Create a Kubernetes cluster with kubeadm

> The Kubernetes version used in this guide is `1.23.1`
>
> Reference
>
> - https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/
> - https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/

This guide demonstrates how to create a Kubernetes cluster using kubeadm. The cluster consists of one master node and two worker nodes. You have to provision three VMs with Ubuntu installed and docker as the container runtime installed.

1. Letting iptables see bridged traffic

Apply below changes to all VM nodes.

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system
```

2. Installing kubeadm, kubelet and kubectl

Install kubeadm, kubelet, and kubectl in all VM nodes.

2.1 Update the apt package index and install packages needed to use the Kubernetes apt repository

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
```

2.2 Download Aliyun public signing key for Kubernetes

```
sudo curl -s https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg | sudo apt-key add
-
```

2.3 Add the Aliyun Kubernetes apt repository

```
sudo tee /etc/apt/sources.list.d/kubernetes.list <<-'EOF'
deb https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial main
EOF
```

2.4 Update apt package index, install kubelet, kubeadm and kubectl, and pin their version

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

3. Configure cluster

3.1 Initialize the control-plane node

- Use `registry.aliyuncs.com/google_containers` as the image repository
- Specify the Kubernetes version to be installed: `1.23.1`
- Specify the CIDR to `192.168.0.0/16`
    - It is required by Container Network Interface(CNI) based Pod network add-on so that your Pods can communicate with each other.
    - In our case, we use `Calico` as the network addon.
- Specify the API Server Advertise Address, it is configured using the master node IP address.

```
kubeadm init \
 --image-repository registry.aliyuncs.com/google_containers \
 --kubernetes-version v1.23.1 \
 --pod-network-cidr=192.168.0.0/16 \
 --apiserver-advertise-address=192.168.56.10
```

When the control-plane node is successfully created, **take a note of the output**

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.56.10:6443 --token iu3r27.idfhx25w952fcjhm \
  --discovery-token-ca-cert-hash
sha256:66bb30b4f8fcbb2ea1141806d3b90b4409c893649845461e089f4d9cba05ce22
```

Then Copy `kubeconfig` File.

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

3.2 Add worker nodes to the cluster

- SSH to the worker nodes

  ```
  ssh sadmin@192.168.56.11
  ssh sadmin@192.168.56.12
  ```

- Become root

```
sudo su -
```

- Run the command that was output by `kubeadm init`

```
kubeadm join --token <token> <control-plane-host>:<control-plane-port> --discovery-token-ca-cert-hash sha256:<hash>

kubeadm join 192.168.56.10:6443 --token iu3r27.idfhx25w952fcjhm \
  --discovery-token-ca-cert-hash
sha256:66bb30b4f8fcbb2ea1141806d3b90b4409c893649845461e089f4d9cba05ce22
```

> If you forgot the token, in master node, run `kubeadm token list` or run `kubeadm token create --print-join-command`

- After we added these two worker nodes, we can verify by running below command in the **master node**. It should list three nodes at the moment.

```
root@k8smaster:~# kubectl get nodes
NAME          STATUS     ROLES                  AGE    VERSION
k8smaster     NotReady   control-plane,master   5m4s   v1.23.1
k8sworker1    NotReady   <none>                 41s    v1.23.1
k8sworker2    NotReady   <none>                 20s    v1.23.1
```

3.3 Configure CNI Calico Plugin

> Reference
>
> https://projectcalico.docs.tigera.io/getting-started/kubernetes/quickstart

- Download the YAML file locally.

```
mkdir calico
cd calico

# use wget
wget https://docs.projectcalico.org/manifests/tigera-operator.yaml
wget https://docs.projectcalico.org/manifests/custom-resources.yaml

# alternatively, use curl
curl -O https://docs.projectcalico.org/manifests/tigera-operator.yaml
curl -O https://docs.projectcalico.org/manifests/custom-resources.yaml
```

- [Optional] Preload Image:

- Sometimes the images from `k8s.gcr.io` and Red Hat `quay.io` are difficult to pull, it's better we can pull these images before we install the CNI addon.
- We can verify the images to be used by Calico in the spec file.

```
# use image: to see the output, only one image is required.
grep image: tigera-operator.yaml

                    image:
          image: quay.io/tigera/operator:v1.23.3
```

- We can choose to pull the image beforehand.

```
# Pull this image beforehand in all VMs
docker pull quay.io/tigera/operator:v1.23.3
```

- Install Calico

```
# You may need to change the default IP pool CIDR to match your pod network CIDR.
# Install the Tigera Calico operator and custom resource definitions.
kubectl create -f tigera-operator.yaml

# Install Calico by creating the necessary custom resource.
kubectl create -f custom-resources.yaml
```

- Confirm that all of the pods relating to Calico are `running`

```
watch kubectl get pods -n calico-system
```

- We can check node status by running below command in the **master node**. All of the three nodes should be in `Ready` state.

```
root@k8smaster:~/calico# kubectl get nodes
NAME         STATUS   ROLES                  AGE   VERSION
k8smaster    Ready    control-plane,master   65m   v1.23.1
k8sworker1   Ready    <none>                 61m   v1.23.1
k8sworker2   Ready    <none>                 61m   v1.23.1
```

4. Extra Cluster Settings

- Set auto complete for `kubectl`
  - Prerequisite is `bash-completion` is installed

```
source <(kubectl completion bash)
echo "source <(kubectl completion bash)" >> /etc/.bashrc
```

- Set alias for `kubectl`

```
alias k=kubectl
echo 'alias k=kubectl' >> /etc/.bashrc
complete -F __start_kubectl k

alias ks='kubectl -n kube-system'
echo 'alias ks="kubectl -n kube-system"' >> /etc/.bashrc
```

- Update Vim

```
vi /root/.vimrc
# add below setting
set paste
```