For this, we will walk-through a multi-node Kubernetes cluster installation on RHEL 8. This tutorial is command-line based so you will need access to your terminal window . Will be performing the steps on GCP.

Starting from RHEL 8, docker has now natively been replaced by podman and buildah which are tools from Redhat. As a matter of fact, the docker package has now been removed from the default package repository and at the end we all are working with the same API , then we don’t have to lock-in into a specific tool.

These are the popular Container runtimes and being used mainly

* [**Docker**](https://kubernetes.io/docs/setup/production-environment/container-runtimes/#docker)
* [**CRI-O**](https://kubernetes.io/docs/setup/production-environment/container-runtimes/#cri-o)
* [**Containerd**](https://kubernetes.io/docs/setup/production-environment/container-runtimes/#containerd)

For this cluster , we are going to use Containerd as it’s Container runtimes.

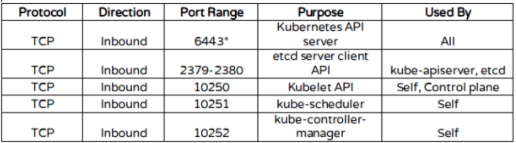
Prerequisites:

1. Three nodes running RHEL 8 out from which 1 Master Node and 2 Worker Nodes.
2. It is recommended that your nodes should have at least 2 CPUs with 2GB RAM or more per machine. This is not a strict requirement but good to have.
3. Internet connectivity on all your nodes. We will be fetching Kubernetes and other required packages from the repository. Equally, you will need to make sure that the DNF package manager is installed by default and can fetch packages remotely.

\* showcased only worker node

Installation of Kubernetes Cluster on Master-Node

Kubernetes makes use of various ports for communication and access and these ports need to be accessible to Kubernetes and not limited by the firewall. If your cluster behaves abnormally , you can configure the firewall rules on the ports.



k8ports

Step 1: Login to the node and here i will be performing all the operation on root

*[root@k8master ~]# cat /etc/redhat-release  
Red Hat Enterprise Linux release 8.2 (Ootpa)*

Step 2 : Install Container runtime on RHEL 8

This section contains the necessary steps to use containerd as CRI runtime.

Use the following commands to install Containerd on your system:

Prerequisites

cat > /etc/modules-load.d/containerd.conf <<EOF  
overlay  
br\_netfilter  
EOFmodprobe overlay  
modprobe br\_netfilter*# Setup required sysctl params, these persist across reboots.*  
cat > /etc/sysctl.d/99-kubernetes-cri.conf <<EOF  
net.bridge.bridge-nf-call-iptables = 1  
net.ipv4.ip\_forward = 1  
net.bridge.bridge-nf-call-ip6tables = 1  
EOFsysctl --system

Install containerd

*### Install required packages*  
dnf install -y yum-utils device-mapper-persistent-data lvm2  
  
  
*## Add docker repository*  
dnf config-manager --add-repo=https://download.docker.com/linux/centos/docker-ce.repo  
Adding repo from: <https://download.docker.com/linux/centos/docker-ce.repo>  
[root@k8master yum.repos.d]# dnf update -y && dnf install -y containerd.io  
  
*## Configure containerd*  
mkdir -p /etc/containerd  
containerd config default > /etc/containerd/config.toml*# Restart containerd*   
systemctl restart containerd*# Enable containerd on boot* root@k8master ~]# systemctl enable containerd

Step 3: Install Kubernetes (Kubeadm, kubelet and kubectl) on RHEL 8

Next, you will need to add Kubernetes repositories manually as they do not come installed by default on RHEL8.

Kubeadm helps you bootstrap a Kubernetes cluster. With kubeadm, you are going to create/enable single-control-plane.

Kubeadm also supports other cluster lifecycle functions, such as upgrades, downgrade, and managing bootstrap tokens

# Add yum repo file for Kubernetes # cat <<EOF > /etc/yum.repos.d/kubernetes.repo  
[kubernetes]  
name=Kubernetes  
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64  
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  
EOF#Install Kubernetes (kubeadm, kubelet and kubectl) [root@k8master ~]# dnf install -y kubeadm-1.17.0 kubelet-1.17.0 kubectl-1.17.0

When the installation completes successfully, enable and start the kubelet service

[root@k8master ~]# systemctl enable kubelet[root@k8master ~]# echo 'KUBELET\_EXTRA\_ARGS="--fail-swap-on=false"' > /etc/sysconfig/kubelet[root@k8master ~]# systemctl start kubelet

Step 4: Create a control-plane Master with kubeadm

[root@k8master]# kubeadm init --pod-network-cidr=192.168.0.0/16  
--  
--  
--  
Your Kubernetes control-plane has initialized successfully!

Next, copy the following join-token and store it somewhere, as we required to run this command on the worker nodes later.

kubeadm join 10.128.15.211:6443 --token 0xbszv.o9j5fim3j21xz47a \ --discovery-token-ca-cert-hash sha256:876637c5e5c74fcafa05372d187b58d36aa418b9faa46e1ec4c7388443143087

Once Kubernetes initialized successfully, you must enable your user to start using the cluster. In our scenario, we will be using the root user. You can also start the cluster using sudo user as shown.

To use root, run:

To start using your cluster, you need to run the following as a regular user:   
mkdir -p $HOME/.kube   
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config   
chown $(id -u):$(id -g) $HOME/.kube/config

Now confirm that the kubectl command is activated.

[root@k8master ~]# kubectl get nodes   
NAME STATUS ROLES AGE VERSION  
k8master NotReady master 35m v1.17.0

ATM, you will see the status of the k8-master is ‘NotReady’. This is because we are yet to deploy the overlay network for the cluster.