Install Kubernetes Cluster on CentOS 7 with kubeadm and CRI-O

|  |  |  |
| --- | --- | --- |
| **Server Type** | **Server Hostname** | **Specs** |
| Master | master-node | 4GB Ram, 2vcpus |
| Worker | node1 | 2GB Ram, 1vcpus |
| Worker | node2 | 2GB Ram, 1vcpus |

nano /etc/hosts

192.168.80.205 master

192.168.80.192 node2

192.168.80.204 node1

**yum -y update**

**Configure Firewalld**

I recommend you disable firewalld on your nodes:

systemctl disable firewalld

**Disable SELinux and Swap**

If you have SELinux in enforcing mode, turn it off or use Permissive mode.

setenforce 0

sed -i 's/^SELINUX=.\*/SELINUX=permissive/g' /etc/selinux/config

Turn off swap.

sed -i '/ swap / s/^\(.\*\)$/#\1/g' /etc/fstab

swapoff -a

Configure sysctl.

* modprobe br\_netfilter: This module is required to enable transparent masquerading and to facilitate Virtual Extensible LAN (VxLAN) traffic for communication between Kubernetes pods across the cluster.
* overlay module Linux: An overlay filesystem combines two filesystems - an 'upper' filesystem and a 'lower' filesystem of kubernetes.

modprobe overlay

modprobe br\_netfilter

tee /etc/sysctl.d/kubernetes.conf<<EOF

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip\_forward = 1

EOF

sysctl --system

**Install kubelet, kubeadm and kubectl**

Once the servers are rebooted, add Kubernetes repository for CentOS 7 to all the servers.

tee /etc/yum.repos.d/kubernetes.repo<<EOF

[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

Then install required packages.

yum clean all && yum -y makecache

yum -y install epel-release git curl wget kubelet kubeadm kubectl --disableexcludes=kubernetes

systemctl start kubelet

systemctl enable kubelet

**Using CRI-O Container Runtime**

For CRI-O below are the installation steps.

*# Add CRI-O repo*

OS=CentOS\_7

VERSION=1.22

curl -L -o /etc/yum.repos.d/devel:kubic:libcontainers:stable.repo <https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/$OS/devel:kubic:libcontainers:stable.repo>

curl -L -o /etc/yum.repos.d/devel:kubic:libcontainers:stable:cri-o:$VERSION.repo <https://download.opensuse.org/repositories/devel:kubic:libcontainers:stable:cri-o:$VERSION/$OS/devel:kubic:libcontainers:stable:cri-o:$VERSION.repo>

*#Install CRI-O*

yum remove docker-ce docker-ce-cli containerd.io

yum install cri-o -y

*# Start and enable Service*

systemctl daemon-reload

systemctl start crio

systemctl enable crio

**--------------------------Common Steps End------------------------**

* **Run Only Master: -**

**Initialize your control-plane node**

Login to the server to be used as master and make sure that the *br\_netfilter* module is loaded:

$ lsmod | grep br\_netfilter

br\_netfilter 22256 0

bridge 151336 2 br\_netfilter,ebtable\_broute

Pull container images:

$ kubeadm config images pull

[config/images] Pulled k8s.gcr.io/kube-apiserver:v1.22.2

[config/images] Pulled k8s.gcr.io/kube-controller-manager:v1.22.2

[config/images] Pulled k8s.gcr.io/kube-scheduler:v1.22.2

[config/images] Pulled k8s.gcr.io/kube-proxy:v1.22.2

[config/images] Pulled k8s.gcr.io/pause:3.5

[config/images] Pulled k8s.gcr.io/etcd:3.5.0-0

[config/images] Pulled k8s.gcr.io/coredns/coredns:v1.8.4

Create cluster:

kubeadm init --pod-network-cidr=10.85.0.0/16 --upload-certs --control-plane-endpoint=master

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

nodePort

This setting makes the service visible *outside* the Kubernetes cluster by the node’s IP address and the port number declared in this property. The service also has to be of type NodePort (if this field isn’t specified, Kubernetes will allocate a node port automatically).

port

Expose the *service* on the specified port internally within the cluster. That is, the service becomes visible on this port, and will send requests made to this port to the pods selected by the service.

targetPort

This is the port on the *pod* that the request gets sent to. Your application needs to be listening for network requests on this port for the service to work.

* **Install network plugin**

In this guide we’ll use [Calico](https://projectcalico.org/). You can choose any other [supported network plugins](https://kubernetes.io/docs/concepts/cluster-administration/addons/).

* kubectl create -f <https://docs.projectcalico.org/manifests/tigera-operator.yaml>
* kubectl create -f <https://docs.projectcalico.org/manifests/custom-resources.yaml>

***Method 2 for Calico:***

***Install Calico***

* Install the Tigera Calico operator and custom resource definitions.

kubectl create -f <https://raw.githubusercontent.com/projectcalico/calico/v3.24.5/manifests/tigera-operator.yaml>

* Install Calico by creating the necessary custom resource. For more information on configuration options available in this manifest, see [the installation reference](https://projectcalico.docs.tigera.io/reference/installation/api).

kubectl create -f <https://raw.githubusercontent.com/projectcalico/calico/v3.24.5/manifests/custom-resources.yaml>