Installing Kubernetes v1.28 with Kubeadm

Before you begin

- A compatible Linux host. The Kubernetes project provides generic instructions for Linux distributions based on Debian and Red Hat, and those distributions without a package manager.
- 4 GB for Master node and 2 GB for workers.
- 2 CPUs or more.
- Full network connectivity between all machines in the cluster (public or private network is fine)

Hostname	IP Address	Used as
k8master.exmaple.com	192.168.43.156	control-plane
worker1.example.com	192.168.43.157	worker node

1st we need to set our hostname on systems for this run this command

[root@k8master ~]# hostnamectl set-hostname k8master.example.com

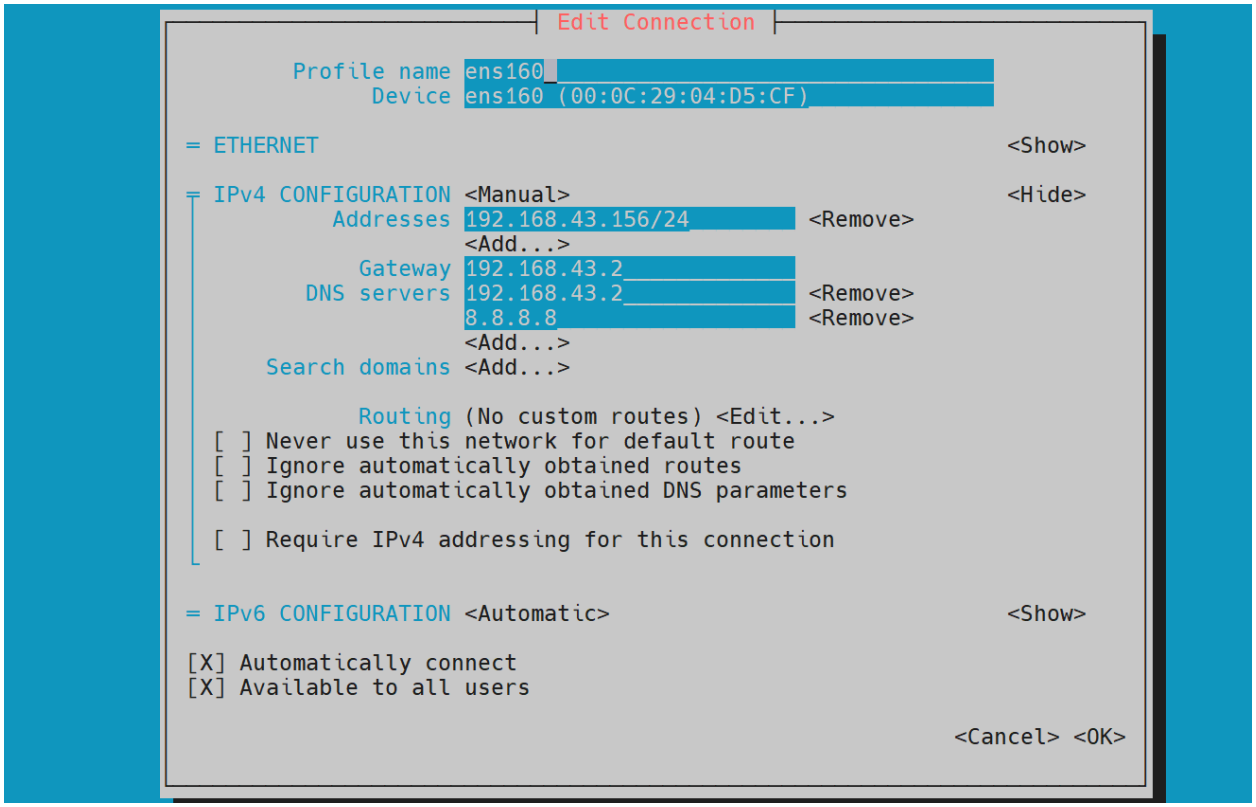
[root@k8master ~]# nmcli -p dev sh
{ run this command to check your gateway, DNS, IP, etc}

```
Device details (ens160)
GENERAL.DEVICE:                         ens160
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         00:0C:29:04:D5:CF
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                     ens160
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/ActiveConnection
WIRED-PROPERTIES.CARRIER:               on
IP4.ADDRESS[1]:                         192.168.43.156/24
IP4.GATEWAY:                            192.168.43.2
IP4.ROUTE[1]:                           dst = 192.168.43.0/24, nh = 0.0.0.0, mt = 100
IP4.ROUTE[2]:                           dst = 0.0.0.0/0, nh = 192.168.43.2, mt = 100
IP4.DNS[1]:                             192.168.43.2
IP4.DOMAIN[1]:                          localdomain
IP6.ADDRESS[1]:                         fe80::20c:29ff:fe04:d5cf/64
IP6.GATEWAY:                            --
IP6.ROUTE[1]:                           dst = fe80::/64, nh = ::, mt = 1024
```

