EXPERIMENT 3

Aim:

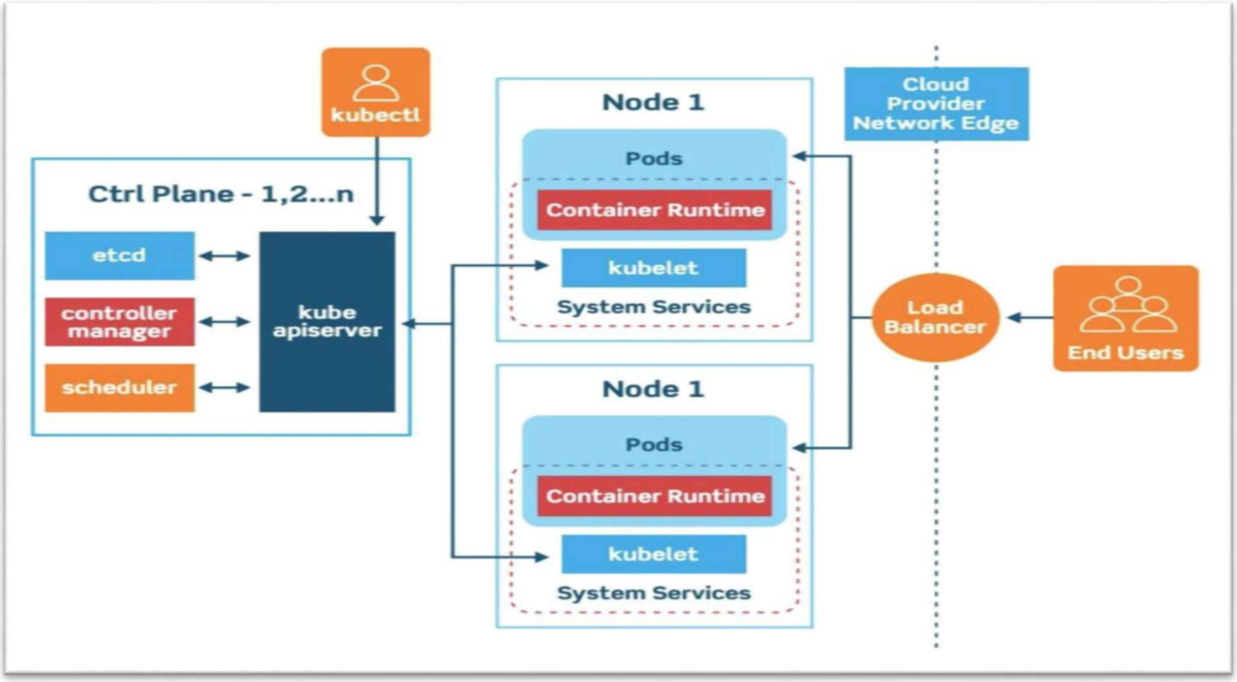
To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Theory:

Kubernetes is an open-source platform for deploying and managing containers. It provides a container runtime, container orchestration, container-centric infrastructure orchestration, self- healing mechanisms, service discovery and load balancing. It’s used for the deployment, scaling, management, and composition of application containers across clusters of hosts.

But Kubernetes is more than just a container orchestrator. It could be thought of as the operating system for cloud-native applications in the sense that it’s the platform that applications run on, just as desktop applications run on MacOS, Windows, or Linux.

From a high level, a Kubernetes environment consists of a control plane (master), a distributed storage system for keeping the cluster state consistent (etcd), and a number of cluster nodes.



The control plane is the system that maintains a record of all Kubernetes objects. It continuously manages object states, responding to changes in the cluster; it also works to make the actual state of system objects match the desired state. As the above illustration shows, the control plane is made up of three major components:

- kube-apiserver
- kube-controller-manager
- kube-scheduler

Cluster nodes are machines that run containers and are managed by the master nodes. The Kubelet is the primary and most important controller in Kubernetes. It's responsible for driving the container execution layer, typically Docker.

Pods are one of the crucial concepts in Kubernetes, as they are the key construct that developers interact with. The previous concepts are infrastructure-focused and internal architecture

Kubernetes Tooling and Clients:

Here are the basic tools you should know:

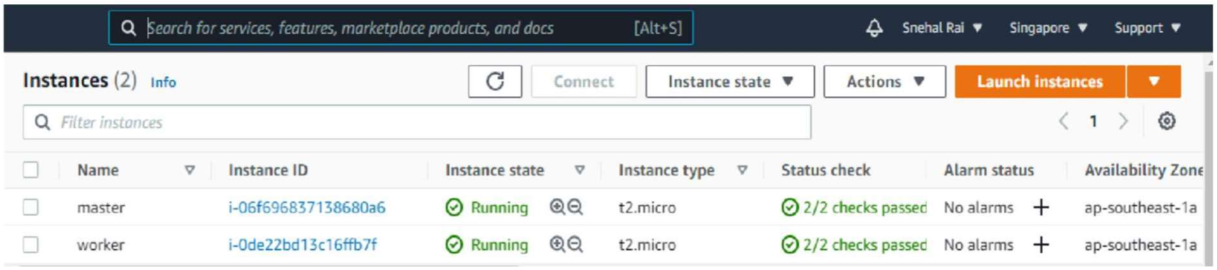
1. Kubeadm bootstraps a cluster. It's designed to be a simple way for new users to build clusters (more detail on this is in a later chapter).
2. Kubectl is a tool for interacting with your existing cluster.
3. Minikube is a tool that makes it easy to run Kubernetes locally. For Mac users, HomeBrew makes using Minikube even simpler.

Pre-requisite:

- Launch two or more Linux servers running Ubuntu 18.04 /20.04 on Virtual box

OR

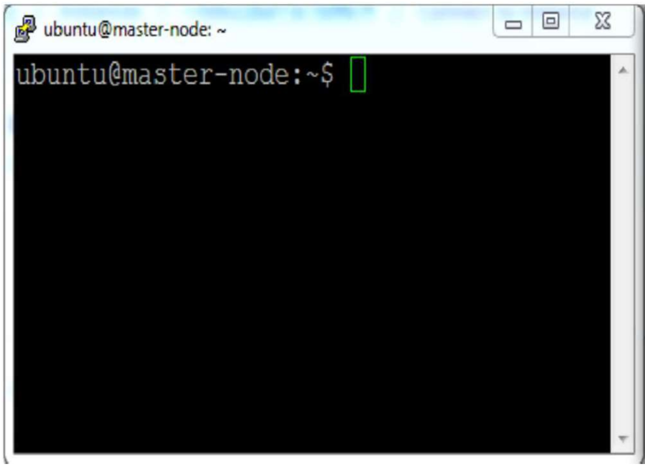
- Launch two or more EC2 instances of Ubuntu 20.04 AMI free tier.



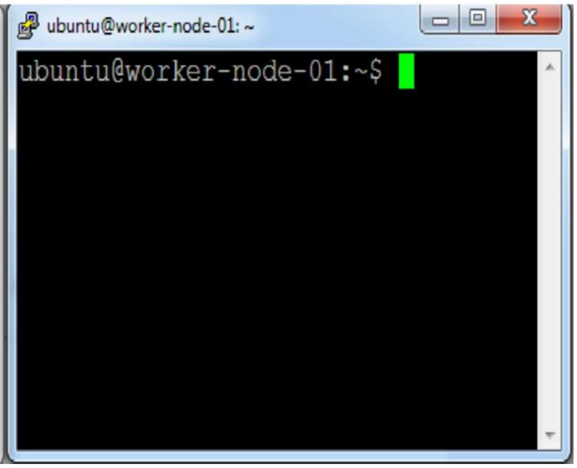
Instances (2) Info

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
□	master	i-06f696837138680a6	✓ Running	t2.micro	✓ 2/2 checks passed	No alarms +	ap-southeast-1a
□	worker	i-0de22bd13c16ffb7f	✓ Running	t2.micro	✓ 2/2 checks passed	No alarms +	ap-southeast-1a

- If using EC2 instances, connect to all instances using PUTTY (on Windows) or using.



```
ubuntu@master-node:~$
```



```
ubuntu@worker-node-01:~$
```

- Access to a user account on each system with sudo or root privileges.

- The apt package manager is included by default.

Steps to Install Kubernetes on Ubuntu.

