The first step in setting up a new cluster is to install a container runtime such as Docker. In this lesson, we will be installing Docker on our three servers in preparation for standing up a Kubernetes cluster. After completing this lesson, you should have three playground servers, all with Docker up and running.

Here are the commands used in this lesson:

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

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

sudo apt-get update

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

sudo apt-mark hold docker-ce
```

You can verify that docker is working by running this command:

```
sudo docker version
```

Now that Docker is installed, we are ready to install the Kubernetes components. In this lesson, I will guide you through the process of installing Kubeadm, Kubelet, and Kubectl on all three playground servers. After completing this lesson, you should be ready for the next step, which is to bootstrap the cluster.

Here are the commands used to install the Kubernetes components in this lesson. Run these on all three servers.

NOTE: There are some issues being reported when installing version 1.12.2-00 from the Kubernetes ubuntu repositories. You can work around this by using version 1.52.7-00 for kubelet, kubeadm, and kubectl.

```
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=1.15.7-00 kubeadm=1.15.7-00 kubectl=1.15.7-00

sudo apt-mark hold kubelet kubeadm kubectl
```

After installing these components, verify that Kubeadm is working by getting the version info.

```
kubeadm version
```

Later versions disable swap:
https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.8.md#before-upgrading

Now we are ready to get a real Kubernetes cluster up and running! In this lesson, we will bootstrap the cluster on the Kube master node. Then, we will join each of the two worker nodes to the cluster, forming an actual multi-node Kubernetes cluster.

Here are the commands used in this lesson:

- On the Kube master node, initialize the cluster:

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

That command may take a few minutes to complete.

- When it is done, set up the local kubeconfig:

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

- Verify that the cluster is responsive and that Kubectl is working:

```
kubectl version
```

You should get `Server Version` as well as `Client Version`. It should look something like this:

```
Client Version: version.Info{Major:"1", Minor:"12", GitVersion:"v1.12.2", GitCommit:"17c77c7898218073f14c8d57
Server Version: version.Info{Major:"1", Minor:"12", GitVersion:"v1.12.2", GitCommit:"17c77c7898218073f14c8d57
```

- The `kubeadm init` command should output a `kubeadm join` command containing a token and hash. Copy that command and run it with `sudo` on both worker nodes. It should look something like this:

```
sudo kubeadm join $some_ip:6443 --token $some_token --discovery-token-ca-cert-hash $some_hash
```

- Verify that all nodes have successfully joined the cluster:

```
kubectl get nodes
```

You should see all three of your nodes listed. It should look something like this:

```
NAME                      STATUS     ROLES    AGE     VERSION
wboyd1c.mylabserver.com   NotReady   master   5m17s   v1.12.2
wboyd2c.mylabserver.com   NotReady   <none>   53s     v1.12.2
wboyd3c.mylabserver.com   NotReady   <none>   31s     v1.12.2
```

Note: The nodes are expected to have a STATUS of `NotReady` at this point.

Once the Kubernetes cluster is set up, we still need to configure cluster networking in order to make the cluster fully functional. In this lesson, we will walk through the process of configuring a cluster network using Flannel. You can find more information on Flannel at the official site: https://coreos.com/flannel/docs/latest/.

Here are the commands used in this lesson:

- On all three nodes, run the following:

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

- Install Flannel in the cluster by running this only on the Master node:

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

- Verify that all the nodes now have a STATUS of `Ready`:

```
kubectl get nodes
```

You should see all three of your servers listed, and all should have a STATUS of `Ready`. It should look something like this:

```
NAME                      STATUS    ROLES    AGE     VERSION
wboyd1c.mylabserver.com   Ready     master   5m17s   v1.12.2
wboyd2c.mylabserver.com   Ready     <none>   53s     v1.12.2
wboyd3c.mylabserver.com   Ready     <none>   31s     v1.12.2
```

Note: It may take a few moments for all nodes to enter the `Ready` status, so if they are not all `Ready`, wait a few moments and try again.

- It is also a good idea to verify that the Flannel pods are up and running. Run this command to get a list of system pods:

```
kubectl get pods -n kube-system
```

You should have three pods with `flannel` in the name, and all three should have a status of `Running`.