

## BareMetal Kubernetes cluster Setup on AWS Using Kubeadm

- Current Kubernetes Version is **v1.22**
- If you are using bare-metal servers or virtual machines (VMs), **Kubeadm** is a good fit
- If you're running on cloud environments, **kops** and **Kubespray** can ease Kubernetes installation, as well as integration with the cloud providers.
- If you want to drop the burden of managing the Kubernetes control plane, almost all cloud providers have their Kubernetes managed services, such as **Google Kubernetes Engine (GKE)**, **Amazon Elastic Kubernetes Service (EKS)**, **Azure Kubernetes Service (AKS)** etc
- If you just want a playground to study Kubernetes, **Minikube** and **Kind** can help you spin up a Kubernetes cluster in minutes.
- Browser based labs: <https://www.katacoda.com/courses/kubernetes>

### Prerequisites

- A compatible Linux hosts
- 2 GB or more of RAM per machine and 2 CPUs or more
- 4 - Ubuntu Servers
  - 1x Manager (4GB RAM, 2 vCPU) **t2.medium** type
  - 3x Workers (1 GB, 1 Core) **t2.micro** type
- Full network connectivity between all machines in the cluster
- Unique hostname, MAC address for each host. Change hostname of the servers at `/etc/hostname` or using `hostnamectl`. Use **Master** for Master nodes and **worker\_01**, **worker\_02** and so on for worker nodes
- Swap disabled. You **MUST** disable swap in order for the kubelet to work properly
- Certain ports are open on your machines(<https://kubernetes.io/docs/reference/ports-and-protocols/>)

#### ○ Master

Port range	Purpose
<b>6443</b>	These ports are used for Kubernetes API access.
<b>2379-2380</b>	These ports are used for etcd server client API
<b>6783/tcp,6784/udp</b>	for Weavenet CNI
<b>10250</b>	This port is used for Kubelet API
<b>10251</b>	This port is used for kube-scheduler
<b>10252</b>	This port is used for kube-controller-manager
<b>10255</b>	Read-Only Kubelet API
<b>10248</b>	Kubelet health
<b>80</b>	For accessing demo apps
<b>8080</b>	
<b>443</b>	

## Control plane

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10259	kube-scheduler	Self
TCP	Inbound	10257	kube-controller-manager	Self

IP version	Type	Protocol	Port range	Source
IPv4	Custom TCP	TCP	10250 - 10260	0.0.0.0/0
IPv4	Custom TCP	TCP	6443	0.0.0.0/0
IPv4	HTTPS	TCP	443	0.0.0.0/0
IPv6	Custom TCP	TCP	6443	::/0
IPv4	Custom TCP	TCP	30000 - 32767	0.0.0.0/0
IPv6	HTTP	TCP	80	::/0
IPv6	Custom TCP	TCP	10250 - 10260	::/0
IPv6	Custom TCP	TCP	2379 - 2380	::/0
IPv6	Custom TCP	TCP	6783	::/0
IPv4	SSH	TCP	22	0.0.0.0/0
IPv4	HTTP	TCP	80	0.0.0.0/0
IPv6	HTTPS	TCP	443	::/0
IPv6	Custom UDP	UDP	6784	::/0
IPv4	Custom TCP	TCP	6783	0.0.0.0/0
IPv4	Custom TCP	TCP	2379 - 2380	0.0.0.0/0
IPv4	Custom UDP	UDP	6784	0.0.0.0/0
IPv6	Custom TCP	TCP	30000 - 32767	::/0

### Worker

Port range      Purpose

**10250**      This port is used for Kubelet API

**10255**      Read-Only Kubelet API

**30000-32767**      NodePort Services

**6783/tcp,6784/udp** for Weavenet

**80,6443,22,10250-10260,30000-32767**

### Worker node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Services†	All

IP version	Type	Protocol	Port range	Source
IPv6	Custom TCP	TCP	6783	::/0
IPv4	HTTP	TCP	80	0.0.0.0/0
IPv4	Custom TCP	TCP	10250 - 10260	0.0.0.0/0
IPv6	Custom TCP	TCP	10250 - 10260	::/0
IPv6	Custom TCP	TCP	6443	::/0
IPv4	SSH	TCP	22	0.0.0.0/0
IPv4	Custom TCP	TCP	30000 - 32767	0.0.0.0/0
IPv4	Custom UDP	UDP	6784	0.0.0.0/0
IPv4	Custom TCP	TCP	6783	0.0.0.0/0
IPv4	Custom TCP	TCP	6443	0.0.0.0/0
IPv6	Custom UDP	UDP	6784	::/0
IPv6	Custom TCP	TCP	30000 - 32767	::/0
IPv6	HTTP	TCP	80	::/0

Instances (3) <a href="#">Info</a>							
<input type="text" value="Filter instances"/>							
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	Slave2	i-0aaebddcf9dc3a808	<span>Running</span>	t2.micro	-	No alarms	ap-south-1a
<input type="checkbox"/>	Slave1	i-03a481de896c56250	<span>Running</span>	t2.micro	-	No alarms	ap-south-1a
<input type="checkbox"/>	Master	i-0cbf3173fc5cf3f4f	<span>Running</span>	t2.medium	<span>Initializing</span>	No alarms	ap-south-1b