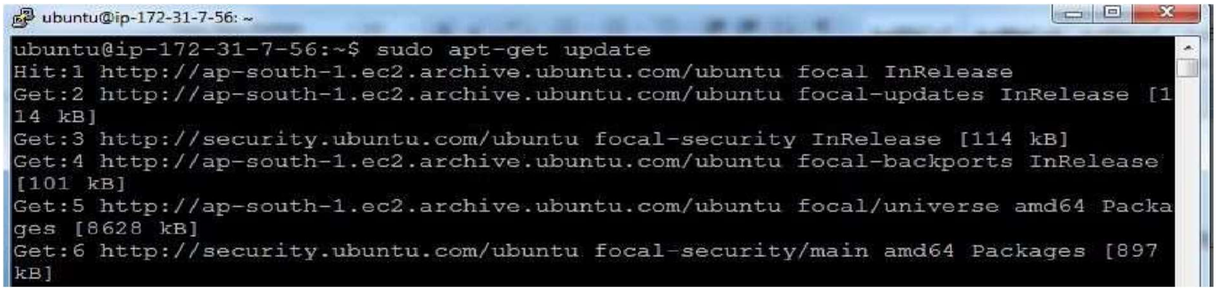
- Set up Docker

Step 1: Install Docker

Kubernetes requires an existing Docker installation. If you already have Docker installed, skip ahead to Step 2. If you do not have Kubernetes, install it by following these steps:

Update the package list with the command:

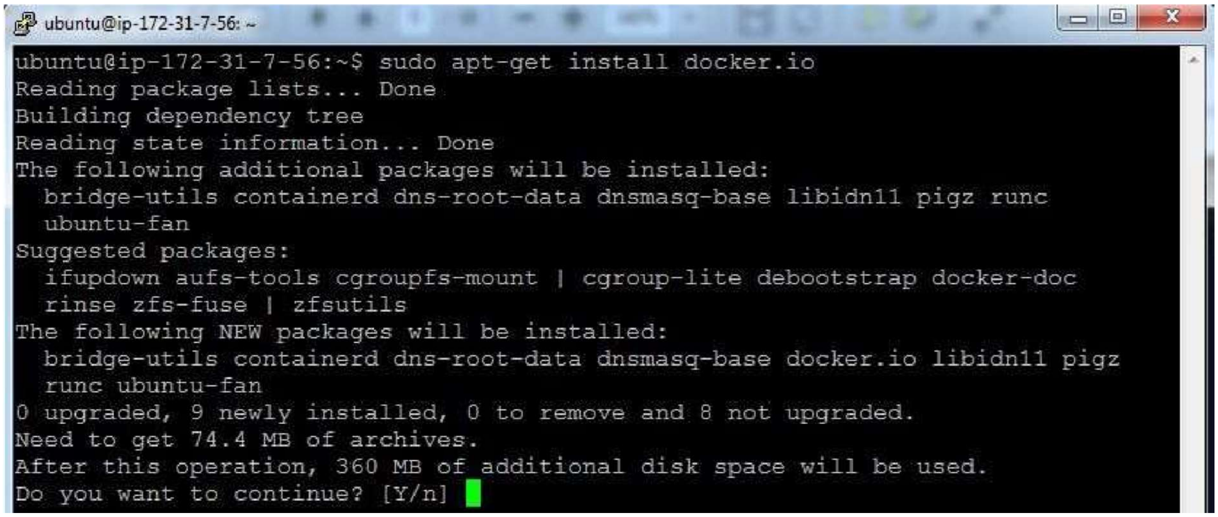
```
$  sudo apt-get update
```



```
ubuntu@ip-172-31-7-56:~$ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [1
14 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease
[101 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 Packa
ges [8628 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [897
kB]
```

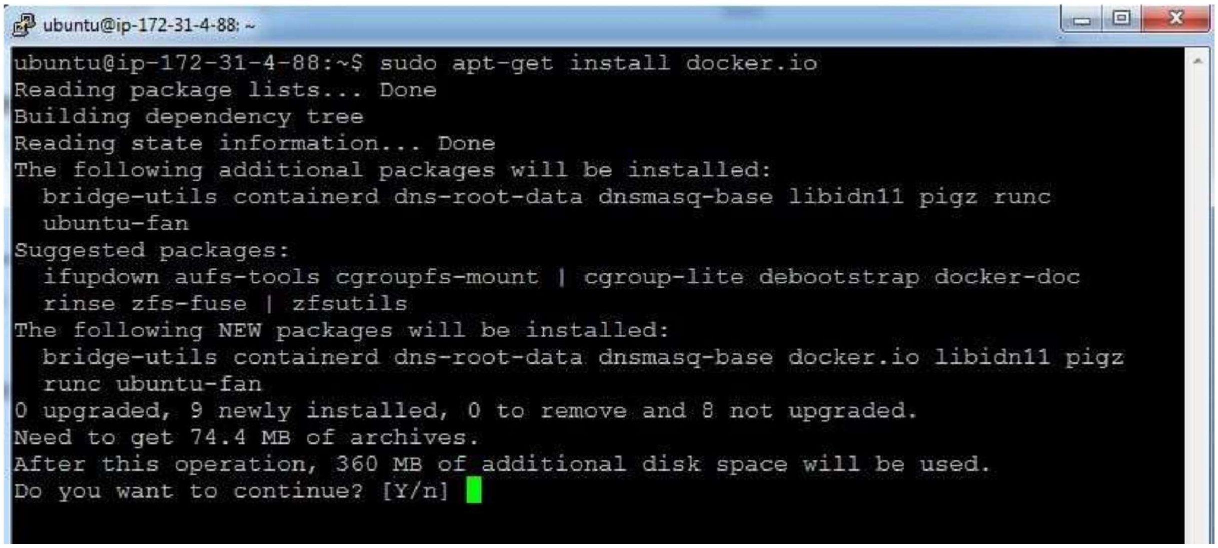
Next, install Docker with the command:

$ sudo apt-get install docker.io



```
ubuntu@ip-172-31-7-56:~$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base libidn11 pigz runc
  ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc
  rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io libidn11 pigz
  runc ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 8 not upgraded.
Need to get 74.4 MB of archives.
After this operation, 360 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

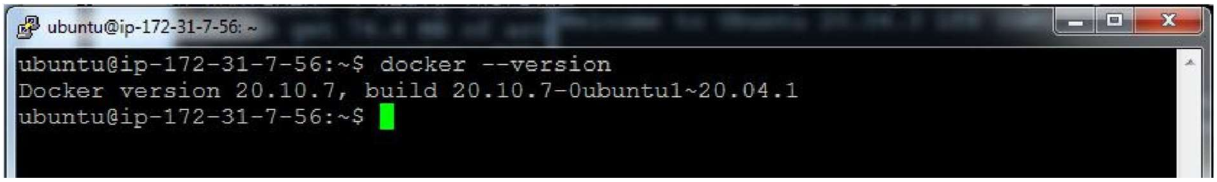
Repeat the process on each server that will act as a node.



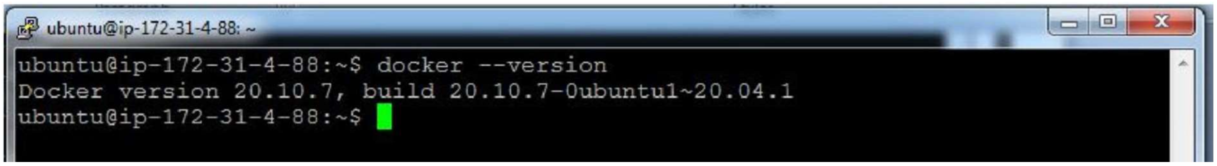
```
ubuntu@ip-172-31-4-88:~$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base libidn11 pigz runc
  ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc
  rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io libidn11 pigz
  runc ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 8 not upgraded.
Need to get 74.4 MB of archives.
After this operation, 360 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Check the installation (and version) by entering the following:

$ docker —version



```
ubuntu@ip-172-31-7-56:~$ docker --version
Docker version 20.10.7, build 20.10.7-0ubuntu1~20.04.1
ubuntu@ip-172-31-7-56:~$
```



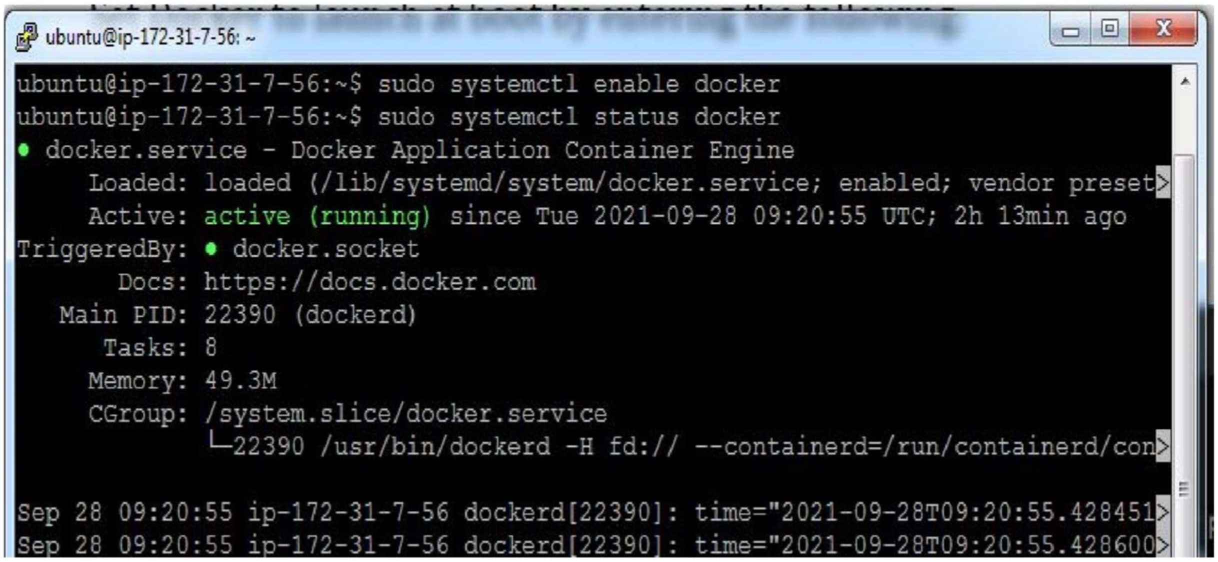
```
ubuntu@ip-172-31-4-88:~$ docker --version
Docker version 20.10.7, build 20.10.7-0ubuntu1~20.04.1
ubuntu@ip-172-31-4-88:~$
```

