



Hands-on Installing Kubernetes

DevOps Training

support@intellipaat.com

+91-7022374614

US: 1-800-216-8930(Toll Free)

Kubernetes Installation

Step 1: Launch 2 instances with the following configuration:

ubuntu 20.04 ami, t2.medium, sg: all traffic.

ubuntu 20.04 ami, t2.micro , sg: all traffic.

Instance state = running

X

Clear filters

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input type="checkbox"/>	K-worker	i-02c2fc5b4a8ce34c4	<div> <div>✓</div> <div>Running</div> <div>🔍</div> </div>	t2.micro	<div> <div>✓</div> <div>2/2 checks passed</div> </div>	No alarms +
<input type="checkbox"/>	K-master	i-0ecc3ba2e3d2ccb2f	<div> <div>✓</div> <div>Running</div> <div>🔍</div> </div>	t2.medium	<div> <div>✓</div> <div>2/2 checks passed</div> </div>	No alarms +

To Install Kubernetes use the following commands:

On Master & worker node

```
sudo su
```

```
apt-get update
```

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

```
service docker restart
```

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

```
echo "deb http://apt.kubernetes.io/ kubernetes-xenial
main" >/etc/apt/sources.list.d/kubernetes.list
```

```
apt-get update
```

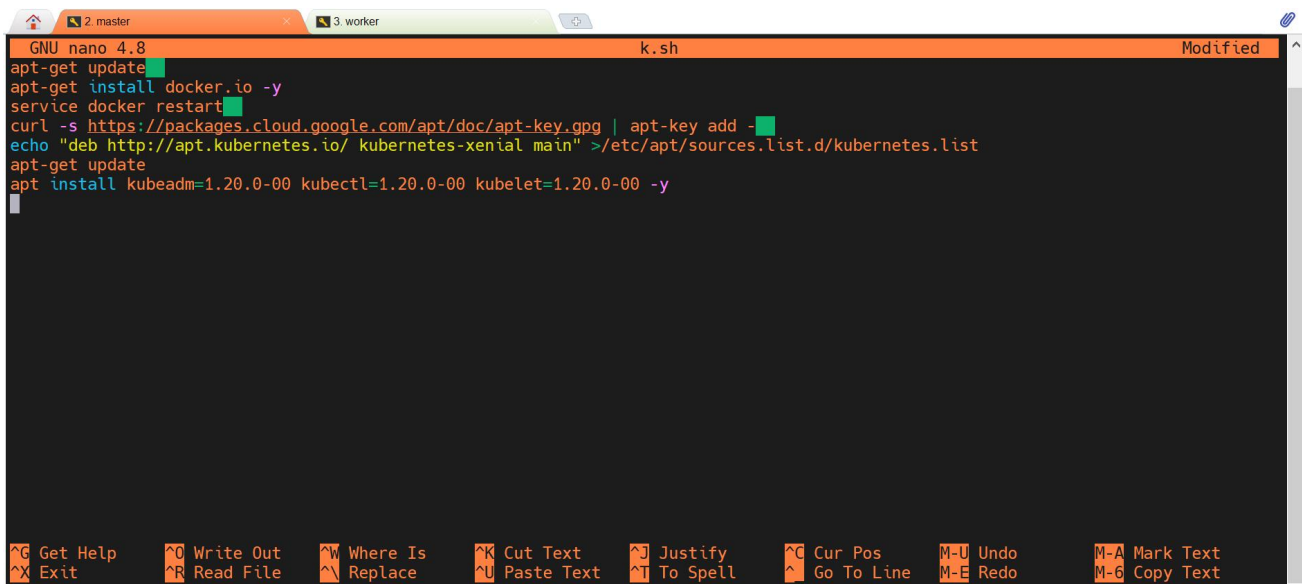
```
apt install kubeadm=1.20.0-00 kubectl=1.20.0-00 kubelet=1.20.0-00 -y
```

Step 2) On both master and worker node run the above command.

2.1) `sudo su`

2.2) create a script file `k.sh`

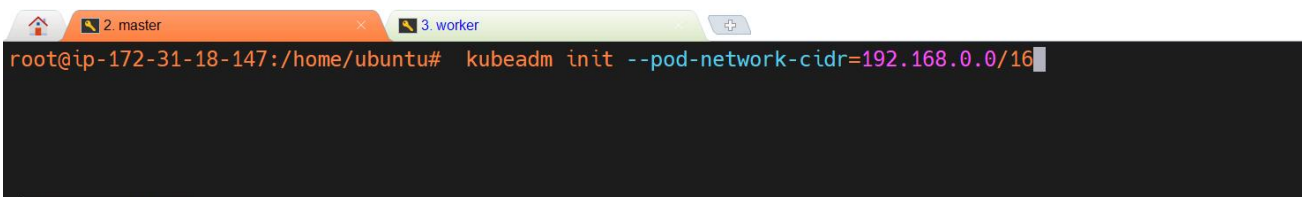
2.3) To execute the script file: `bash k.sh`



```
GNU nano 4.8 k.sh Modified
apt-get update
apt-get install docker.io -y
service docker restart
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" >/etc/apt/sources.list.d/kubernetes.list
apt-get update
apt install kubeadm=1.20.0-00 kubectl=1.20.0-00 kubelet=1.20.0-00 -y
```

On Master:

