QA Consulting

Kubernetes kubeadm

TRAINING MATERIALS - MODULE HANDOUT

Contacts

team.qac.all.trainers@qa.com

www.consulting.qa.com

Contents

Overview 2

Tasks 2

- Overview . . . 2
- Container Network Interface Plugins (CNI) . . . 2
- Kubelet Container Runtime Interface (CRI) . . . 2
- kubeadm, kubelet Service & kubectl . . . 3
- Install Dependencies . . . 3
- Initialise Cluster . . . 3
- Calico . . . 4
- Allow Pods on the Master Node . . . 4
- Configure non-root User for Docker and Kubectl access . . . 4

Overview

Kubeadm is a very helpful tool that allows us to bootstrap a Kubernetes cluster. It also can allow us to perform upgrades and downgrades on the Kubernetes cluster in place. The simplicity of this tool allows us to apply it to various use cases such as:

- New users wanting to try out Kubernetes
- Testing application
- As a tool for much larger projects creating multi-node clusters

Tasks

Overview

This exercise will take you through the installation of a simple single-node Kubernetes cluster using **kubeadm**. The benefit of installing Kubernetes cluster as opposed to using a managed solution, is that you get a much better understanding of some of the inner workings of Kubernetes.

Requirements

- Root access to a Linux Machine
- 2 CPUs, 4GB or more RAM

Container Network Interface Plugins (CNI)

CNI is a specification and set of libraries for configuring network interfaces on Linux containers. The method we are going to be using for installing Kubernetes here (kubeadm) requires the network to be configured with a CNI.

```
CNI_VERSION="v0.6.0"
mkdir -p /opt/cni/bin
curl -L "https://github.com/containernetworking/plugins/releases/download/${CNI_VERSION}/
    cni-plugins-amd64-${CNI_VERSION}.tgz" | tar -C /opt/cni/bin -xz
```

Kubelet Container Runtime Interface (CRI)

Inside a Kubernetes node, at the lowest level exists a software that starts and stop the containers, among other things as well. The most popular container runtime is Docker, there are many other great runtimes about though. CRI has been introduced for Kubernetes to support these various runtimes. In our case, we'll be installing it so that the kubelet service can interface with the Docker container runtime.

```
CRICTL_VERSION="v1.11.1"
mkdir -p /opt/bin
curl -L "https://github.com/kubernetes-incubator/cri-tools/releases/download/${
    CRICTL_VERSION}/crictl-${CRICTL_VERSION}-linux-amd64.tar.gz" | tar -C /opt/bin -xz
```

kubeadm, kubelet Service & kubectl

kubeadm
Kubeadm is the tool that automates the configurations of our cluster and creates it.
kubectl
The way we can interface with Kubernetes is via an API, which is very versatile for any application to be able to connect to. The client we are using to connect to the API and issue instructions is the kubectl CLI tool.
kubelet
The kubelet is a service that's constantly running to maintain the desired state of our deployments, pods and services in Kubernetes.

```
RELEASE="$(curl␣-sSL␣https://dl.k8s.io/release/stable.txt)"

mkdir -p /opt/bin

cd /opt/bin

curl -L --remote-name-all https://storage.googleapis.com/kubernetes-release/release/${
    RELEASE}/bin/linux/amd64/{kubeadm,kubelet,kubectl}

chmod +x {kubeadm,kubelet,kubectl}

curl -sSL "https://raw.githubusercontent.com/kubernetes/kubernetes/${RELEASE}/build/debs/
    kubelet.service" | sed "s:/usr/bin:/opt/bin:g" > /etc/systemd/system/kubelet.service

mkdir -p /etc/systemd/system/kubelet.service.d

curl -sSL "https://raw.githubusercontent.com/kubernetes/kubernetes/${RELEASE}/build/debs
    /10-kubeadm.conf" | sed "s:/usr/bin:/opt/bin:g" > /etc/systemd/system/kubelet.service.
    d/10-kubeadm.conf

systemctl enable --now kubelet
```

Install Dependencies

We are going to be using the Docker container runtime, so we will of course need to have that installed. Kubeadm utilises the **ebtables** and **socat** CLI tools on Linux and will show warnings if they are not present in the preflight check. For a successfully cluster initialisation we need to get these installed beforehand.

```
apt install -y ebtables socat

curl https://get.docker.com | bash
```

Initialise Cluster

We can now start creating the cluster with kubeadm. We'll add the downloaded binaries to our **PATH** environment variable so we can access them. A network will be specified for the pods when we initialise the cluster, this is so the CNI plugin that we are going to use (Calico) will function, more on this in the next section.

```
PATH=$PATH:/opt/bin:/opt/cni/bin
```

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

Calico

Calico is the CNI plugin that will be using. Kubernetes is a container orchestration tool, of which is does a good job. However Kubernetes does not manage the network for communication between Pods, this is why we need a CNI, such as Calico. There are other choices available such as Flannel and Weave Net. We'll be using kubectl to apply Calico configurations.

```
export KUBECONFIG=/etc/kubernetes/admin.conf

kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/
    installation/hosted/rbac-kdd.yaml
kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/
    installation/hosted/kubernetes-datastore/calico-networking/1.7/calico.yaml
```

Allow Pods on the Master Node

For security reasons, the cluster that we have just created will not schedule Pods to be created on the master Node by default. Because we only have a single-node cluster at this point, we are going to change this.

```
kubectl taint nodes --all node-role.kubernetes.io/master-
```

Configure non-root User for Docker and Kubectl access

We'd like to be able to use our own user for managing Docker containers and the Kubernetes cluster. To do this, we can copy the admin kube config to our user folder and add our user to the docker group.

```
sudo mkdir ~/.kube
sudo cp /etc/kubernetes/admin.conf ~/.kube/config
sudo chown ${USER}:${USER} ~/.kube/config
echo 'PATH=$PATH:/opt/bin:/opt/cni/bin' >> ~/.bashrc
sudo usermod -aG docker ${USER}
```