Installation LAB

16 December 2020 17:12

ISO URL - http://centos.excellmedia.net/7.9.2009/isos/x86_64/CentOS-7-x86_64-Minimal-2009.iso

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

- **Pre-requisite**

  - 2 GB or more of RAM per machine (any less will leave little room for your apps)
  - 2 CPUs or more
  - Full network connectivity between all machines in the cluster (public or private network is fine)
  - Unique hostname, MAC address, and product_uuid for every node. See here for more details.
    - You can get the MAC address of the network interfaces using the command `ip link` or `ifconfig -a`
    - The product_uuid can be checked by using the command `sudo cat /sys/class/dmi/id/product_uuid` **(VM)**

  - Certain ports are open on your machines. See here for more details.
  - Swap disabled. You **MUST** disable swap in order for the kubelet to work properly.

From <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/>

As a requirement for your Linux Node's iptables to correctly see bridged traffic, you should ensure net.bridge.bridge-nf-call-iptables is set to 1 in your sysctl config, e.g.

ON ALL YOUR MACHINES (VMs)

Prepare the VMs (Master & Worker):

There are a few things you need to do to get the VM ready. Specifically, you need to turn off swap, tweak some configuration settings, and make sure you have the prerequisites libraries installed. To do that, follow these steps:

1. Networking (Assign IP address [static])
   ```
   # nmcli -p dev              # List network devices
   # nmtui edit enps03
           - 192.168.1.76   g/w 192.168.1.1
   ```
2. Change to root:
   ```
   sudo su
   ```
3. Set hostname and add entries into /etc/hosts
   a. Master:
   ```
   # hostnamectl set-hostname k8s_master
   ```
   ```
   [root@k8s-master ~]# cat /etc/hosts
   192.168.1.73 k8s-master
   192.168.1.74 worker1
   ```
   b. Workers
   ```
   # hostnamectl set-hostname worker1
   ```
   ```
   # hostnamectl set-hostname worker2
   ```
   ```
   [root@worker1 ~]# cat /etc/hosts
   192.168.1.73 k8s-master
   192.168.1.74 worker1
   ```
4. Turn off swap: To do this, you will first need to turn it off directly …
   ```
   swapoff -a
   ```
   … then comment out the reference to swap in /etc/fstab. Start by editing the file:
   a.
   ```
   vi /etc/fstab
   ```
5. Letting iptables see bridged traffic
   - Make sure that the br_netfilter module is loaded.
     ```
     [root@localhost ~]# lsmod | grep br_netfilter
     [root@localhost ~]# modprobe br_netfilter
     [root@localhost ~]# lsmod | grep br_netfilter
     br_netfilter          22256  0
     bridge              151336  2 br_netfilter,ebtable_broute
     ```
   - Disable firewalld [optional - else we need to open specific ports]
     ```
     [root@k8s-master ~]# systemctl stop firewalld
     [root@k8s-master ~]# systemctl disable firewalld
     Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.
     Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
     ```
   - As a requirement for your Linux Node's iptables to correctly see bridged traffic, you should ensure net.bridge.bridge -nf-call-iptables is set to 1 in your sysctl
     ```
     cat <<EOF | sudo teenet.bridge.bridge-nf-call-ip6tables = 1
     net.bridge.bridge-nf-call-iptables = 1
      /etc/sysctl.d/k8s.conf
     EOF
     sudo sysctl --system
     ```
6. Reboot so the changes take effect [optional]

Installing runtime (docker/CRI-O/Containerd)

From <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/>

```
# (Install Docker CE)
## Set up the repository
```

### Install required packages

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

## Add the Docker repository

```
sudo yum-config-manager --add-repo \
  https://download.docker.com/linux/centos/docker-ce.repo
```

# Install Docker CE

```
sudo yum update -y && sudo yum install -y \
  containerd.io-1.2.13 \
  docker-ce-19.03.11 \
  docker-ce-cli-19.03.11
```

## Create /etc/docker

```
sudo mkdir /etc/docker
```

# Set up the Docker daemon

```
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2",
  "storage-opts": [
    "overlay2.override_kernel_check=true"
  ]
}
EOF
```