After verifying your dns, gateway you need to set manual IP configuration

Run this command to set static IP on your machine

[root@k8master ~]# nmtui



After adding the network you need to configure local dns for this open /etc/hosts file and enter these lines

[root@k8master ~]# vim /etc/hosts

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.43.156 k8master.example.com k8master
192.168.43.157 worker1.example.com  worker1
```

Now we need to deactivate the swap space
1st we need to verify whether swap space is working or not running this command to verify

[root@k8master ~]# free -m

```
[root@k8master ~]# free -m
              total        used        free      shared  buff/cache   available
Mem:           3627         535        3108           9         210        3092
Swap:          3071           0        3071
[root@k8master ~]# swapon -s
Filename                                Type            Size            Used            Priority
/dev/dm-1                               partition       3145724         0               -2
[root@k8master ~]#
```

If swap space is activated you need to disable swap temporarily and permanent

To disable the swap temporarily you need to run
[root@k8master ~]# swapoff -a

To disable swap permanently you need to open the/etc/fstab file and comment swap entry

```
# units generated from this file.
#
/dev/mapper/rl-root     /                       xfs     defaults        0 0
UUID=63aa319a-771c-4d57-b812-d1c51d5c34a6 /boot                   xfs     defaults        0 0
#/dev/mapper/rl-swap     none                    swap    defaults        0 0
~
~
```

Now we need to disable se-linux temporarily and permanent

In this command, we check the se-linux status
[root@k8master ~]# **sestatus**
SELinux status: enabled
SELinuxfs mount: /sys/fs/selinux
SELinux root directory: /etc/selinux
Loaded policy name: targeted
Current mode: enforcing
Mode from config file: enforcing
Policy MLS status: enabled
Policy deny_unknown status: allowed
Memory protection checking: actual (secure)
Max kernel policy version: 33

Run this command to disable se-linux temporarily
[root@k8master ~]# setenforce 0

Verify whether se-linux is disabled or not to verify run this command
[root@k8master ~]# getenforce
Permissive

Now disabled se-linux permanently for this open /etc/selinux/config
[root@k8master ~]# vim /etc/selinux/config
{ in this file you need to change SELINUX=enforcing to SELINUX=disabled }

```
# To revert back to SELinux enabled:
#
#     grubby --update-kernel ALL --remove-args selinux
#
SELINUX=disabled
# SELINUXTYPE= can take one of these three values:
```

After this, we need to add a port on the firewall
As Kubernetes 1.28 needs these ports
[6443,2379,2380,10250,10251,10252,10257,10259,179] for TCP
[4789] for udp
We need to enable all these ports to enable running this command

[root@k8master ~]# firewall-cmd
--add-port={6443,2379,2380,10250,10251,10252,10257,10259,179}/tcp --permanent
success

[root@k8master ~]# firewall-cmd --add-port=4789/udp --permanent
success
[root@k8master ~]# firewall-cmd --reload
Success

After adding ports to the firewall verify whether ports were added or not for this run of this command
[root@k8master ~]# firewall-cmd --list-all

