K — KUBERNETES

How To Setup Kubernetes Cluster Using Kubeadm

by **Bibin Wilson** · May 18, 2021

Kubeadm is an excellent tool to set up a working kubernetes cluster in less time. It does all the heavy lifting in terms of setting up all kubernetes cluster components. Also, It follows all the configuration best practices for a kubernetes cluster.

In this blog post, I have covered the step-by-step guide to set up a kubernetes cluster using Kubeadm with one master and two worker nodes.

What is Kubeadm?

Kubeadm is a tool to set up a minimum viable Kubernetes cluster without much complex configuration.

Also, Kubeadm makes the whole process easy by running a series of prechecks to ensure that the server has all the essential components and configs to run Kubernetes.

It is developer and maintained by the official Kubernetes community.

There are other options like minikube, kind, etc., that are pretty easy to set up. Those

But if you want to play around with the cluster components or test utilities that are part of cluster administration, Kubeadm is the best option.

Also, you can create a production-like cluster locally on a workstation for development and testing purposes.

Kubeadm Setup Prerequisites

Following are the prerequisites for Kubeadm Kubernetes cluster setup.

1. Minimum two **Ubuntu nodes** [One master and one worker node]. You can have more worker nodes as per your requirement.
2. The master node should have a minimum of **2 vCPU and 2GB RAM**.
3. For the worker nodes, a minimum of 1vCPU and 2 GB RAM is recommended.
4. 10.X.X.X/X network range with static IPs for master and worker nodes. We will be using the 192 series as the pod network range that will be used by the Calico network plugin. Make sure the Node IP range and pod IP range don't overlap.

Kubeadm for Kubernetes Certification Exams

If you are preparing for Kubernetes certifications like **CKA, CKAD, or CKS**, you can use the local kubeadm clusters to practice for the certification exam.

In fact, kubeadm itself part of the CKA and CKS exam. For CKA you might be asked to bootstrap a cluster using Kubeadm. For CKS, you have to upgrade the cluster using kubeadm.

If you use Vagrant-based VMs on your workstation, you can start and stop the cluster whenever you need.

By having the local Kubeadm clusters, you can play around with all the cluster configurations and learn to troubleshoot different components in the cluster.

> **Important Note**: CKA/CKAD/CKS certification cost will increase from July 1st. If you plan for these certifications, make use of the CKA/CKAD/CKS Voucher Codes before the price increases.

Kubeadm Port Requirements

Please refer to the following image and make sure all the ports are allowed for the control plane (master) and the worker nodes. If you set up this on a cloud, make sure you allow the ports in the firewall configuration.

Control-plane node(s)

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	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

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

If you are using vagrant-based Ubuntu VMs, the firewall would be disabled by default. So you don't have to do any firewall configurations.

Vagrantfile, Scripts & Manifests

Also, all the commands used in this guide for master and worker nodes config are hosted as scripts in Github. You can clone the repository for reference.

```
git clone https://github.com/scriptcamp/kubeadm-scripts.git
```

This guide intends to make you understand each config required for the Kubeadm setup. If you don't want to run the command one by one, you can run the script file directly.

If you are using Vagrant to set up the Kubernetes cluster, you can make use of my Vagrantfile. It launches 3 VMs. A self-explanatory basic Vagrantfile.

If you are new to Vagrant, check the Vagrant tutorial.

Kubernetes Cluster Setup Using Kubeadm

Following are the high level steps involved in setting up a Kubernetes cluster using kubeadm.

1. Install container runtime on all nodes- We will be using Docker.
2. Install Kubeadm, Kubelet, and kubectl on all the nodes.
3. Initiate Kubeadm control plane configuration on the master node.
4. Save the node join command with the token.
5. Install the Calico network plugin.
6. Join worker node to the master node (control plane) using the join command.
7. Validate all cluster components and nodes.
8. Install Kubernetes Metrics Server
9. Deploy a sample app and validate the app

All the steps given in this guide are referred from the official Kubernetes documentation and related Github project pages.

Now lets get started with the setup.