# Create /etc/systemd/system/docker.service.d

```
sudo mkdir -p /etc/systemd/system/docker.service.d
```

# Restart Docker

```
sudo systemctl daemon-reload
sudo systemctl restart docker
If you want the docker service to start on boot, run the following command:
sudo systemctl enable docker
```

Installing kubeadm, kubelet and kubectl

- Add repository

```
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
exclude=kubelet kubeadm kubectl
EOF
```

- Set SELinux to permissive

```
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

- Start kubelet service

```
sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
sudo systemctl enable --now kubelet
```

Master Node

```
[root@k8s_master ~]# docker image ls
REPOSITORY   TAG      IMAGE ID   CREATED   SIZE
[root@k8s_master ~]# kubeadm config images pull
```

```
[root@k8s_master ~]# kubeadm config images pull >/dev/null 2>&1
[root@k8s_master ~]# docker image ls
REPOSITORY                          TAG        IMAGE ID       CREATED         SIZE
k8s.gcr.io/kube-proxy               v1.20.1    e3f6fcd87756   6 days ago      118MB
k8s.gcr.io/kube-apiserver           v1.20.1    75c7f7112080   6 days ago      122MB
k8s.gcr.io/kube-controller-manager  v1.20.1    2893d78e47dc   6 days ago      116MB
k8s.gcr.io/kube-scheduler           v1.20.1    4aa0b4397bbb   6 days ago      46.4MB
k8s.gcr.io/etcd                     3.4.13-0   0369cf4303ff   3 months ago    253MB
k8s.gcr.io/coredns                  1.7.0      bfe3a36ebd25   6 months ago    45.2MB
k8s.gcr.io/pause                    3.2        80d28bedfe5d   10 months ago   683kB
```

--apiserver-advertise-address	The IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--image-repository	Choose a container registry to pull control plane images from (default "k8s.gcr.io")
--kubernetes-version	Choose a specific Kubernetes version for the control plane. (default "stable-1")
--pod-network-cidr	Specify range of IP addresses for the pod network. If set, the control plane will automatically allocate CIDRs for every node.
--service-cidr	Use alternative range of IP address for service VIPs. (default "10.96.0.0/12")

```
# kubeadm init --apiserver-advertise-address=<Master_IP> --pod-network-cidr=192.168.0.0/16
```

Master IP = Internal IP Address

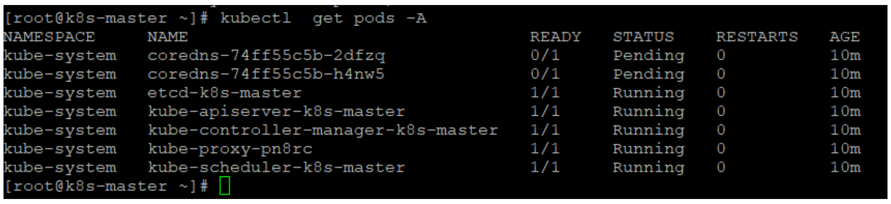
```
Your Kubernetes control-plane has initialized successfully!