Step 2 : Start and Enable Docker

Set Docker to launch at boot by entering the following:

$ sudo systemctl enable docker

$ sudo systemctl status docker

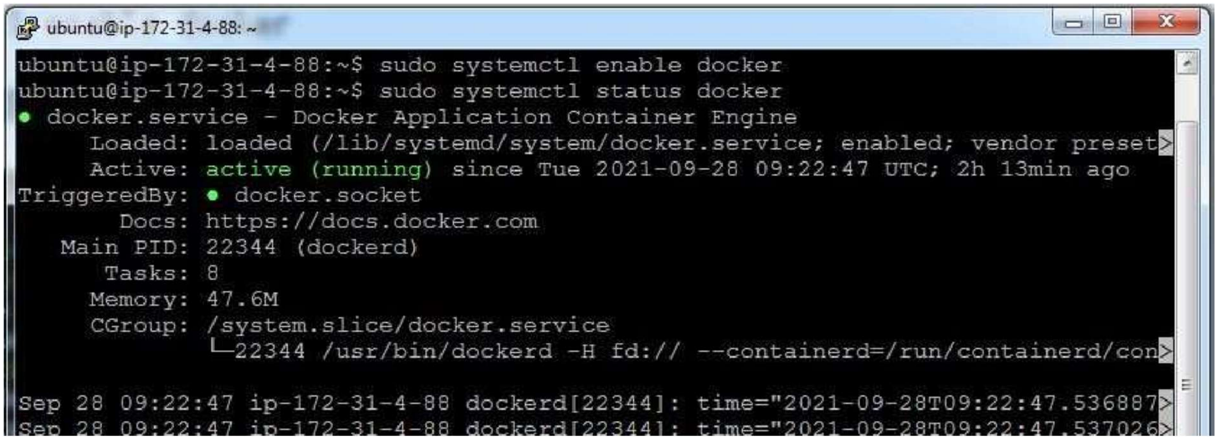


```
ubuntu@ip-172-31-7-56:~$ sudo systemctl enable docker
ubuntu@ip-172-31-7-56:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset>
     Active: active (running) since Tue 2021-09-28 09:20:55 UTC; 2h 13min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 22390 (dockerd)
      Tasks: 8
     Memory: 49.3M
     CGroup: /system.slice/docker.service
             └─22390 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/con>

Sep 28 09:20:55 ip-172-31-7-56 dockerd[22390]: time="2021-09-28T09:20:55.428451>
Sep 28 09:20:55 ip-172-31-7-56 dockerd[22390]: time="2021-09-28T09:20:55.428600>
```

To start Docker if it’s not running:

$ sudo systemctl start docker

Repeat on all the other nodes.



```
ubuntu@ip-172-31-4-88:~$ sudo systemctl enable docker
ubuntu@ip-172-31-4-88:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset>
     Active: active (running) since Tue 2021-09-28 09:22:47 UTC; 2h 13min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 22344 (dockerd)
      Tasks: 8
     Memory: 47.6M
     CGroup: /system.slice/docker.service
             └─22344 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/con>

Sep 28 09:22:47 ip-172-31-4-88 dockerd[22344]: time="2021-09-28T09:22:47.536887>
Sep 28 09:22:47 ip-172-31-4-88 dockerd[22344]: time="2021-09-28T09:22:47.537026>
```

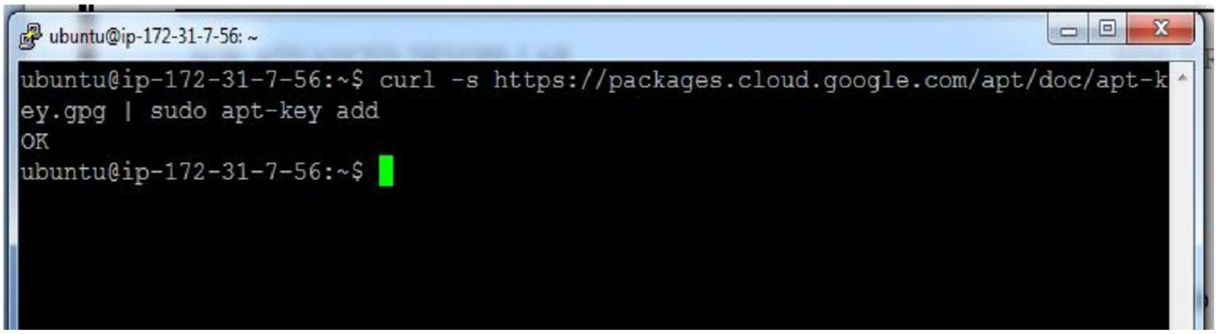
- Install Kubernetes

Step 3 : Add Kubernetes Signing Key

Since you are downloading Kubernetes from a non-standard repository, it is essential to ensure that the software is authentic. This is done by adding a signing key.

Enter the following to add a signing key:

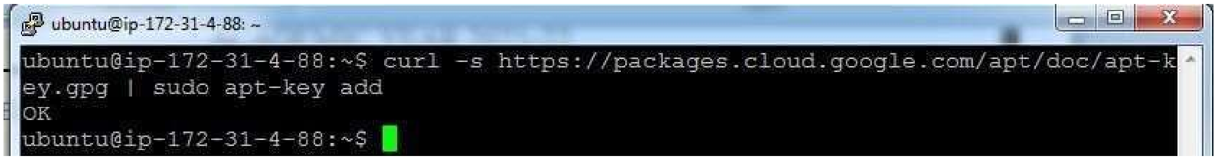
$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add



```
ubuntu@ip-172-31-7-56:~$ curl -s https://packages.cloud.google.com/apt/doc/apt-k
ey.gpg | sudo apt-key add
OK
ubuntu@ip-172-31-7-56:~$
```

If you get an error that curl is not installed, install it with:

$ sudo apt-get install curlThen repeat the previous command to install the signing keys. Repeat for each server node.

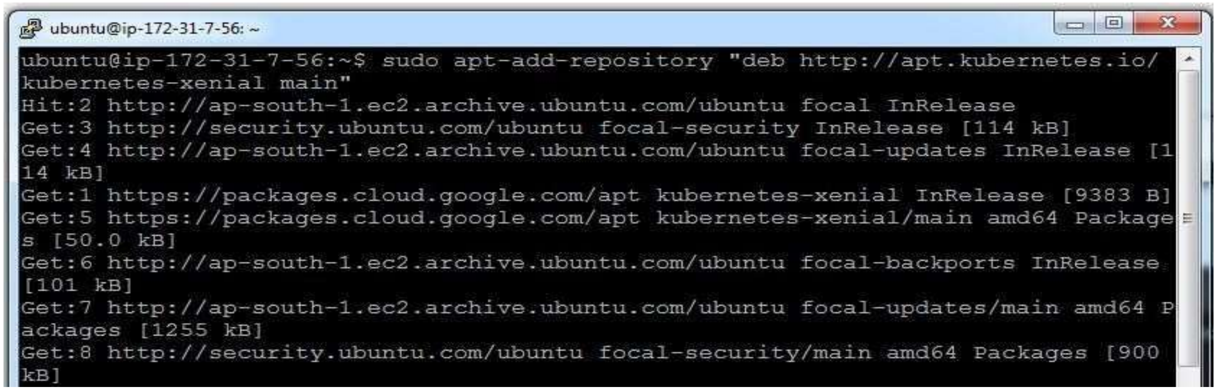


```
ubuntu@ip-172-31-4-88:~$ curl -s https://packages.cloud.google.com/apt/doc/apt-k
ey.gpg | sudo apt-key add
OK
ubuntu@ip-172-31-4-88:~$
```

Step 4: Add Software Repositories

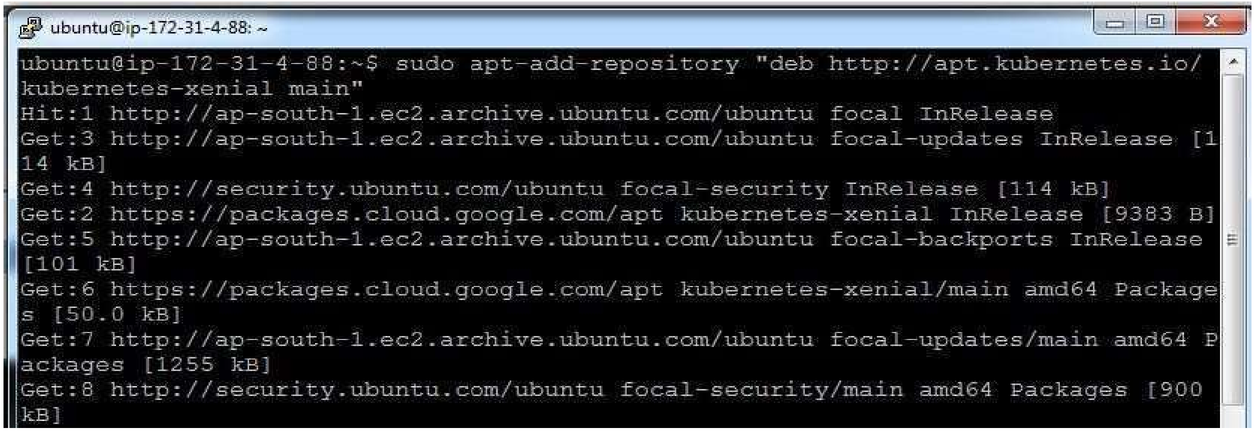
Kubernetes is not located in the default repositories. To add them enter the following:

$ sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"



```
ubuntu@ip-172-31-7-56:~$ sudo apt-add-repository "deb http://apt.kubernetes.io/
kubernetes-xenial main"
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:3 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [1
14 kB]
Get:1 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [9383 B]
Get:5 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Package
s [50.0 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease
[101 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 P
ackages [1255 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [900
kB]
```

Repeat on each server node.