```
[root@k8master ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160
  sources:
  services: cockpit dhcpv6-client ssh
  ports: 6443/tcp 2379/tcp 2380/tcp 10250/tcp 10251/tcp 10252/tcp 10257/tcp 10259/tcp 179/tcp 4789/udp
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[root@k8master ~]#
```

Now we need to load a kernel module for k8s load {overlay and br_netfilter} for load modules temporary, run this command
[root@k8master ~]# modprobe overlay
[root@k8master ~]# modprobe br_netfilter

Run this command to verify whether modules are loaded or not
[root@k8master ~]# lsmod | grep br_
br_netfilter 32768 0
bridge 315392 1 br_netfilter
[root@k8master ~]# lsmod | grep over
overlay 155648 0

For loading modules for permanent we need to create a config file on /etc/modules-load.d this location for creating a config file run this command
In this file enter these two lines
overlay
br_netfilter
[root@k8master ~]# vim /etc/modules-load.d/k8s.conf
overlay
br_netfilter

To verify the config file run this command
[root@k8master ~]# cat /etc/modules-load.d/k8s.conf
overlay
br_netfilter

Now we are creating a config file for iptables for this run this command on your terminal directly
[root@k8master ~]# cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
EOF

To verify whether rules are added or not run this command
[root@k8master ~]# sysctl --system

```
[root@k8master ~]# sysctl --system
* Applying /usr/lib/sysctl.d/10-default-yama-scope.conf ...
* Applying /usr/lib/sysctl.d/50-coredump.conf ...
* Applying /usr/lib/sysctl.d/50-default.conf ...
* Applying /usr/lib/sysctl.d/50-libkcapi-optmem_max.conf ...
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
* Applying /usr/lib/sysctl.d/50-redhat.conf ...
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/k8s.conf ...
* Applying /etc/sysctl.conf ...
kernel.yama.ptrace_scope = 0
kernel.core_pattern = |/usr/lib/systemd/systemd-coredump %P %u %g %s %t %c %h
kernel.core_pipe_limit = 16
fs.suid_dumpable = 2
kernel.sysrq = 16
kernel.core_uses_pid = 1
net.ipv4.conf.default.rp_filter = 2
net.ipv4.conf.ens160.rp_filter = 2
net.ipv4.conf.lo.rp_filter = 2
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.ens160.accept_source_route = 0
net.ipv4.conf.lo.accept_source_route = 0
net.ipv4.conf.default.promote_secondaries = 1
net.ipv4.conf.ens160.promote_secondaries = 1
net.ipv4.conf.lo.promote_secondaries = 1
net.ipv4.ping_group_range = 0 2147483647
net.core.default_qdisc = fq_codel
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
fs.protected_regular = 1
fs.protected_fifos = 1
net.core.optmem_max = 81920
kernel.pid_max = 4194304
kernel.kptr_restrict = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.ens160.rp_filter = 1
net.ipv4.conf.lo.rp_filter = 1
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
[root@k8master ~]#
```

All prerequisites for **worker and master** are fulfilled now we are moving towards installing and configuring the Kubernetes cluster…
==========

Master Node Configurations

Run this command to install yum utilities on the master

[root@k8master ~]# yum install yum-utils -y

Removing podman
[root@k8master ~]# yum remove podman -y

Adding docker repo to install container runtime in our case we are using containerd
[root@k8master ~]# yum-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

After adding docker repo installing containerd for intsall containerd run this command

[root@k8master ~]# yum install containerd.io -y

To configure containerd run this command

This command is for renaming the containerd config file
[root@k8master ~]# mv /etc/containerd/config.toml /etc/containerd/config.toml.backup

This command is for generating new config file for containerd
[root@k8master]# containerd config default > /etc/containerd/config.toml

Verify file generated or not
[root@k8master]# ls /etc/containerd
config.toml config.toml.backup

Now open /etc/containerd/config.toml file search SystemdCgroup line and add
SystemdCgroup = true
[root@k8master ~]# vim /etc/containerd/config.toml

