

## INSTALLING KUBERNETES

### STEPS FOR BOTH MASTER AND SLAVE

**Step 1:** Create 2 servers on AWS, master and slave with t2.micro configuration and Ubuntu 18.04 OS.

<div> <div>Launch Instance</div> <div>Connect</div> <div>Actions</div> </div>							
<div> <div>Q</div> <div>Filter by tags and attributes or search by keyword</div> </div>							
<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
<input type="checkbox"/>	master	i-0047e2978d43d98...	t2.micro	us-east-2c	running	Initializing	None
<input checked="" type="checkbox"/>	slave	i-04943ff993b2c299b	t2.micro	us-east-2c	running	Initializing	None

**Note:** Sometimes, t2.micro might not work. In that case, please use **t2.medium**, t2.medium is chargeable instance in AWS, please delete all your resources once your practice is done.

**Step 2:** On both these servers, please run the following command:

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

```
[ubuntu@ip-172-31-6-75:~]$ sudo apt-get update
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease [481 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease [71.6 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/universe InRelease [84.0 kB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security InRelease [84.0 kB]
Get:6 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/universe/main amd64 Packages [5,294 kB]
Get:7 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [26.1 kB]
Get:8 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/multiverse/main amd64 Packages [26.1 kB]
Get:9 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [5,294 kB]
Get:10 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [5,294 kB]
Get:11 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [5,294 kB]
Get:12 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [5,294 kB]
Get:13 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [5,294 kB]
```

**Step 3:** Now install docker on both the machines, for that please follow the below link. Skip to the docker section and run all the commands.

**Note:** all these commands have to run as root. To change your use to root, please issue the following command:

```
$ sudo su
```

```
ubuntu@ip-172-31-6-75:~$ sudo su
root@ip-172-31-6-75:/home/ubuntu#
```

Now, goto the following link, skip to the docker section and run the docker installation commands:

<https://kubernetes.io/docs/setup/production-environment/container-runtimes/>

## Docker

On each of your machines, install Docker. Version 19.03.11 is recommended, but 1.13.1, 17.03, 17.06, 17.09, 18.06 and 18.09 are known to work as well. Keep track of the latest verified Docker version in the Kubernetes release notes.

Use the following commands to install Docker on your system:

Ubuntu 16.04+

CentOS/RHEL 7.4+

```
# (Install Docker CE)
```

Only, docker has to be installed. Please skip the CRI-O and containerd commands sections.

**Step 4:** Now, let's go ahead and install Kubernetes run time. Please visit the following link:

<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/#installing-kubeadm-kubelet-and-kubectl>

Now, run the commands mentioned in on the above link. The commands look something like this:

Ubuntu, Debian or HyprIoTOS

CentOS, RHEL or Fedora

Container Linux

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

Please visit the link to copy the commands.

That's it. Your master and slave are now configured with the pre-requisites.

## STEPS FOR MASTER

**Step 1:** Initialize the Kubernetes Master, for doing that please run the following command:

```
$ kubeadm init --apiserver-advertise-address=<Private-IP-of-Master> --pod-network-cidr=192.168.0.0/16
```

The above command, will give you a pre-flight-error, if you are running it on t2.micro. To avoid that error, please add the following flag to the command:

```
$ kubeadm init --apiserver-advertise-address=<Private-IP-of-Master> --pod-network-cidr=192.168.0.0/16 --ignore-preflight-errors=NumCPU
```

```
root@ip-172-31-6-75:/home/ubuntu# kubeadm init --apiserver-advertise-address=172.31.6.75 --pod-network-cidr=192.168.0.0/16 --ignore-preflight-errors=NumCPU
W0619 12:47:56.754749 26632 configset.go:202] WARNING: kubeadm cannot validate component configs for API groups [kubelet.config.k8s.io kubeproxy.config.k8s.io]
[init] Using Kubernetes version: v1.18.4
[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'
```

Now, your kubernetes master is getting initialized. It will take some time. Sometimes, because of internet connectivity, timeout might occur, if that happens simply reset the initialization by using the command, **kubeadm reset**.