Disable swap on all the Nodes

For kubeadm to work properly, you need to disable swap on all the nodes using the following command.

```
sudo swapoff -a
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

The `fstab` entry will make sure the swap if off on system reboots.

You can also, control swap errors using the kubeadm parameter `--ignore-preflight-errors Swap` we will look at it in the latter part.

Install Docker Container Runtime On All The Nodes

The basic requirement for a Kubernetes cluster is a container runtime. You can have any one of the following container runtimes.

1. containerd

We will be using Docker for this setup.

As a first step, we need to install Docker on all the nodes. Execute the following commands on all the nodes.

Install the required packages for Docker.

```
sudo apt-get update -y
sudo apt-get install -y \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

Add the Docker GPG key and apt repository.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg

echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null
```

Install the Docker community edition.

```
sudo apt-get update -y
sudo apt-get install docker-ce docker-ce-cli containerd.io -y
```

Add the docker daemon configurations to use systemd as the cgroup driver.

```
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
```

Start and enable the docker service.

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

Install Kubeadm & Kubelet & Kubectl on

Install the required dependencies.

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

Add the GPG key and apt repository.

```
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
```

Update apt and install kubelet, kubeadm and kubectl.

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

Note: If you want to install a specific version of kubernetes, you can specify the version as shown below.

```
sudo apt-get install -y kubelet=1.20.6-00 kubectl=1.20.6-00 kubeadm=1.20.6-00
```

Add hold to the packages to prevent upgrades.

```
sudo apt-mark hold kubelet kubeadm kubectl
```

Now we have all the required utilities and tools for configuring Kubernetes components using kubeadm.

Initialize Kubeadm On Master Node To Setup Control Plane

Execute the commands in this section only on the master node.

First, set two environment variables. Replace `10.0.0.10` with the IP of your master node.

```
IPADDR="10.0.0.10"
NODENAME=$(hostname -s)
```

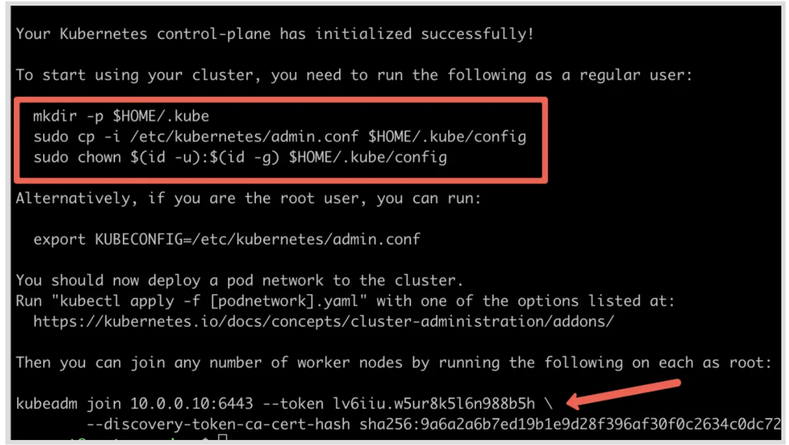
Now, initialize the master node control plane configurations using the following

```
sudo kubeadm init --apiserver-advertise-address=$IPADDR  --apiserver-cert-extra-sans=$IPADDR  --pod-network-cidr=192.168.0.0/16 --node-name $NODENAME --ignore-preflight-errors Swap
```

`--ignore-preflight-errors Swap` is actually not required as we disabled the swap initially. I just added it for the safer side.

> **Note:** You can also pass the kubeadm configs as a file when initializing the cluster. See Kubeadm Init with config file

On a successful kubeadm initialization you should get an output with kubeconfig file location and the join command with the token as shown below. Copy that and save it to the file. we will need it for joining the worker node to the master.



Use the following commands from the output to create the `kubeconfig` in master so that you can use `kubectl` to interact with cluster API.

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

Now, verify the kubeconfig by executing the following kubectl command to list all the pods in the `kube-system` namespace.

