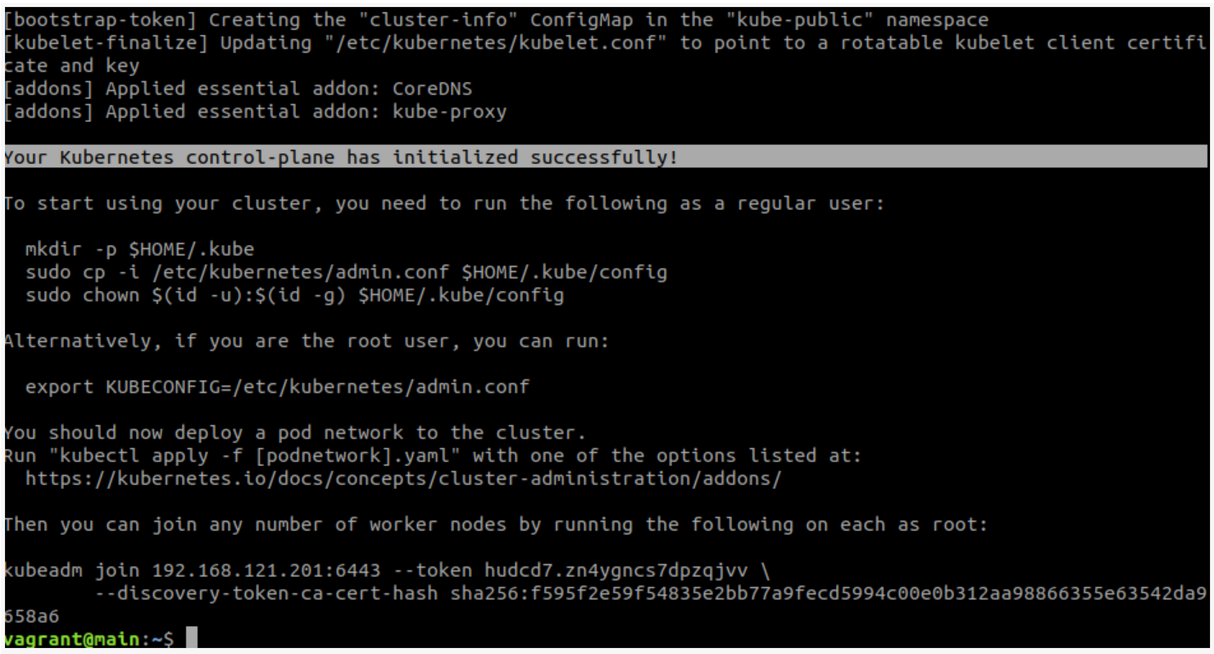
```
ubecome@ubecome:~/vagrant-ubuntu$ sudo vagrant global-status
id       name    provider   state    directory
-----------------------------------------------------------------------------------------------
03c7f9a  master  virtualbox poweroff /home/ubecome/Devops_gmc/GoMyCodeLabProject/vagrant-ubuntu
6ce54b2  node01  virtualbox poweroff /home/ubecome/Devops_gmc/GoMyCodeLabProject/vagrant-ubuntu
6cdc205  node02  virtualbox poweroff /home/ubecome/Devops_gmc/GoMyCodeLabProject/vagrant-ubuntu
b4c2284  main    libvirt    running  /home/ubecome/vagrant-ubuntu
75fdbee  worker  libvirt    running  /home/ubecome/vagrant-ubuntu

The above shows information about all known Vagrant environments
on this machine. This data is cached and may not be completely
up-to-date. To interact with any of the machines, you can go to
that directory and run Vagrant, or you can use the ID directly
with Vagrant commands from any directory. For example:
"vagrant destroy 1a2b3c4d"
```

`INIT CLUSTER : From the master Node :`

- kubeadm init

it should be successful with generating token :



```
[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 certifi
cate 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 192.168.121.201:6443 --token hudcd7.zn4ygncs7dpzqjvv \
        --discovery-token-ca-cert-hash sha256:f595f2e59f54835e2bb77a9fecd5994c00e0b312aa98866355e63542da9
658a6
vagrant@main:~$
```

