Kubernetes Lab Setup: Step-by-Step Guide

Table of Contents

1. Introduction

2. Prerequisites

3. Environment Setup

4. Installing Docker

5. Installing Kubernetes Components

6. Initializing the Kubernetes Control Plane

7. Joining Worker Nodes to the Cluster

8. Deploying a Sample Application

9. Exploring Basic Kubernetes Commands

10. Conclusion

---

1. Introduction

Kubernetes is an open-source platform designed to automate deploying, scaling, and operating containerized applications. Setting up a lab environment will help you understand how Kubernetes works and allow you to experiment freely.

---

2. Prerequisites

Hardware:

At least 2 machines (virtual or physical): one for the control plane (master node) and one or more worker nodes.

Each machine should have:

2 CPUs or more

2 GB RAM or more

10 GB of free disk space

Operating System:

Ubuntu 20.04 LTS (or a similar Linux distribution)

User Privileges:

sudo privileges on all machines

Network:

All nodes can communicate over the network.

Internet access to download packages.

---

3. Environment Setup

a. Prepare Virtual Machines (If Using VMs)

You can use virtualization software like VirtualBox or VMware to create virtual machines.

Install Virtualization Software:

Download and install VirtualBox or VMware Workstation Player.

Create VMs:

Create one VM for the control plane and at least one VM for worker nodes.

Allocate resources as per the hardware prerequisites.

b. Update System Packages

On all nodes, update the package lists:

sudo apt update && sudo apt upgrade -y

---

4. Installing Docker

Docker is the container runtime Kubernetes uses to run containers.

a. Install Docker on All Nodes

1. Uninstall Old Versions (if any):

sudo apt remove docker docker-engine docker.io containerd runc

2. Install Required Packages:

sudo apt install -y apt-transport-https ca-certificates curl gnupg lsb-release

3. Add Docker’s Official GPG Key:

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

4. Set Up the Stable Repository:

echo \

"deb [arch=$(dpkg --print-architecture) 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

5. Install Docker Engine:

sudo apt update

sudo apt install -y docker-ce docker-ce-cli containerd.io

6. Verify Docker Installation:

sudo docker run hello-world

b. Configure Docker

1. Create or Modify Docker Daemon Configuration:

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

2. Restart Docker:

sudo systemctl enable docker

sudo systemctl daemon-reload

sudo systemctl restart docker

---

5. Installing Kubernetes Components

Install kubeadm, kubelet, and kubectl on all nodes.

a. Add Kubernetes Signing Key

sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg \

https://packages.cloud.google.com/apt/doc/apt-key.gpg

b. Add Kubernetes 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

c. Install Kubernetes Packages

sudo apt update

sudo apt install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

---

6. Initializing the Kubernetes Control Plane

Perform the following steps only on the control plane node.

a. Disable Swap

sudo swapoff -a

sudo sed -i '/ swap / s/^.\*$/#\1/g' /etc/fstab

b. Initialize Kubernetes Cluster

sudo kubeadm init --pod-network-cidr=192.168.0.0/16

Note: The --pod-network-cidr parameter specifies the CIDR for the pod network. This must match the network plugin you plan to use.

c. Set Up Local kubectl Configuration

mkdir -p $HOME/.kube

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

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

d. Deploy a Pod Network

We'll use Calico as the network plugin.

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

Alternatively, for Flannel:

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

e. Verify the Control Plane Status

kubectl get nodes

The control plane node should be listed as Ready.

---

7. Joining Worker Nodes to the Cluster

Perform the following steps on each worker node.

a. Disable Swap

sudo swapoff -a

sudo sed -i '/ swap / s/^.\*$/#\1/g' /etc/fstab

b. Join the Cluster

Use the kubeadm join command provided after initializing the control plane. If you didn't save it, you can get it again by running the following on the control plane node:

kubeadm token create --print-join-command

Example Join Command:

sudo kubeadm join <control-plane-host>:6443 --token <token> \

--discovery-token-ca-cert-hash sha256:<hash>

Replace <control-plane-host>, <token>, and <hash> with the appropriate values.

c. Verify Nodes Are Joined

On the control plane node, run:

kubectl get nodes

All nodes should be listed and show Ready status.

---

8. Deploying a Sample Application

Let's deploy a simple Nginx web server to test the cluster.

a. Create a Deployment

kubectl create deployment nginx --image=nginx

b. Expose the Deployment

kubectl expose deployment nginx --port=80 --type=NodePort

c. Get Service Details

kubectl get services

Note the PORT(S) under the nginx service. It will be in the format 80:<NodePort>/TCP.

d. Access the Application

Open a web browser and navigate to http://<worker-node-IP>:<NodePort>.

Replace <worker-node-IP> with the IP address of any worker node.

Replace <NodePort> with the port number obtained in the previous step.

You should see the default Nginx welcome page.

---

9. Exploring Basic Kubernetes Commands

a. Get Cluster Information

kubectl cluster-info

b. List All Nodes

kubectl get nodes -o wide

c. Describe a Node

kubectl describe node <node-name>

d. List All Pods

kubectl get pods -o wide

e. Describe a Pod

kubectl describe pod <pod-name>

f. Scaling Deployments

Scale Up:

kubectl scale deployment nginx --replicas=3

Verify Pods:

kubectl get pods -o wide

g. Rolling Updates

Update the Deployment Image:

kubectl set image deployment/nginx nginx=nginx:1.19

Check Rollout Status:

kubectl rollout status deployment/nginx

---

10. Conclusion

Congratulations! You've successfully set up a Kubernetes lab environment. You can now explore Kubernetes features, deploy applications, and practice cluster management.

---

Additional Tips

Cleaning Up Resources:

Delete Deployment and Service:

kubectl delete service nginx

kubectl delete deployment nginx

Troubleshooting:

Check Pod Logs:

kubectl logs <pod-name>

Get Events:

kubectl get events --sort-by=.metadata.creationTimestamp

---

Further Learning

Official Kubernetes Documentation: https://kubernetes.io/docs/home/

Interactive Tutorials: Katacoda Kubernetes Tutorials

Kubernetes for Beginners Playlist: YouTube - KodeKloud

---

Feel free to ask if you have any questions or need further assistance!