```
kubectl get po -n kube-system
```

You should see the following output. You will see the two Coredns pods in a pending state. It is the expected behavior. Once we install the network plugin, it will be in a

```
vagrant@master-node:~$ kubectl get po -n kube-system
NAME                                   READY   STATUS    RESTARTS   AGE
coredns-558bd4d5db-88z28               0/1     Pending   0          9m54s
coredns-558bd4d5db-c9gr4               0/1     Pending   0          9m54s
etcd-master-node                       1/1     Running   0          10m
kube-apiserver-master-node             1/1     Running   0          10m
kube-controller-manager-master-node    1/1     Running   0          10m
kube-proxy-lh5bv                       1/1     Running   0          9m54s
kube-scheduler-master-node             1/1     Running   0          10m
vagrant@master-node:~$
```

Note: You can copy the `admin.conf` file from the master to your workstation in $HOME/.kube/config location if you want to execute `kubectl` commands from the workstation.

By default, apps won't get scheduled on the master node. If you want to use the master node for scheduling apps, taint the master node.

```
kubectl taint nodes --all node-role.kubernetes.io/master-
```

Install Calico Network Plugin for Pod Networking

Kubeadm does not configure any network plugin. You need to install a network plugin of your choice.

I am using Calico network plugin for this setup.

Make sure you execute the kubectl command from where you have configured the `kubeconfig` file. Either from the master of your workstation with the connectivity to the kubernetes API.

Execute the following command to install the calico network plugin on the cluster.

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

Now if you check the pods in kube-system namespace, you will see calico pods and running CoreDNS pods.

```
vagrant@master-node:~$ kubectl get po -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
calico-kube-controllers-b656ddcfc-rptcc    1/1     Running   0          4m21s
calico-node-x2mxd                          1/1     Running   0          4m21s
coredns-558bd4d5db-88z28                   1/1     Running   0          42m
coredns-558bd4d5db-c9gr4                   1/1     Running   0          42m
etcd-master-node                           1/1     Running   0          42m
kube-apiserver-master-node                 1/1     Running   0          42m
kube-controller-manager-master-node        1/1     Running   0          42m
kube-proxy-lh5bv                           1/1     Running   0          42m
kube-scheduler-master-node                 1/1     Running   0          42m
```

Join Worker Nodes To Kubernetes Master Node

We have setup Docker, kubelet and kubeadm utilities on the worker nodes as well.

Now, lets join the worker node to the master node using the Kubeadm join command you have got in the output while setting up the master node.

If you missed the join command, execute the following command in the master node to recreate the token with the join command.

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

Here is how the command looks like. Use `sudo` if you running as a normal user.

```
sudo kubeadm join 10.128.0.37:6443 --token j4eice.33vgvgyf5cxw4u8i \
    --discovery-token-ca-cert-hash
sha256:37f94469b58bcc8f26a4aa44441fb17196a585b37288f85e22475b00c36f1c61
```

Now execute the kubectl command to check if the node is added to the master.

```
kubectl get nodes
```

Example output,

```
vagrant@master-node:~$ kubectl get nodes
NAME            STATUS     ROLES                  AGE   VERSION
master-node     Ready      control-plane,master   55m   v1.21.0
worker-node01   NotReady   <none>                 6s    v1.21.0
```

You can further add more nodes with the same join command.

Setup Kubernetes Metrics Server

To verify this, if you run the top command, you will see the `Metrics API not available` error.

```
vagrant@master-node:~$ kubectl top nodes
W0510 08:09:17.831030   28821 top_node.go:119] Using json format to get metrics.
Next release will switch to protocol-buffers, switch early by passing --use-
protocol-buffers flag
error: Metrics API not available
vagrant@master-node:~$
```

To install the metrics server, execute the following metric server manifest file. It deploys metrics server version v0.4.4

```
kubectl apply -f https://raw.githubusercontent.com/scriptcamp/kubeadm-
scripts/main/manifests/metrics-server.yaml
```

This manifest is taken from the official metrics server repo. I have added the `--kubelet-insecure-tls` flag to the container to make it work in the local setup and hosted it separately.

Once the metrics server objects are deployed, it takes a minute for you to see the node and pod metrics using the top command.

```
kubectl top nodes
```

You should be able to view the node metrics as shown below.

```
vagrant@master-node:/vagrant$ kubectl top nodes
NAME           CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
master-node    161m         8%     1578Mi          41%
worker-node01  53m          5%     739Mi           39%
```

You can also view the pod CPU and memory metrics using the following command.

