Building a Kubernetes Cluster

Relevant Documentation

- Installing kubeadm
- Creating a cluster with kubeadm

Lesson Reference

If you are using cloud playground, create three servers with the following settings:

- Distribution: Ubuntu 20.04 Focal Fossa LTS
- Size: medium

If you wish, you can set an appropriate hostname for each node.

On the control plane node:

```
sudo hostnamectl set-hostname k8s-control
```

On the first worker node:

```
sudo hostnamectl set-hostname k8s-worker1
```

On the second worker node:

```
sudo hostnamectl set-hostname k8s-worker2
```

On all nodes, set up the hosts file to enable all the nodes to reach each other using these hostnames.

```
sudo vi /etc/hosts
```

On all nodes, add the following at the end of the file. You will need to supply the actual **private IP address** for each node.

```
<control plane node private IP> k8s-control
<worker node 1 private IP> k8s-worker1
<worker node 2 private IP> k8s-worker2
```

Log out of all three servers and log back in to see these changes take effect.

On all nodes, set up containerd. You will need to load some kernel modules and modify some system settings as part of this process.

```
cat << EOF | sudo tee /etc/modules-load.d/containerd.conf
overlay
br_netfilter
EOF

sudo modprobe overlay

sudo modprobe br_netfilter

cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF

sudo sysctl --system
```

Install and configure containerd.

```
sudo apt-get update && sudo apt-get install -y containerd

sudo mkdir -p /etc/containerd

sudo containerd config default | sudo tee /etc/containerd/config.toml

sudo systemctl restart containerd
```

On all nodes, disable swap.

```
sudo swapoff -a
```

On all nodes, install kubeadm, kubelet, and kubectl.

```
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=1.24.0-00 kubeadm=1.24.0-00 kubectl=1.24.0-00

sudo apt-mark hold kubelet kubeadm kubectl
```

On the control plane node only, initialize the cluster and set up kubectl access.

```
sudo kubeadm init --pod-network-cidr 192.168.0.0/16 --kubernetes-version 1.24.0

mkdir -p $HOME/.kube

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

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

Verify the cluster is working.

```
kubectl get nodes
```

Install the Calico network add-on.

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

Get the join command (this command is also printed during `kubeadm init`. Feel free to simply copy it from there).

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

Copy the join command from the control plane node. Run it on each worker node as root (i.e. with `sudo`).

```
sudo kubeadm join ...
```

On the control plane node, verify all nodes in your cluster are ready. Note that it may take a few moments for all of the nodes to enter the `READY` state.

```
kubectl get nodes
```