And then, re-initialize the master using the previous command.

On successful completion, you will get the following screen. Pay attention, to the highlighted commands.

```
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
```

1.

```
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:
```

2.

```
kubeadm join 172.31.6.75:6443 --token ldchv3.0sky0zdstrnsepg7 \
--discovery-token-ca-cert-hash sha256:f657191c6bb5f320aea931a43e8286ae37d9d93332f165c4bce70a17022a0f33
root@ip-172-31-6-75:/home/ubuntu#
```

The commands highlighted by number 1, are the commands that you have to run as a normal user on your master. To do this, simply type exit, and then run the three commands as in the screenshot below.

```
root@ip-172-31-6-75:/home/ubuntu# exit
exit
ubuntu@ip-172-31-6-75:~$ mkdir -p $HOME/.kube
ubuntu@ip-172-31-6-75:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
ubuntu@ip-172-31-6-75:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-6-75:~$
```

All the kubectl commands will now be running as a normal user. All kubeadm commands should be run as root user.

Command highlighted as 2. Copy it in a notepad, will be required later.

**Step 2:** Now, we will install the pod network Calico. Please visit the following link for the commands:

<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/#pod-network>

Scroll down to the following section, and copy the command and run it on your master.

Calico

Cilium

Contiv-VPP

Kube-router

Weave Net

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, including non-overlay and overlay networks, with or without BGP. Calico uses the same engine to enforce network policy for hosts, pods, and (if using Istio & Envoy) applications at the service mesh layer. Calico works on several architectures, including `amd64`, `arm64`, and `ppc64le`.

Calico will automatically detect which IP address range to use for pod IPs based on the value provided via the `--pod-network-cidr` flag or via kubeadm's configuration.

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



Following is the output that you will get when you run the command on the master.

```
ubuntu@ip-172-31-6-75:~$ kubectl apply -f https://docs.projectcalico.org/v3.14/manifests/calico.yaml
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
serviceaccount/calico-node created
deployment.apps/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
ubuntu@ip-172-31-6-75:~$
```

**Step 3:** Wait for 4-5 minutes, before checking status of all pods using the following command:

```
$ kubectl get po --all-namespaces
```

```
ubuntu@ip-172-31-6-75:~$ kubectl get po --all-namespaces
NAMESPACE      NAME                                                    READY   STATUS    RESTARTS   AGE
kube-system    calico-kube-controllers-76d4774d89-gf77v             1/1     Running   0           11m
kube-system    calico-node-dkq7z                                     1/1     Running   0           11m
kube-system    calico-node-kh5xq                                     1/1     Running   0           11m
kube-system    coredns-66bff467f8-2kr9j                             1/1     Running   0           44m
kube-system    coredns-66bff467f8-598ns                             1/1     Running   0           44m
kube-system    etcd-ip-172-31-6-75                                   1/1     Running   0           44m
kube-system    kube-apiserver-ip-172-31-6-75                         1/1     Running   0           44m
kube-system    kube-controller-manager-ip-172-31-6-75               1/1     Running   0           44m
kube-system    kube-proxy-gfh94                                       1/1     Running   0           44m
kube-system    kube-proxy-x8frf                                       1/1     Running   0           17m
kube-system    kube-scheduler-ip-172-31-6-75                         1/1     Running   0           44m
ubuntu@ip-172-31-6-75:~$
```

## STEPS FOR SLAVE

**Step 1:** Now, let's run the command we copied earlier to the slave as a root user. This will connect our slave to master.

```
root@ip-172-31-11-234:/home/ubuntu# kubeadm join 172.31.6.76:6443 --token ldchv3.0sky0zdstrnsepg/ \
> --discovery-token-ca-cert-hash sha256:f657191c6bb5f320aea931a43e8286ae37d9d93332f165c4bce70a17022a0f33
W0619 13:16:04.983929 22708 join.go:346] [preflight] WARNING: JoinControlPlane.controlPlane settings will be ignored when control
-plane flag is not set.
[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 -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.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.
```

If the connection is successful, you will get the above output!

This concludes, the Kubernetes Installation.