```
kubectl top pod -n kube-system
```

Deploy A Sample Nginx Application

Now that we have all the components to make the cluster and applications work, let's deploy a sample Nginx application and see if we can access it over a NodePort

Create an Nginx deployment. Execute the following directly on the command line.

```
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
EOF
```

Expose the Nginx deployment on a NodePort 32000

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 32000
EOF
```


Once the deployment is up, you should be able to access Nginx home page on the allocated NodePort.

For example,

Possible Kubeadm Issues

Following are the possible issues you might encounter in kubeadm setup.

2. **Nodes cannot connect to Master:** Check the firewall between nodes and make sure all the nodes can talk to each other on the required kubernetes ports.
3. **Calico Pod Restarts:** Sometimes, if you use the same IP range for the node and pod network, Calico pods may not work as expected. So make sure the node and pod IP ranges don't overlap. Overlapping IP addresses could result in issues for other applications running on the cluster as well.

Conclusion

As a DevOps engineer, it is good to have an understanding of the Kubernetes cluster components. With companies using managed Kubernetes services, we miss learning the basic building blocks to kubernetes.

This Kubeadm setup is good for learning and playing around with kubernetes.

Also, there are many other Kubeadm configs that I did not cover in this guide as it is out of the scope of this guide. Please refer to the official Kubeadm documentation.

By having the whole cluster setup in VMs, you can learn all the cluster components configs and troubleshoot the cluster on component failures.

Also, with vagrant, you can create simple automation to bring up and tear down Kubernetes clusters on-demand in your local workstation.

Bibin Wilson

An author, blogger and DevOps practitioner. In spare time, he loves to try out the latest open source technologies. He works as an Associate Technical Architect

VIEW COMMENTS (0)

YOU MAY ALSO LIKE

K — KUBERNETES

CKA Exam Study Guide: A Complete Resource For CKA Aspirants

by **Shishir Khandelwal** · June 25, 2021

This CKA Exam study guide will help you prepare for the Certified Kubernetes Administrator (CKA) exam with all...

K — KUBERNETES

Kubernetes Certification Coupon: 18% Off + $100 Off On Kubernetes Course bundle

by **devopscube** · July 20, 2021

The cloud-native foundation has three Kubernetes certifications. It is one of the most preferred certifications for organizations due to...

K — KUBERNETES

How to Setup Ingress on GKE using GKE Ingress Controller

by **devopscube** · June 24, 2021

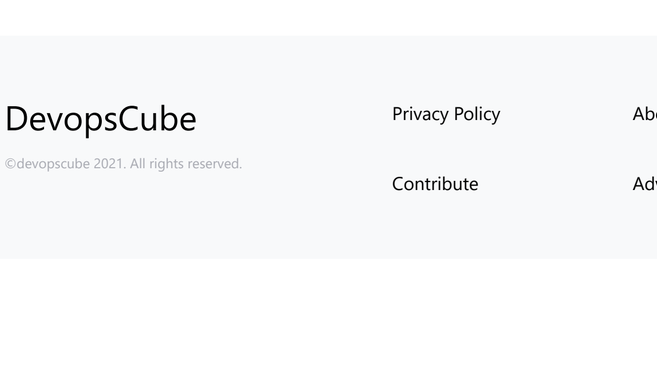
This tutorial will guide you to setup Ingress on GKE using a GKE ingress controller that covers: The...

D — DEVOPS

Helm Tutorial: How To Install and

by **Bibin Wilson** · May 26, 2021

This post explains steps to install helm 3 on kubernetes and installing helm charts for managing and deploying...



C — CI/CD

How To Setup Jenkins On Kubernetes Cluster – Beginners Guide

by **Bibin Wilson** · May 9, 2021

Hosting Jenkins on a Kubernetes cluster is beneficial for Kubernetes-based deployments and dynamic container-based scalable Jenkins agents. In...

K — KUBERNETES

CKAD Exam Study Guide: A Complete Resource for CKAD Aspirants

by **Shishir Khandelwal** · June 28, 2021

This CKAD Exam study guide will help you prepare for the Certified Kubernetes Developer (CKAD) exam with all...