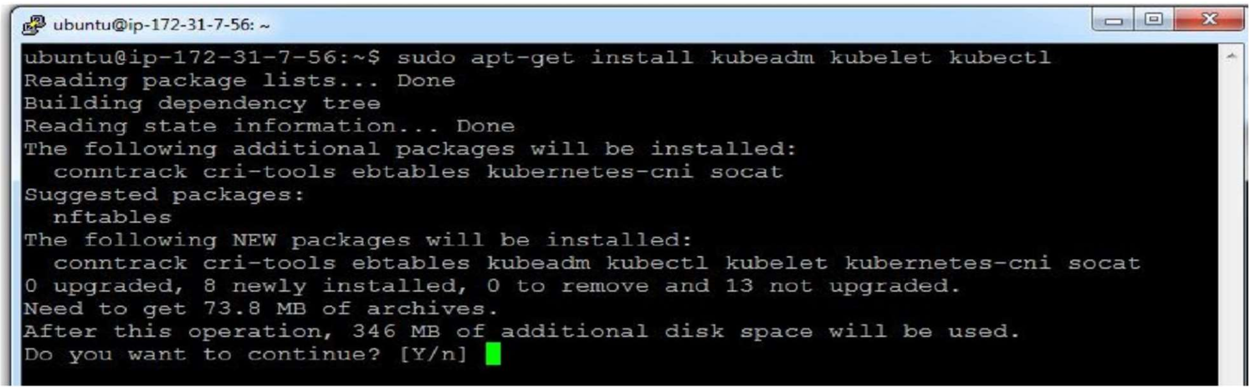
```
ubuntu@ip-172-31-4-88:~$ sudo apt-add-repository "deb http://apt.kubernetes.io/
kubernetes-xenial main"
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [1
14 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:2 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [9383 B]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease
[101 kB]
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Package
s [50.0 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 P
ackages [1255 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [900
kB]
```

Step 5 : Kubernetes Installation Tools

Kubeadm (Kubernetes Admin) is a tool that helps initialize a cluster. It fast-tracks setup by using community-sourced best practices. Kubelet is the work package, which runs on every node and starts containers. The tool gives you command-line access to clusters.

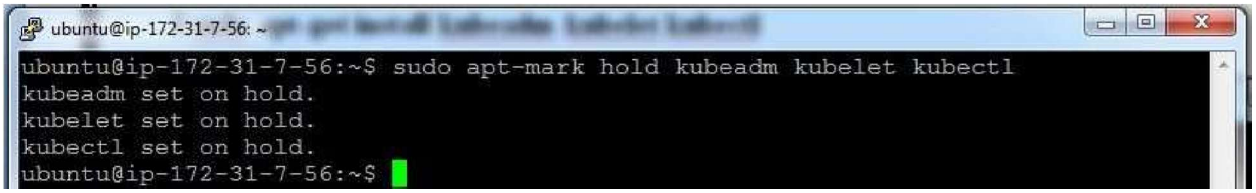
Install Kubernetes tools with the command:

$ sudo apt-get install kubeadm kubelet kubectl



```
ubuntu@ip-172-31-7-56:~$ sudo apt-get install kubeadm kubelet kubectl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools ebtables kubernetes-cni socat
Suggested packages:
  nftables
The following NEW packages will be installed:
  conntrack cri-tools ebtables kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 8 newly installed, 0 to remove and 13 not upgraded.
Need to get 73.8 MB of archives.
After this operation, 346 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

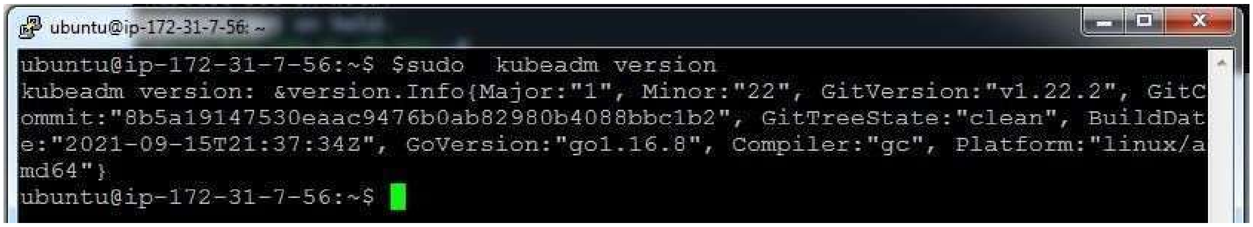
$ sudo apt-mark hold kubeadm kubelet kubectl



```
ubuntu@ip-172-31-7-56:~$ sudo apt-mark hold kubeadm kubelet kubectl
kubeadm set on hold.
kubelet set on hold.
kubectl set on hold.
ubuntu@ip-172-31-7-56:~$
```

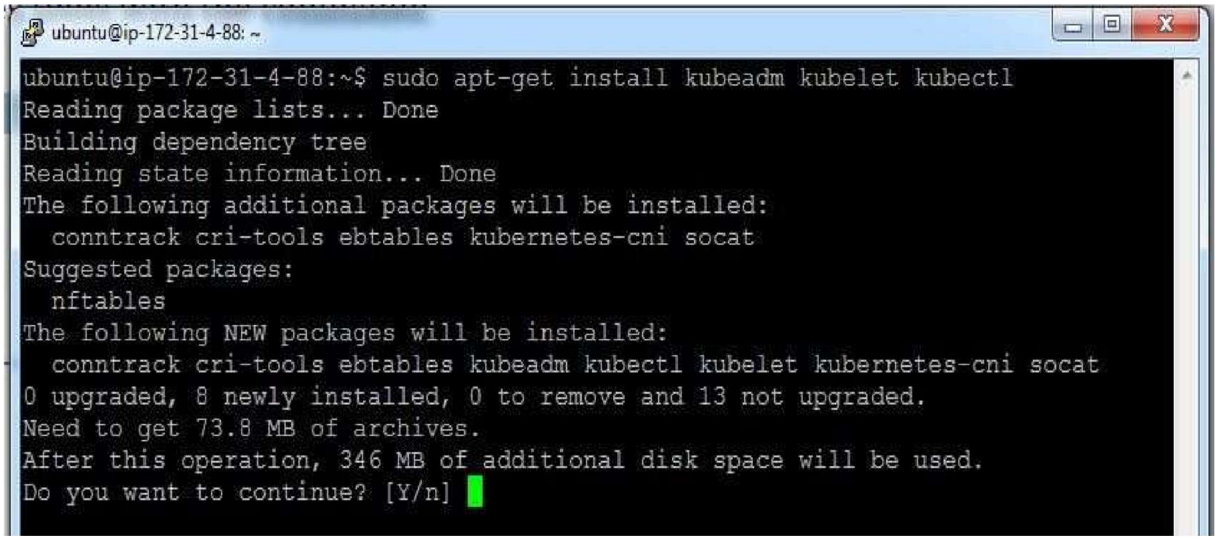
Allow the process to complete. Verify the installation with:

$ kubeadm version

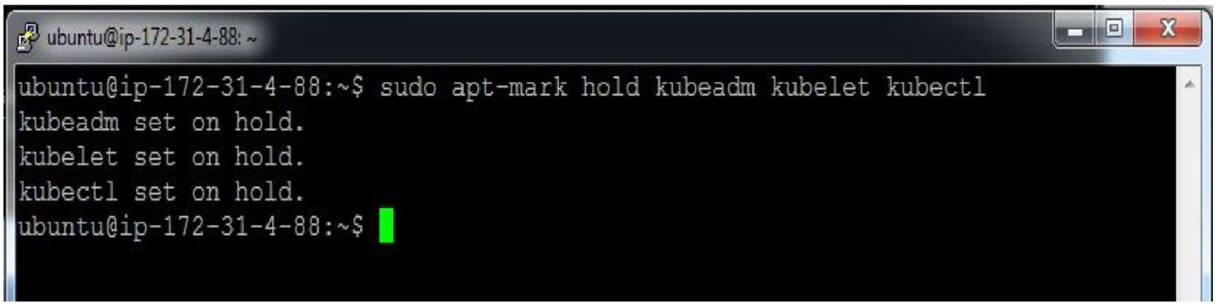


