Introduction

A Kubernetes cluster is a powerful tool for managing containers in a highly-available manner. Kubeadm greatly simplifies the process of setting up a simple cluster. In this hands-on lab, you will build your own working Kubernetes cluster using Kubeadm.

Install Docker on all three nodes

Do the following on all three nodes:

1. Add the Docker GPG key:

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

1. Add the Docker repository:

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

1. Update packages:

sudo apt-get update

1. Install Docker:

sudo apt-get install -y docker-ce=18.06.1~ce~3-0~ubuntu

1. Hold Docker at this specific version:

sudo apt-mark hold docker-ce

1. Verify that Docker is up and running with:

sudo systemctl status docker

Make sure the Docker service status is active (running)!

Install Kubeadm, Kubelet, and Kubectl on all three nodes

Install the Kubernetes components by running this on all three nodes:

1. Add the Kubernetes GPG key:

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

1. Add the Kubernetes repository:

sudo bash -c "cat <<EOF >/etc/apt/sources.list.d/kubernetes.list

deb https://apt.kubernetes.io/ kubernetes-xenial main

EOF"

1. Update packages:

sudo apt-get update

1. Install kubelet, kubeadm, and kubectl:

sudo apt-get install -y kubelet kubeadm kubectl

1. Hold the Kubernetes components at this specific version:

sudo apt-mark hold kubelet kubeadm kubectl

Bootstrap the cluster on the Kube master node

1. On the Kube master node, do this:

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

That command may take a few minutes to complete.

1. When it is done, set up the local kubeconfig:
2. mkdir -p $HOME/.kube
3. sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

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

Take note that the kubeadm init command printed a long kubeadm join command to the screen. You will need that kubeadm join command in the next step!

1. Run the following command on the Kube master node to verify it is up and running:

kubectl version

This command should return both a Client Version and a Server Version.

Join the two Kube worker nodes to the cluster

1. Copy the kubeadm join command that was printed by the kubeadm init command earlier, with the token and hash. Run this command on both worker nodes, but make sure you add sudo in front of it:

sudo kubeadm join $some\_ip:6443 --token $some\_token --discovery-token-ca-cert-hash $some\_hash

1. Now, on the Kube master node, make sure your nodes joined the cluster successfully:

kubectl get nodes

Verify that all three of your nodes are listed. It will look something like this:

NAME STATUS ROLES AGE VERSION

ip-10-0-1-101 NotReady master 30s v1.12.2

ip-10-0-1-102 NotReady <none> 8s v1.12.2

ip-10-0-1-103 NotReady <none> 5s v1.12.2

Note that the nodes are expected to be in the NotReady state for now.

Set up cluster networking with flannel

1. Turn on iptables bridge calls on all three nodes:

echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf

sudo sysctl -p

1. Next, run this only on the Kube master node:

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

Now flannel is installed! Make sure it is working by checking the node status again:

kubectl get nodes

After a short time, all three nodes should be in the Ready state. If they are not all Ready the first time you run kubectl get nodes, wait a few moments and try again. It should look something like this:

NAME STATUS ROLES AGE VERSION

ip-10-0-1-101 Ready master 85s v1.12.2

ip-10-0-1-102 Ready <none> 63s v1.12.2

ip-10-0-1-103 Ready <none> 60s v1.12.2

## Creating a New Token

1. Using the kubeadm command, list your current tokens on the Master node. If your cluster was initialized over 24-hour ago, the list will likely be empty, since a token’s lifespan is only 24-hours.

kubeadm tokens list

1. Create a new token using kubeadm. By using the –print-join-command argument kubeadm will output the token and SHA hash required to securely communicate with the master.

kubeadm token create --print-join-command

Joining the New Worker to the Cluster

With our new token in hand it is time to join our worker to the cluster.

1. Using SSH, log onto the new worker node.
2. Use the kubeadm join command with our new token to join the node to our cluster.

kubeadm join 192.168.1.130:6443 --token qt57zu.wuvqh64un13trr7x --discovery-token-ca-cert-hash sha256:5ad014cad868fdfe9388d5b33796cf40fc1e8c2b3dccaebff0b066a0532e8723

1. List your cluster’s nodes to verify your new worker has successfully joined the cluster.

kubectl get nodes

ubuntu@ip-172-31-44-57:~$ kubectl label node ip-172-31-45-255 node-role.kubernetes.io/worker=worker2

node/ip-172-31-45-255 labeled

ubuntu@ip-172-31-44-57:~$ kubectl get nodes

NAME STATUS ROLES AGE VERSION

ip-172-31-44-57 Ready master 84m v1.12.7

ip-172-31-44-73 Ready worker 81m v1.12.7

ip-172-31-45-255 Ready worker 6m1s v1.12.7

root@ip-172-31-41-82:~# kubectl get nodes -o wide

NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-RUNTIME

ip-172-31-4-141 Ready worker 37m v1.12.7 172.31.4.141 <none> Ubuntu 16.04.6 LTS 4.4.0-1087-aws docker://18.6.1

ip-172-31-41-82 Ready master 38m v1.12.7 172.31.41.82 <none> Ubuntu 16.04.6 LTS 4.4.0-1087-aws docker://18.6.1

root@ip-172-31-41-82:~# kubectl get nodes -o wide --show-labels

NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-RUNTIME LABELS

ip-172-31-4-141 Ready worker 7d1h v1.12.7 172.31.4.141 <none> Ubuntu 16.04.6 LTS 4.4.0-1092-aws docker://18.6.1 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/hostname=ip-172-31-4-141,node-role.kubernetes.io/worker=worker1

ip-172-31-41-82 Ready master 7d1h v1.12.7 172.31.41.82 <none> Ubuntu 16.04.6 LTS 4.4.0-1092-aws docker://18.6.1 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/hostname=ip-172-31-41-82,node-role.kubernetes.io/master=