# copying config file :

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

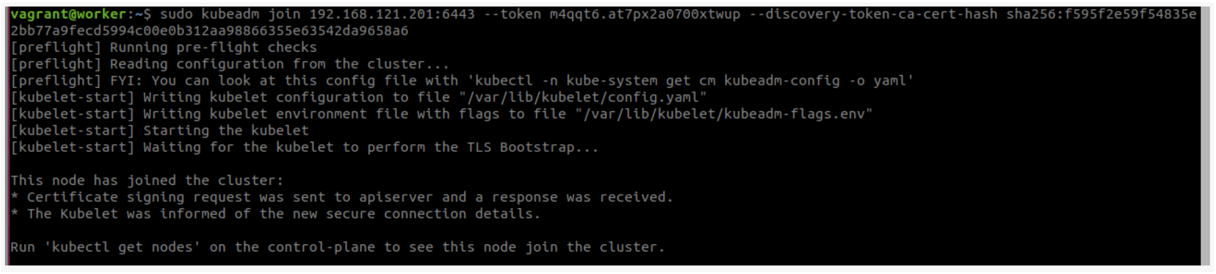
# adding calico networking solution :

- kubectl apply -f https://docs.projectcalico.org/v3.20/manifests/calico.yaml

# generating kubejoin.sh : This will return the command join for worker Node to join the cluster

- kubeadm token create --print-join-command

# now copy the output of the last command, and execute it on the worker node(Maybe we need to execute the command as root with **sudo**):

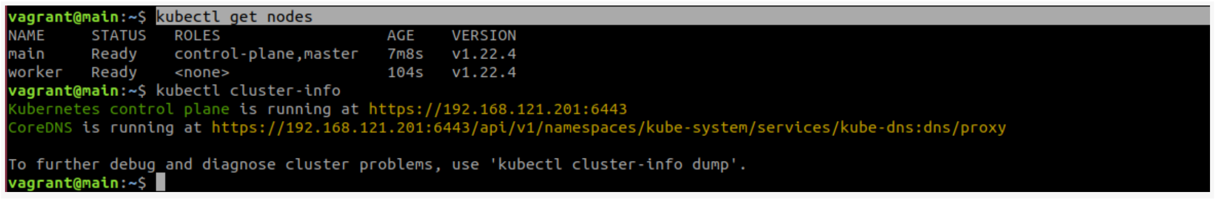


```
vagrant@worker:~$ sudo kubeadm join 192.168.121.201:6443 --token m4qqt6.at7px2a0700xtwup --discovery-token-ca-cert-hash sha256:f595f2e59f54835e
2bb77a9fecd5994c00e0b312aa98866355e63542da9658a6
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[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.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.
```

Now we get The Node joined the cluster:

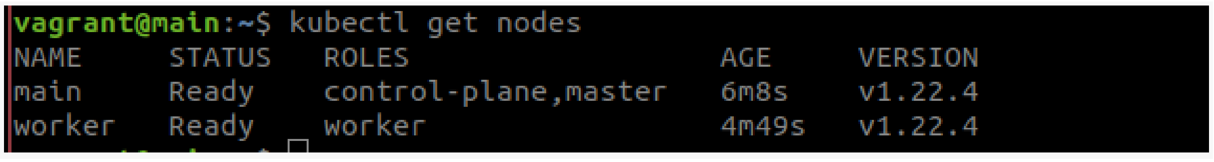


```
vagrant@main:~$ kubectl get nodes
NAME     STATUS   ROLES                  AGE    VERSION
main     Ready    control-plane,master   7m8s   v1.22.4
worker   Ready    <none>                 104s   v1.22.4
vagrant@main:~$ kubectl cluster-info
Kubernetes control plane is running at https://192.168.121.201:6443
CoreDNS is running at https://192.168.121.201:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
vagrant@main:~$
```

We need To label our worker node Role as worker instead of NONE :

- kubectl label nodes worker kubernetes.io/role=worker



```
vagrant@main:~$ kubectl get nodes
NAME     STATUS   ROLES                  AGE     VERSION
main     Ready    control-plane,master   6m8s    v1.22.4
worker   Ready    worker                 4m49s   v1.22.4
```