```
ubuntu@ip-172-31-7-56:~$ $sudo  kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.2", GitC
ommit:"8b5a19147530eaac9476b0ab82980b4088bbc1b2", GitTreeState:"clean", BuildDat
e:"2021-09-15T21:37:34Z", GoVersion:"go1.16.8", Compiler:"gc", Platform:"linux/a
md64"}
ubuntu@ip-172-31-7-56:~$
```

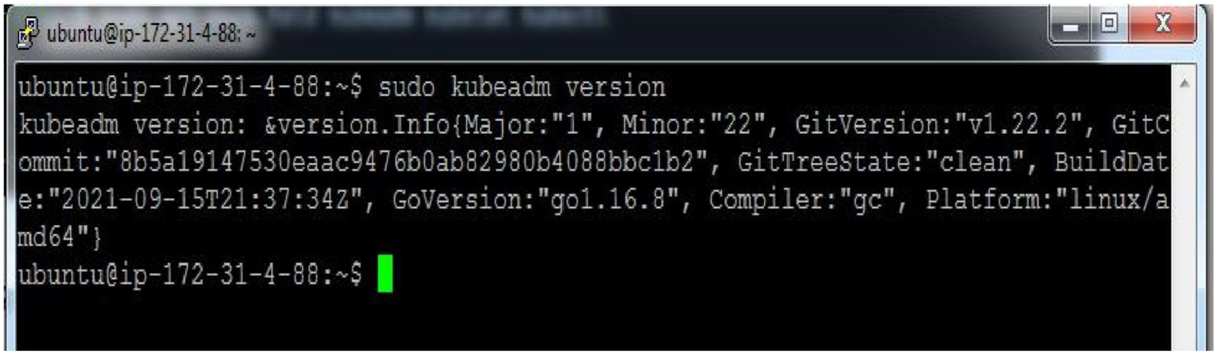
Repeat for each server node.



```
ubuntu@ip-172-31-4-88:~$ sudo apt-get install kubeadm kubelet kubectl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools ebtables kubernetes-cni socat
Suggested packages:
  nftables
The following NEW packages will be installed:
  conntrack cri-tools ebtables kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 8 newly installed, 0 to remove and 13 not upgraded.
Need to get 73.8 MB of archives.
After this operation, 346 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```



```
ubuntu@ip-172-31-4-88:~$ sudo apt-mark hold kubeadm kubelet kubectl
kubeadm set on hold.
kubelet set on hold.
kubectl set on hold.
ubuntu@ip-172-31-4-88:~$
```



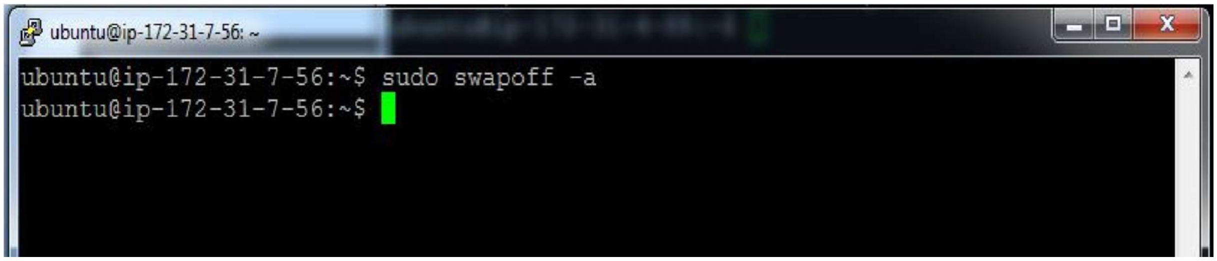
```
ubuntu@ip-172-31-4-88:~$ sudo kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.2", GitC
ommit:"8b5a19147530eaac9476b0ab82980b4088bbc1b2", GitTreeState:"clean", BuildDat
e:"2021-09-15T21:37:34Z", GoVersion:"go1.16.8", Compiler:"gc", Platform:"linux/a
md64"}
ubuntu@ip-172-31-4-88:~$
```

- Kubernetes Deployment

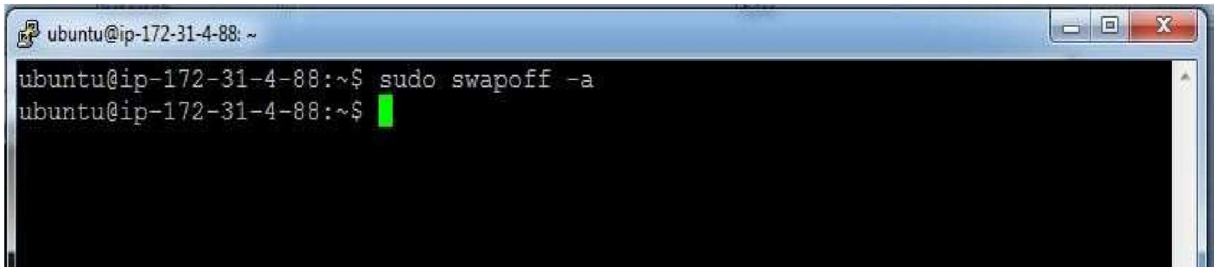
Step 6 : Begin Kubernetes Deployment

Start by disabling the swap memory on each server:

$ sudo swapoff –a



```
ubuntu@ip-172-31-7-56:~$ sudo swapoff -a
ubuntu@ip-172-31-7-56:~$
```

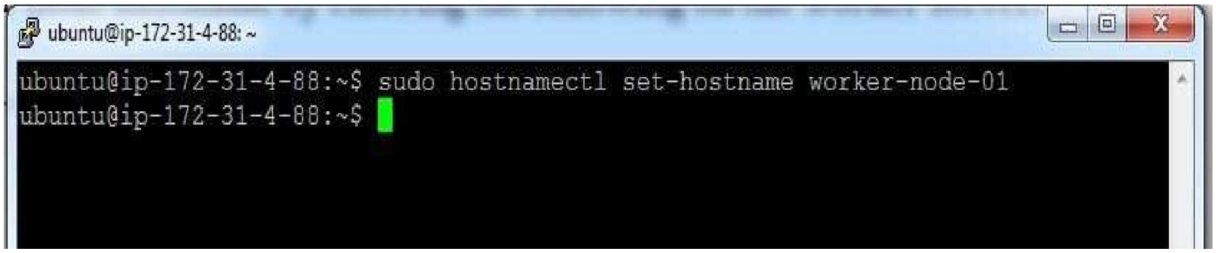


```
ubuntu@ip-172-31-4-88:~$ sudo swapoff -a
ubuntu@ip-172-31-4-88:~$
```

Step 7 : Assign Unique Hostname for Each Server Node

Decide which server to set as the master node. Then enter the command:

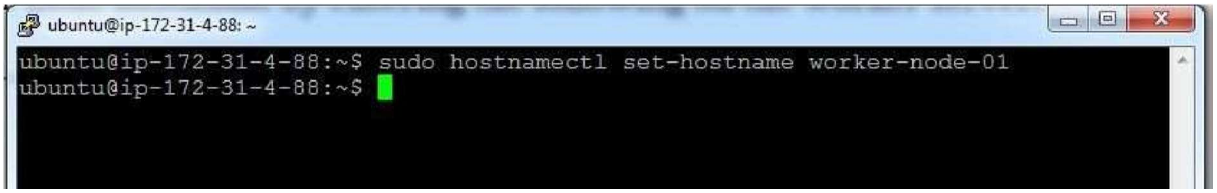
$ sudo hostnamectl set-hostname master-node



```
ubuntu@ip-172-31-4-88:~$ sudo hostnamectl set-hostname worker-node-01
ubuntu@ip-172-31-4-88:~$
```

Next, set a worker node hostname by entering the following on the worker server:

$ sudo hostnamectl set-hostname worker-node-01



```
ubuntu@ip-172-31-4-88:~$ sudo hostnamectl set-hostname worker-node-01
ubuntu@ip-172-31-4-88:~$
```

If you have additional worker nodes, use this process to set a unique hostname on each. Logout and again login to see the results.

Step 8 : Initialize Kubernetes on Master Node

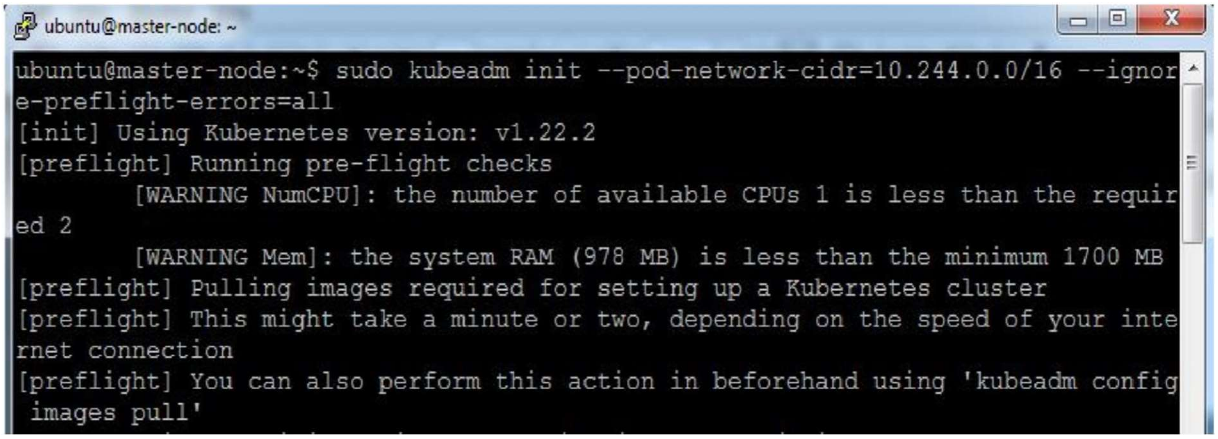
Switch to the master server node, and enter the following:

$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16

If you are trying to run this on EC2 you'll get an error message saying less CPU and memory to override the error run the above command with --ignore-preflight-errors=all

For eg:

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



```
ubuntu@master-node:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignor
e-preflight-errors=all
[init] Using Kubernetes version: v1.22.2
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the requir
ed 2
        [WARNING Mem]: the system RAM (978 MB) is less than the minimum 1700 MB
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your inte
rnet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config
 images pull'
```

If the kubeadm init command ran without error then ignore this part. If you receive this error "kubelet isn't running or healthy", then do the following.