Step 3) `kubeadm init --pod-network-cidr=192.168.0.0/16`



```
root@ip-172-31-18-147:/home/ubuntu# kubeadm init --pod-network-cidr=192.168.0.0/16
```

```
[bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[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 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
```

Alternatively, if you are the root user, you can run:

```
export KUBECONFIG=/etc/kubernetes/admin.conf
```

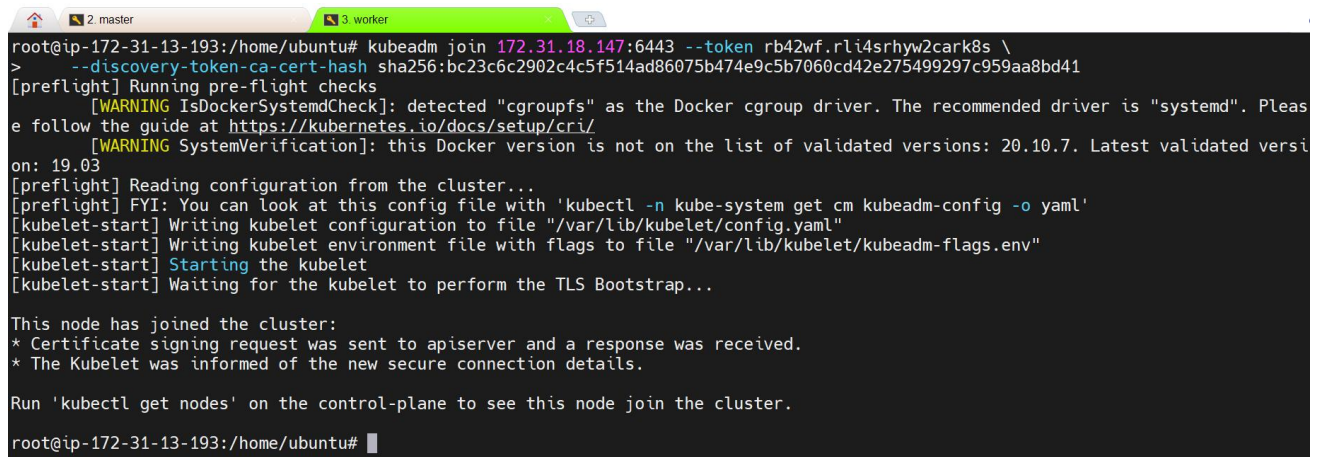
You should now deploy a pod network to the cluster.

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 172.31.18.147:6443 --token rb42wf.rli4srhyw2cark8s \
  --discovery-token-ca-cert-hash sha256:bc23c6c2902c4c5f514ad86075b474e9c5b7060cd42e275499297c959aa8bd41
root@ip-172-31-18-147:/home/ubuntu#
```

Copy the token and paste it into the worker node.



```
root@ip-172-31-13-193:/home/ubuntu# kubeadm join 172.31.18.147:6443 --token rb42wf.rli4srhyw2cark8s \
> --discovery-token-ca-cert-hash sha256:bc23c6c2902c4c5f514ad86075b474e9c5b7060cd42e275499297c959aa8bd41
[preflight] Running pre-flight checks
[WARNING IsDockerSystemdCheck]: detected "cgroups" as the Docker cgroup driver. The recommended driver is "systemd". Please
follow the guide at https://kubernetes.io/docs/setup/cri/
[WARNING SystemVerification]: this Docker version is not on the list of validated versions: 20.10.7. Latest validated versi
on: 19.03
[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 apiservert 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@ip-172-31-13-193:/home/ubuntu#
```

Step 4)

On Master:

```
exit
```

```
mkdir -p $HOME/.kube
```

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

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

support@intellipaat.com - +91-7022374614 - US: 1-800-216-8930 (Toll Free)

Note: In case we want to retrieve the join token use the below-mentioned command.

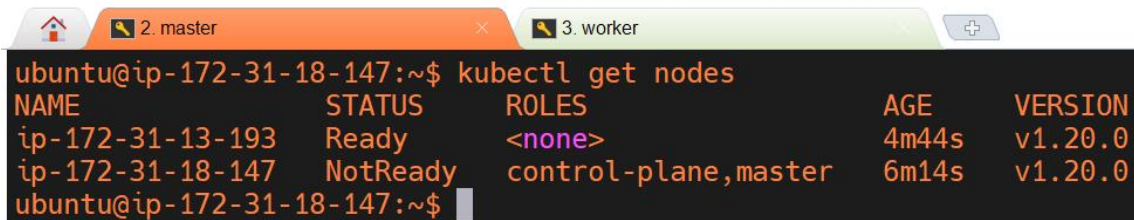
```
kubeadm token create --print-join-command
```

Step 5)

On Master:

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

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.49.0/deploy/static/provider/baremetal/deploy.yaml
```

A terminal window with two tabs: "2. master" (orange) and "3. worker" (green). The terminal shows the command "kubectl get nodes" and its output. The output is a table with columns: NAME, STATUS, ROLES, AGE, and VERSION. The first row shows a node at ip-172-31-13-193 with status "Ready" and no roles. The second row shows a node at ip-172-31-18-147 with status "NotReady" and roles "control-plane,master".

NAME	STATUS	ROLES	AGE	VERSION
ip-172-31-13-193	Ready	<none>	4m44s	v1.20.0
ip-172-31-18-147	NotReady	control-plane,master	6m14s	v1.20.0

Our Kubernetes installation and configuration are complete