## Common Steps for Master & Workers

- Run all commands as **sudo**
- Install Docker & required packages
 

```
sudo su
apt update -y
apt install -y apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
apt update -y
apt-cache policy docker-ce
apt install -y docker-ce
```
- Configure the Docker daemon, in particular to use systemd for the management of the container's cgroups
 

```
mkdir /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
systemctl enable --now docker
usermod -aG docker ubuntu
systemctl restart docker
```
- Turn off swap space
 

```
swapoff -a
```

- ```
sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```
5. Ensure net.bridge.bridge-nf-call-iptables is set to 1 in your sysctl config (<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/#letting-iptables-see-bridged-traffic>)  
`sysctl net.bridge.bridge-nf-call-iptables=1`
  6. Install kubectl, kubelet and kubeadm  
`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`  
`apt update -y`  
`apt install -y kubelet kubeadm kubectl`
  7. apt-mark hold is used so that these packages will not be updated/removed automatically  
`sudo apt-mark hold kubelet kubeadm kubectl`
  8. Setup kubectl autocompletion  
`echo "source <(kubectl completion bash)" >> ~/.bashrc`

### On Master node

1. Start the cluster using Kubeadm init. This will print a join token. Take backup of that token (<https://www.mankier.com/1/kubeadm-init>)  
`kubeadm config images pull`  
`kubeadm init`
2. Save the kube config to ubuntu's home directory. Switch to ubuntu or type exit from root mode  
`mkdir -p $HOME/.kube`  
`sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config`  
`sudo chown $(id -u):$(id -g) $HOME/.kube/config`
3. Install any CNI plugin  
`kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"`

### On Slave node

1. Copy the join token obtained from `kubeadm init` output to all Workers node and run it. Example  
`kubeadm join \`  
`192.168.56.2:6443 --token ... --discovery-token-ca-cert-hash sha256:....`

### Test the setup

1. On master node, run  
`kubectl get nodes`

### Demo App

```
kubectl run nginx --image=nginx --port=80
kubectl expose pod nginx --port=80 --type=NodePort
```

Go to browser, visit <http://<master-ip>:<NodePort>> to check the nginx default page.  
Make sure the port range 30000-32767 is opened on all/master node

## Get PodCIDR

- `kubectl get nodes -o jsonpath='{.items[*].spec.podCIDR}'`
- `kubectl cluster-info dump | grep -m 1 cluster-cidr`
- 

## Setup Dashboard

- K8dash/Skooner: <https://github.com/skooner-k8s/skooner>

## References

- <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/>
- <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/>
- <https://www.mirantis.com/blog/how-install-kubernetes-kubeadm/>
- <https://www.mankier.com/1/kubeadm-init>
- <https://kubernetes.io/docs/setup/production-environment/container-runtimes/#docker>
- <https://www.weave.works/docs/net/latest/kubernetes/kube-addon/#install>
- <https://github.com/skooner-k8s/skooner>
- <https://www.weave.works/docs/net/latest/kubernetes/kube-addon/#eks>

## Common Errors

```
[kubelet-check] Initial timeout of 40s passed.
[kubelet-check] It seems like the kubelet isn't running or healthy.
[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error:
Get "http://localhost:10248/healthz": dial tcp 127.0.0.1:10248: connect: connection refused.
[kubelet-check] It seems like the kubelet isn't running or healthy.
[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error:
Get "http://localhost:10248/healthz": dial tcp 127.0.0.1:10248: connect: connection refused.
[kubelet-check] It seems like the kubelet isn't running or healthy.
[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error:
Get "http://localhost:10248/healthz": dial tcp 127.0.0.1:10248: connect: connection refused.
[kubelet-check] It seems like the kubelet isn't running or healthy.
[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error:
Get "http://localhost:10248/healthz": dial tcp 127.0.0.1:10248: connect: connection refused.
[kubelet-check] It seems like the kubelet isn't running or healthy.
[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error:
Get "http://localhost:10248/healthz": dial tcp 127.0.0.1:10248: connect: connection refused.
```

```
[kubelet-check] It seems like the kubelet isn't running or healthy.
[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error: Get
"http://localhost:10248/healthz": dial tcp 127.0.0.1:10248: connect: connection refused.
[kubelet-check] It seems like the kubelet isn't running or healthy.
```

[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error: Get "http://localhost:10248/healthz": dial tcp 127.0.0.1:10248: connect: connection refused.

[kubelet-check] It seems like the kubelet isn't running or healthy.

[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error: Get "http://localhost:10248/healthz": dial tcp 127.0.0.1:10248: connect: connection refused.

[kubelet-check] It seems like the kubelet isn't running or healthy.

[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error: Get "http://localhost:10248/healthz": dial tcp 127.0.0.1:10248: connect: connection refused.

[kubelet-check] It seems like the kubelet isn't running or healthy.

[kubelet-check] The HTTP call equal to 'curl -sSL http://localhost:10248/healthz' failed with error: Get "http://localhost:10248/healthz": dial tcp 127.0.0.1:10248: connect: connection refused.

- **kubeadm reset**



DevOps  
Made Easy