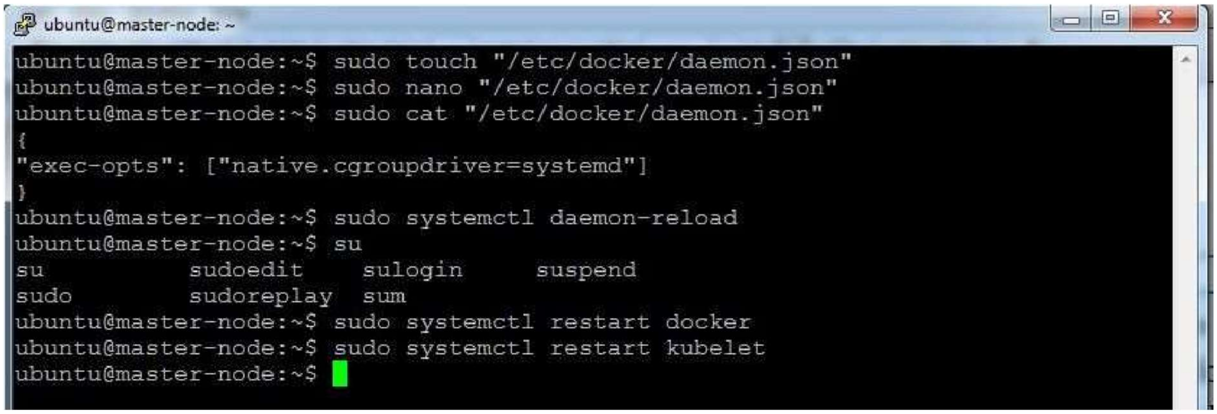
Create file daemon.json in /etc/docker/ and add following lines in the file.

{

"exec-opts": ["native.cgroupdriver=systemd"]

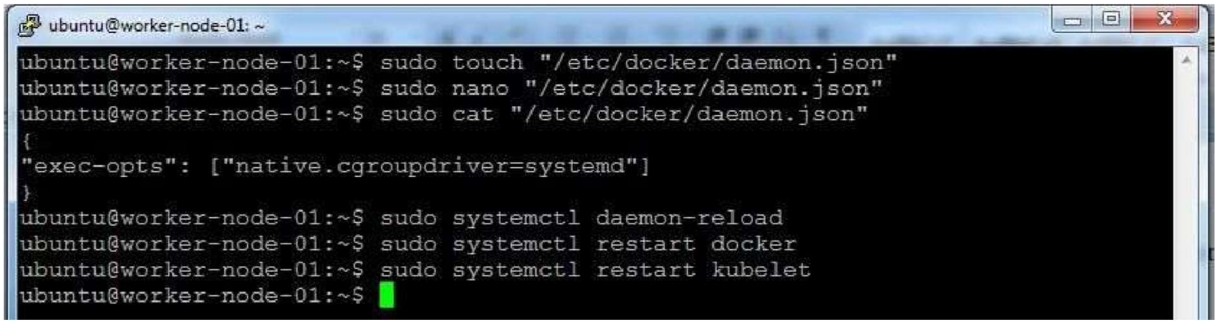
}

And run the following commands.



```
ubuntu@master-node:~$ sudo touch "/etc/docker/daemon.json"
ubuntu@master-node:~$ sudo nano "/etc/docker/daemon.json"
ubuntu@master-node:~$ sudo cat "/etc/docker/daemon.json"
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@master-node:~$ sudo systemctl daemon-reload
ubuntu@master-node:~$ su
su          sudoedit    sulogin     suspend
sudo        sudoreplay  sum
ubuntu@master-node:~$ sudo systemctl restart docker
ubuntu@master-node:~$ sudo systemctl restart kubelet
ubuntu@master-node:~$
```

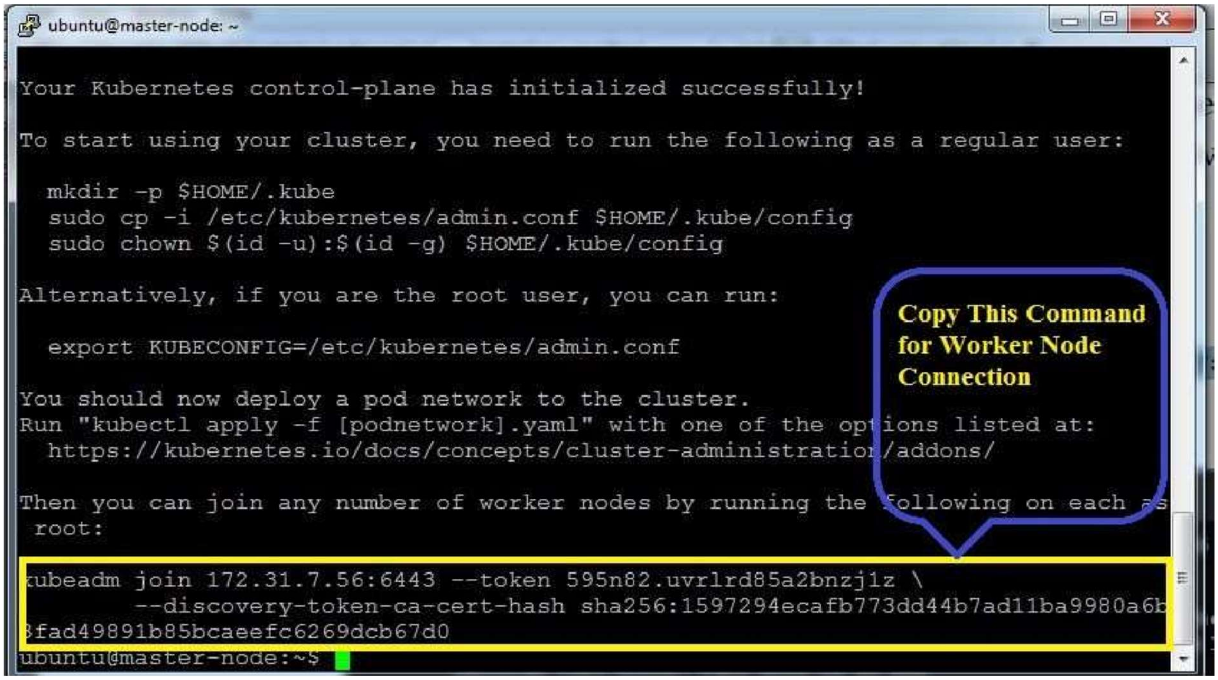
Do this on both master and worker nodes.



```
ubuntu@worker-node-01:~$ sudo touch "/etc/docker/daemon.json"
ubuntu@worker-node-01:~$ sudo nano "/etc/docker/daemon.json"
ubuntu@worker-node-01:~$ sudo cat "/etc/docker/daemon.json"
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@worker-node-01:~$ sudo systemctl daemon-reload
ubuntu@worker-node-01:~$ sudo systemctl restart docker
ubuntu@worker-node-01:~$ sudo systemctl restart kubelet
ubuntu@worker-node-01:~$
```

After this run sudo kubeadm reset command and then the init or join command .

Once this command finishes, it will display a kubeadm join message at the end. Make a note of the whole enty. This will be used to join the worker nodes to the cluster.



```
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 172.31.7.56:6443 --token 595n82.uvrlrd85a2bnzj1z \
        --discovery-token-ca-cert-hash sha256:1597294ecafb773dd44b7ad11ba9980a6b
3fad49891b85bcaeefc6269dcb67d0
ubuntu@master-node:~$
```