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 192.168.1.73:6443 --token clrxr9.qelzztg0zns24hwu \
   --discovery-token-ca-cert-hash sha256:31a30fdb81a7ff3bcd0c17234b0da37b10b700c5fcfa0e3f2bf2f8e478322e85
```

Note: Currently Calico is the only CNI plugin that the kubeadm project performs e2e tests against. If you find an issue related to a CNI plugin you should log a ticket in its respective issue tracker instead of the kubeadm or kubernetes issue trackers.

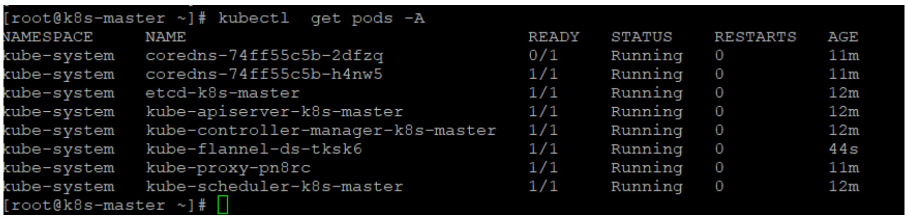
From <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/#pod-network>

POD Network



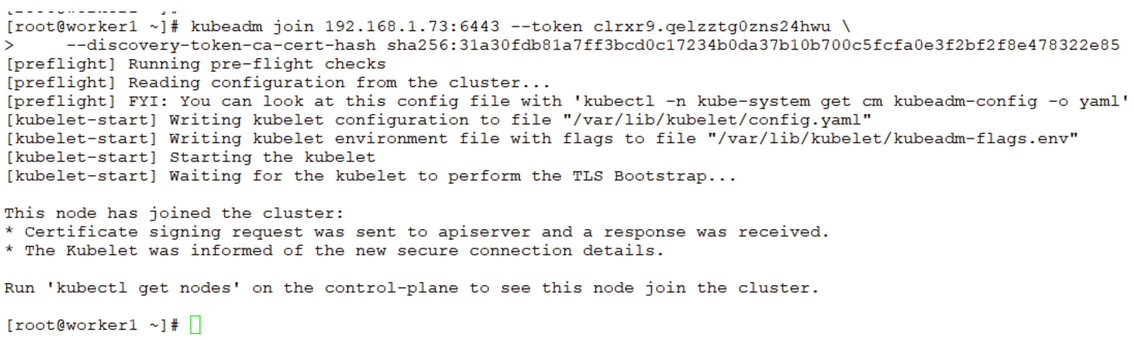
```
[root@k8s-master ~]# kubectl  get pods -A
NAMESPACE     NAME                                 READY   STATUS    RESTARTS   AGE
kube-system   coredns-74ff55c5b-2dfzq              0/1     Pending   0          10m
kube-system   coredns-74ff55c5b-h4nw5              0/1     Pending   0          10m
kube-system   etcd-k8s-master                      1/1     Running   0          10m
kube-system   kube-apiserver-k8s-master            1/1     Running   0          10m
kube-system   kube-controller-manager-k8s-master   1/1     Running   0          10m
kube-system   kube-proxy-pn8rc                     1/1     Running   0          10m
kube-system   kube-scheduler-k8s-master            1/1     Running   0          10m
[root@k8s-master ~]#
```

```
[root@k8s-master ~]# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
podsecuritypolicy.policy/psp.flannel.unprivileged created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

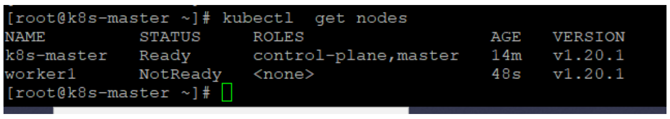


```
[root@k8s-master ~]# kubectl  get pods -A
NAMESPACE     NAME                                 READY   STATUS    RESTARTS   AGE
kube-system   coredns-74ff55c5b-2dfzq              0/1     Running   0          11m
kube-system   coredns-74ff55c5b-h4nw5              1/1     Running   0          11m
kube-system   etcd-k8s-master                      1/1     Running   0          12m
kube-system   kube-apiserver-k8s-master            1/1     Running   0          12m
kube-system   kube-controller-manager-k8s-master   1/1     Running   0          12m
kube-system   kube-flannel-ds-tksk6                1/1     Running   0          44s
kube-system   kube-proxy-pn8rc                     1/1     Running   0          11m
kube-system   kube-scheduler-k8s-master            1/1     Running   0          12m
[root@k8s-master ~]#
```

Worker Nodes



```
[root@worker1 ~]# kubeadm join 192.168.1.73:6443 --token clrxr9.qelzztg0zns24hwu \
>     --discovery-token-ca-cert-hash sha256:31a30fdb81a7ff3bcd0c17234b0da37b10b700c5fcfa0e3f2bf2f8e478322e85
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

[root@worker1 ~]#
```



```
[root@k8s-master ~]# kubectl  get nodes
NAME         STATUS     ROLES                  AGE   VERSION
k8s-master   Ready      control-plane,master   14m   v1.20.1
worker1      NotReady   <none>                 48s   v1.20.1
[root@k8s-master ~]#
```