```
124            ShimCgroup = ""
125            SystemdCgroup = true
```

Enabling and starting containerd

[root@k8master ~]# systemctl enable --now containerd.service
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
[root@k8master ~]# systemctl restart containerd.service
[root@k8master ~]# systemctl restart containerd.service

enabled
[root@k8master ~]# systemctl is-active containerd.service
active

Now configure k8s repo for v1.28 to configure run this command

For reference [k8s docs]

[root@k8master ~]# cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF

After adding k8s repo installing kubelet,kubeadm,kubectl

[root@k8master ~]# yum install -y kubelet kubeadm kubectl
--disableexcludes=kubernetes
Enabled kubelet
[root@k8master ~]# systemctl enable kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service →
/usr/lib/systemd/system/kubelet.service.

Adding user in your master node
[root@k8master ~]# useradd kiosk
[root@k8master ~]# passwd kiosk
Changing password for user kiosk.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.

Giving sudo access to kiosk user
[root@k8master ~]# vim /etc/sudoers

```
110 # %wheel        ALL=(ALL)       NOPASSWD: ALL
111   %kiosk        ALL=(ALL)       NOPASSWD: ALL
```

==========

All configurations for the master are done, now we move towards worker node configurations…

To configure worker nodes, in my case only 1 worker node is available

==========

1st we need to set our hostname on systems for this run this command

[root@k8master ~]# hostnamectl set-hostname worker1.example.com

Run this command to set static ip on your machine

[root@k8master ~]# nmtui

```
                          Edit Connection
                                                                    ↑
          Profile name ens160
                Device ens160 (00:0C:29:00:32:6F)

 = ETHERNET                                                    <Show>

 = IPv4 CONFIGURATION <Manual>                                 <Hide>
            Addresses 192.168.43.157/24      <Remove>
                      <Add...>
              Gateway 192.168.43.2
          DNS servers 192.168.43.2           <Remove>
                      8.8.8.8                <Remove>
                      <Add...>
       Search domains <Add...>

              Routing (No custom routes) <Edit...>
   [ ] Never use this network for default route
   [ ] Ignore automatically obtained routes
   [ ] Ignore automatically obtained DNS parameters

   [ ] Require IPv4 addressing for this connection


 = IPv6 CONFIGURATION <Automatic>                              <Show>
```

After adding the network you need to configure local dns for this open /etc/hosts file and enter these lines

[root@worker1 ~]# vim /etc/hosts

```
[root@worker1 ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.43.156 k8master.example.com k8master
192.168.43.157 worker1.example.com worker1
```

Now we need to deactivate the swap space

1st we need to verify whether swap space is working or not running this command to verify

```
root@worker1 ~]# free -m
              total        used        free      shared  buf
lem:           1743         924          80          10
wap:              0           0           0
root@worker1 ~]#
```

To disable the swap temporarily you need to run

[root@worker1 ~]# swapoff -a

To disable swap permanently you need to open /etc/fstab file and comment swap entry

```
#
/dev/mapper/rl-root     /                       xfs     defaults        0 0
UUID=63aa319a-771c-4d57-b812-d1c51d5c34a6 /boot                   xfs     defaults        0 0
#/dev/mapper/rl-swap     none                    swap    defaults        0 0
~
~
```

Now we need to disable the selinux temporarily and permanent

In this command, we check the se-linux status

[root@worker1 ~]# sestatus

SELinux status: enabled
SELinuxfs mount: /sys/fs/selinux
SELinux root directory: /etc/selinux
Loaded policy name: targeted
Current mode: enforcing
Mode from config file: enforcing
Policy MLS status: enabled
Policy deny_unknown status: allowed
Memory protection checking: actual (secure)
Max kernel policy version: 33

Run this command to disable se-linux temporarily

[root@worker1 ~]# setenforce 0

Verify whether se-linux is disabled or not to verify run this command

[root@worker1 ~]# getenforce

Permissive

Now disabled se-linux permanently for this open /etc/selinux/config

[root@worker1 ~]# vim /etc/selinux/config

{ in this file you need to change SELINUX=enforcing to SELINUX=disabled }