Copy This Command for Worker Node Connection

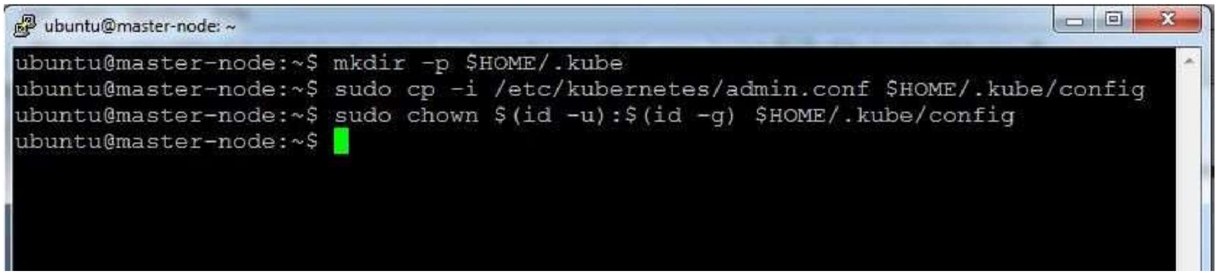
Next, enter the following to create a directory for the cluster:

kubernetes-master $ mkdir -p $HOME/.kube

kubernetes-master $ sudo cp -i /etc/kubernetes/admin.conf

$HOME/.kube/config

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



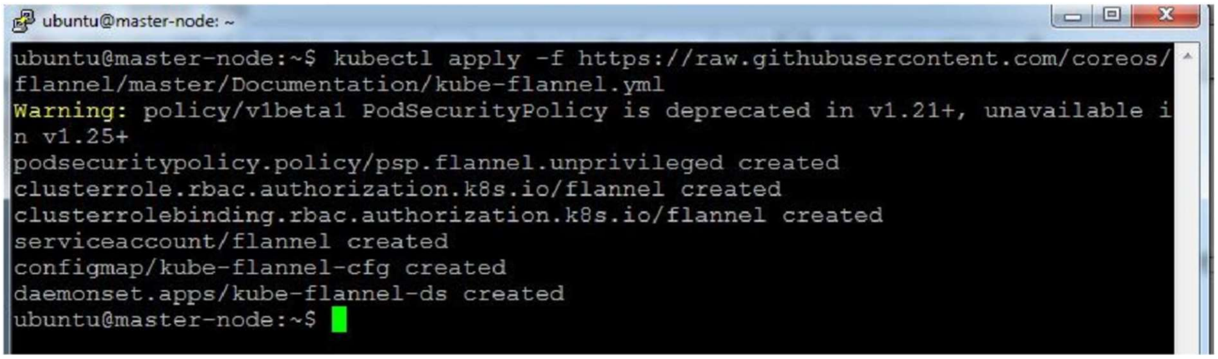
```
ubuntu@master-node:~$ mkdir -p $HOME/.kube
ubuntu@master-node:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
ubuntu@master-node:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@master-node:~$
```

Step 9 : Deploy Pod Network to Cluster

A Pod Network is a way to allow communication between different nodes in the cluster. This tutorial uses the flannel virtual network.

Enter the following:

$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kubeflannel.yml

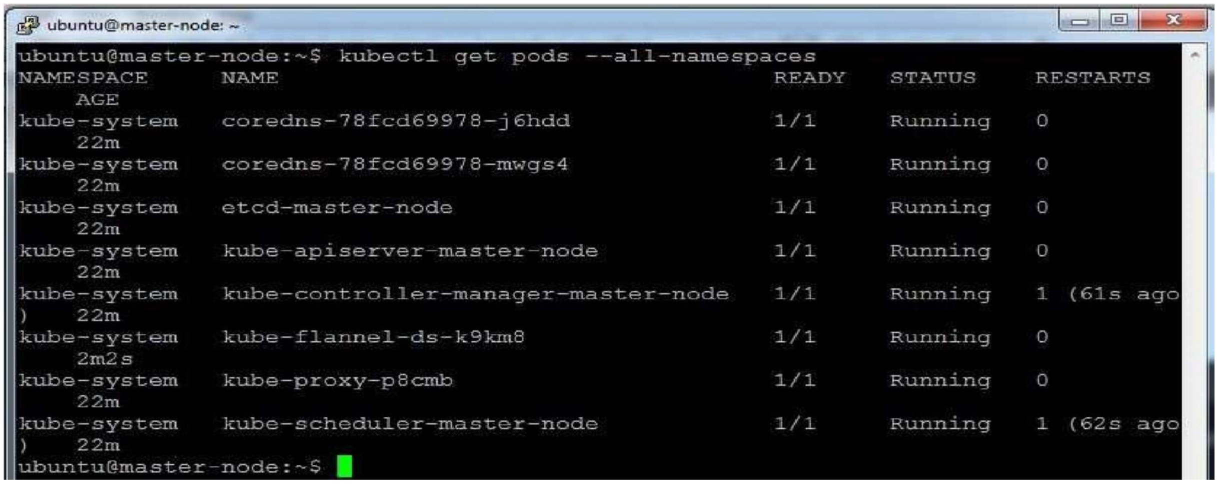


```
ubuntu@master-node:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/
flannel/master/Documentation/kube-flannel.yml
Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable i
n v1.25+
podsecuritypolicy.policy/psp.flannel.unprivileged created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
ubuntu@master-node:~$
```

Allow the process to complete.

Verify that everything is running and communicating:

$ kubectl get pods --all-namespaces



```
ubuntu@master-node:~$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                  READY   STATUS    RESTARTS
    AGE
kube-system   coredns-78fcd69978-j6hdd              1/1     Running   0
    22m
kube-system   coredns-78fcd69978-mwgs4              1/1     Running   0
    22m
kube-system   etcd-master-node                      1/1     Running   0
    22m
kube-system   kube-apiserver-master-node            1/1     Running   0
    22m
kube-system   kube-controller-manager-master-node   1/1     Running   1 (61s ago
)   22m
kube-system   kube-flannel-ds-k9km8                 1/1     Running   0
    2m2s
kube-system   kube-proxy-p8cmb                      1/1     Running   0
    22m
kube-system   kube-scheduler-master-node            1/1     Running   1 (62s ago
)   22m
ubuntu@master-node:~$
```

Step 10 : Join Worker Node to Cluster

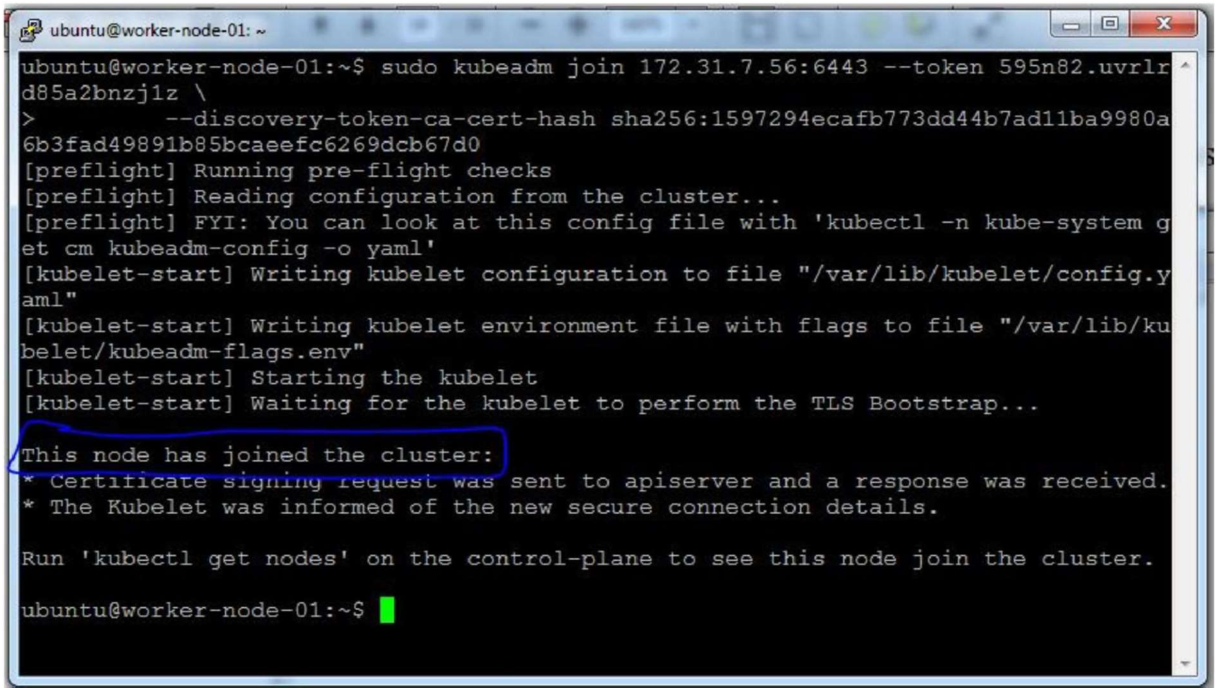
As indicated in Step 7, you can enter the kubeadm join command on each worker node to connect it to the cluster.

Switch to the worker-node-01 system and enter the command you noted from Step 7:

$ kubeadm join --discovery-token abcdef.1234567890abcdef --discovery-token-ca-cert-hash sha256:1234..cdef 1.2.3.4:6443

Replace the alphanumeric codes with those from your master server. Repeat for each worker node on the cluster. Wait a few minutes; then you can check the status of the nodes.

