Kubernetes cluster

26 March 2025 14:26

To configure a **Kubernetes cluster** with **one master node and two worker nodes** on **RHEL 8**, follow these steps:

Prerequisites

- Three RHEL 8 instances (1 master, 2 workers)
- Minimum hardware requirements:
  - **Master Node:** 2 CPUs, 4GB RAM
  - **Worker Nodes:** 2 CPUs, 2GB RAM each
  - At least 20GB disk space on each node
- Root or sudo access
- A stable internet connection

Step 1: Configure Hostnames & Hosts File

Perform these steps **on all nodes**:

1.1 Update the System

```
sudo dnf update -y
sudo dnf install -y vim git wget curl bash-completion
```

1.2 Set Unique Hostnames

Run this command on the **master node**:

```
sudo hostnamectl set-hostname master-node
```

Run this command on **worker nodes**:

```
sudo hostnamectl set-hostname worker-node1  # First worker
sudo hostnamectl set-hostname worker-node2  # Second worker
```

1.3 Configure /etc/hosts File

Edit the file on **all nodes**:

```
sudo vim /etc/hosts
```

Add the following lines (adjust IP addresses accordingly):

```
192.168.1.100 master-node
192.168.1.101 worker-node1
192.168.1.102 worker-node2
```

Step 2: Disable SELinux, Swap, and Configure Firewall

Perform these steps **on all nodes**:

2.1 Disable SELinux

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

2.2 Disable Swap

```
sudo swapoff -a
sudo sed -i '/swap/d' /etc/fstab
```

2.3 Enable IP Forwarding

```
sudo modprobe br_netfilter

echo '1' | sudo tee /proc/sys/net/bridge/bridge-nf-call-iptables
```

```
echo '1' | sudo tee /proc/sys/net/ipv4/ip_forward
```

Make it persistent:

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF

sudo sysctl --system
```

2.4 Open Firewall Ports

On the **master node**:

```
sudo firewall-cmd --permanent --add-port=6443/tcp
sudo firewall-cmd --permanent --add-port=2379-2380/tcp
sudo firewall-cmd --permanent --add-port=10250/tcp
sudo firewall-cmd --permanent --add-port=10257/tcp
sudo firewall-cmd --permanent --add-port=10259/tcp
sudo firewall-cmd --reload
```

On the **worker nodes**:

```
sudo firewall-cmd --permanent --add-port=10250/tcp
sudo firewall-cmd --permanent --add-port=30000-32767/tcp
sudo firewall-cmd --reload
```

Step 3: Install Docker and Kubernetes Components

Perform this on **all nodes**.

3.1 Install Docker

```
sudo dnf config-manager --add-repo=https://download.docker.com/linux/centos/docker-ce.repo
sudo dnf install -y docker-ce docker-ce-cli containerd.io
sudo systemctl enable --now docker
```

Verify Docker installation:

```
docker --version
```

3.2 Install Kubernetes (Kubeadm, Kubelet, and Kubectl)

```
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.29/rpm/
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.29/rpm/repodata/repomd.xml.key
EOF

sudo dnf install -y kubelet kubeadm kubectl
sudo systemctl enable --now kubelet
```

Step 4: Initialize the Kubernetes Cluster (On Master Node Only)

Run this on the **master node**:

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

Copy and save the kubeadm join command output; you will need it for the worker nodes.

4.1 Set Up kubectl for the Master Node

```
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 status:

```
kubectl get nodes
```

Step 5: Install a Network Plugin (CNI)

Kubernetes requires a networking plugin for pod communication. Install **Calico**:

```
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/calico.yaml
```

Verify that the pods in the kube-system namespace are running:

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

Step 6: Join Worker Nodes to the Cluster

On **each worker node**, use the kubeadm join command output from Step 4. If you lost it, retrieve it from the master:

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

Example join command (Run on worker nodes):

```
sudo kubeadm join 192.168.1.100:6443 --token <TOKEN> --discovery-token-ca-cert-hash sha256:<HASH>
```

Verify the nodes have joined:

```
kubectl get nodes
```

Step 7: Deploy a Test Application

Test the cluster by deploying an Nginx pod:

```
kubectl create deployment nginx --image=nginx
kubectl expose deployment nginx --port=80 --type=NodePort
```

Check the service:

```
kubectl get svc nginx
```

Access the application using any worker node's IP and the **NodePort**.

Conclusion

You now have a **Kubernetes cluster with one master and two worker nodes** running on **RHEL 8**. Let me know if you need further assistance! 🚀

Note: Prepared by - K.T. Harsha . Images are taken from the web site, used only for training and understanding purpose