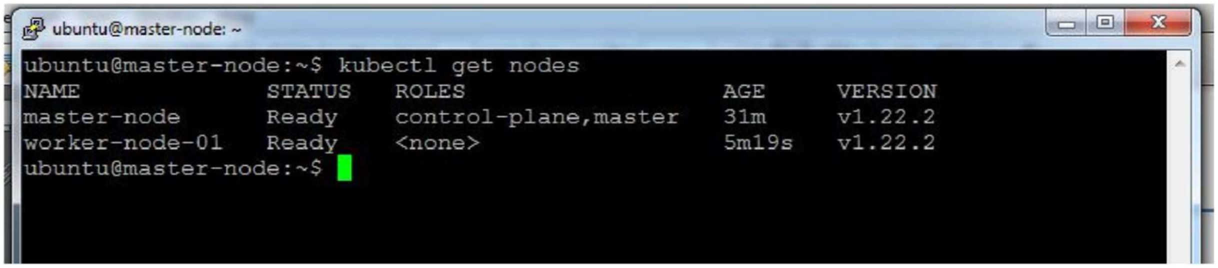
If you are trying to run this on EC2 you'll get an error message saying less CPU and memory to override the error run the above command with --ignore-preflight-errors=all



```
ubuntu@worker-node-01:~$ sudo kubeadm join 172.31.7.56:6443 --token 595n82.uvrlr
d85a2bnzj1z \
>         --discovery-token-ca-cert-hash sha256:1597294ecafb773dd44b7ad11ba9980a
6b3fad49891b85bcaeefc6269dcb67d0
[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 g
et cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.y
aml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/ku
belet/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.

ubuntu@worker-node-01:~$
```

Switch to the master server, and enter:

$ kubectl get nodes

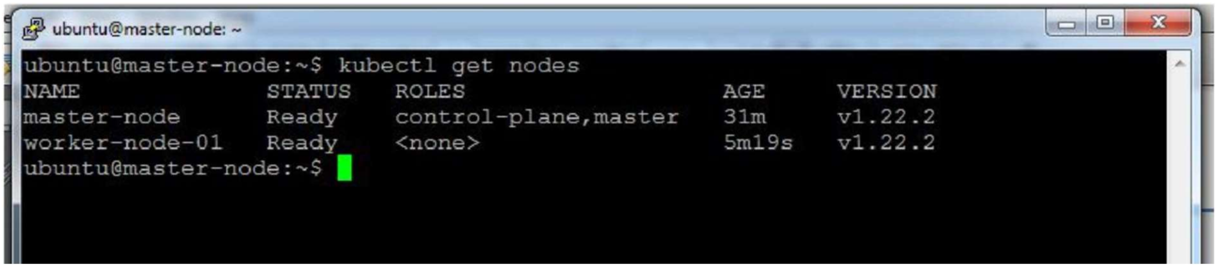


```
ubuntu@master-node:~$ kubectl get nodes
NAME              STATUS   ROLES                  AGE     VERSION
master-node       Ready    control-plane,master   31m     v1.22.2
worker-node-01    Ready    <none>                 5m19s   v1.22.2
ubuntu@master-node:~$
```

The system should display the worker nodes that you joined to the cluster.
How to gracefully rezove a node from Kubernetes?

On Master Node, find the node

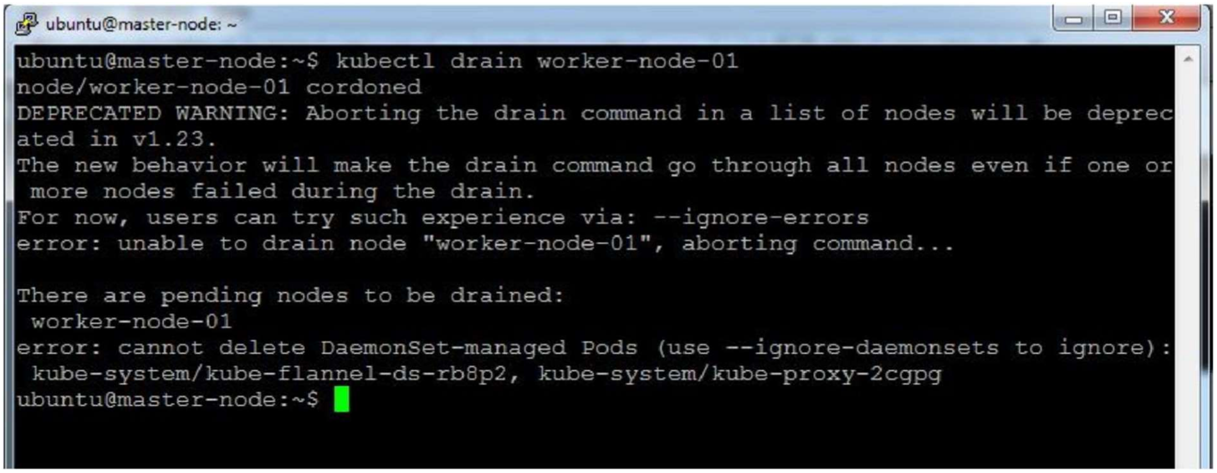
kubernetes-master $ kubectl get nodes



```
ubuntu@master-node:~$ kubectl get nodes
NAME             STATUS   ROLES                  AGE     VERSION
master-node      Ready    control-plane,master   31m     v1.22.2
worker-node-01   Ready    <none>                 5m19s   v1.22.2
ubuntu@master-node:~$
```

Drain it

kubernetes-master $kubectl drain name_of_node



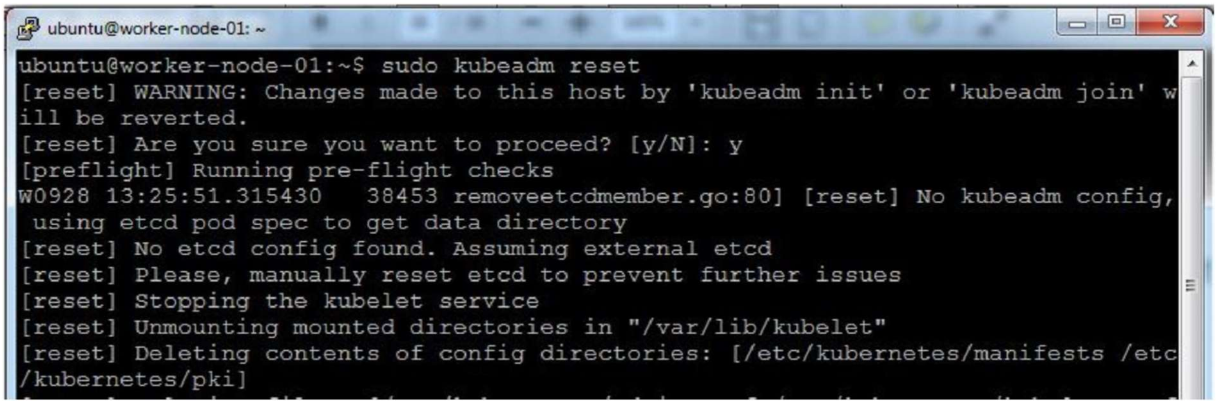
```
ubuntu@master-node:~$ kubectl drain worker-node-01
node/worker-node-01 cordoned
DEPRECATED WARNING: Aborting the drain command in a list of nodes will be deprec
ated in v1.23.
The new behavior will make the drain command go through all nodes even if one or
 more nodes failed during the drain.
For now, users can try such experience via: --ignore-errors
error: unable to drain node "worker-node-01", aborting command...

There are pending nodes to be drained:
 worker-node-01
error: cannot delete DaemonSet-managed Pods (use --ignore-daemonsets to ignore):
 kube-system/kube-flannel-ds-rb8p2, kube-system/kube-proxy-2cgpg
ubuntu@master-node:~$
```

Delete it

kubernetes-master $ kubectl delete node name_of_node

On Worker Node (node-to-be-removed). Remove join/init setting from node

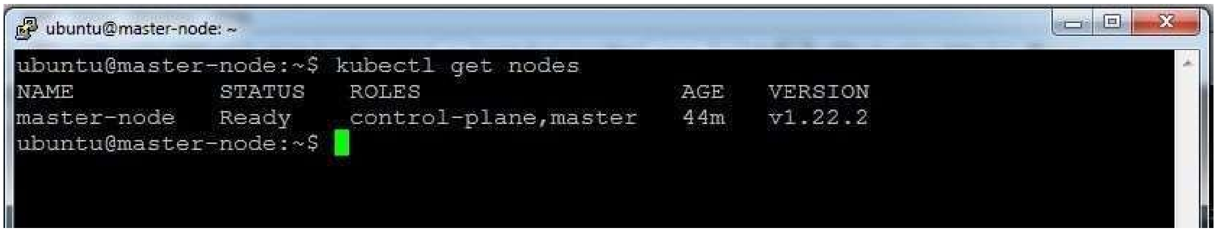
kubernetes-slave $ kubeadm reset



```
ubuntu@worker-node-01:~$ sudo kubeadm reset
[reset] WARNING: Changes made to this host by 'kubeadm init' or 'kubeadm join' w
ill be reverted.
[reset] Are you sure you want to proceed? [y/N]: y
[preflight] Running pre-flight checks
W0928 13:25:51.315430   38453 removeetcdmember.go:80] [reset] No kubeadm config,
 using etcd pod spec to get data directory
[reset] No etcd config found. Assuming external etcd
[reset] Please, manually reset etcd to prevent further issues
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
[reset] Deleting contents of config directories: [/etc/kubernetes/manifests /etc
/kubernetes/pki]
```

Output: all the containers and service related to the kubernetes cluster are deleted.

kubernetes-master $ kubectl get nodes



```
ubuntu@master-node:~$ kubectl get nodes
NAME          STATUS   ROLES                  AGE   VERSION
master-node   Ready    control-plane,master   44m   v1.22.2
ubuntu@master-node:~$
```

The node has been removed from Kubernetes-master.

Conclusion:

The primary strength of Kubernetes is its modularity and generality. Nearly every kind of application that you might want to deploy you can fit within Kubernetes, and no matter what kind of adjustments or tuning you need to make to your system, they're generally possible. Kubernetes is a production-grade container orchestration system that helps you maximize the benefits of using containers. Kubernetes provides you with a toolbox to automate deploying, scaling, and operating containerized applications in production. After following all the steps kubernetes successfully.