```
# To revert back to SELinux enabled:
#
#    grubby --update-kernel ALL --remove-args selinux
#
SELINUX=disabled
# SELINUXTYPE= can take one of these three values:
```

After this, we need to add a port on the firewall
As Kubernetes 1.28 needs these ports for workers we need these ports only
[179,10250,30000-32767] for TCP
[4789] for udp
We need to enable all these ports to enable running this command

[root@worker1 ~]# firewall-cmd --permanent --add-port={179,10250,30000-32767}/tcp
success

[root@worker1 ~]# firewall-cmd --permanent --add-port=4789/udp
success
[root@worker1 ~]# firewall-cmd --reload
Success

After adding ports to the firewall verify whether ports were added or not for this run of this command
[root@worker1 ~]# firewall-cmd --list-all

```
[root@worker1 ~]# ^C
[root@worker1 ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160
  sources:
  services: cockpit dhcpv6-client ssh
  ports: 179/tcp 10250/tcp 30000-32767/tcp 4789/udp
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[root@worker1 ~]#
```

Now we need to load a kernel module for k8s load {overlay and br_netfilter} for load modules temporary, run this command

[root@worker1 ~]# modprobe overlay
[root@worker1 ~]# modprobe br_netfilter
Run this command to verify whether modules are loaded or not
[root@worker1 ~]# lsmod | grep br_
br_netfilter 32768 0
bridge 315392 1 br_netfilter

[root@worker1 ~]# lsmod | grep over
overlay 155648 0

For loading modules for permanent we need to create a config file on /etc/modules-load.d this location for creating a config file run this command
In this file enter these two lines
overlay
br_netfilter
[root@worker1 ~]# vim /etc/modules-load.d/k8s.conf
overlay
br_netfilter

To verify the config file run this command
[root@worker1 ~]# cat /etc/modules-load.d/k8s.conf
overlay
br_netfilter

Now we are creating a config file for iptables for this run this command on your terminal directly
[root@worker1 ~]# cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
EOF

To verify whether rules are added or not run this command
[root@worker1 ~]# sysctl --system

```
[root@k8master ~]# sysctl --system
* Applying /usr/lib/sysctl.d/10-default-yama-scope.conf ...
* Applying /usr/lib/sysctl.d/50-coredump.conf ...
* Applying /usr/lib/sysctl.d/50-default.conf ...
* Applying /usr/lib/sysctl.d/50-libkcapi-optmem_max.conf ...
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
* Applying /usr/lib/sysctl.d/50-redhat.conf ...
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/k8s.conf ...
* Applying /etc/sysctl.conf ...
kernel.yama.ptrace_scope = 0
kernel.core_pattern = |/usr/lib/systemd/systemd-coredump %P %u %g %s %t %c %h
kernel.core_pipe_limit = 16
fs.suid_dumpable = 2
kernel.sysrq = 16
kernel.core_uses_pid = 1
net.ipv4.conf.default.rp_filter = 2
net.ipv4.conf.ens160.rp_filter = 2
net.ipv4.conf.lo.rp_filter = 2
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.ens160.accept_source_route = 0
net.ipv4.conf.lo.accept_source_route = 0
net.ipv4.conf.default.promote_secondaries = 1
net.ipv4.conf.ens160.promote_secondaries = 1
net.ipv4.conf.lo.promote_secondaries = 1
net.ipv4.ping_group_range = 0 2147483647
net.core.default_qdisc = fq_codel
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
fs.protected_regular = 1
fs.protected_fifos = 1
net.core.optmem_max = 81920
kernel.pid_max = 4194304
kernel.kptr_restrict = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.ens160.rp_filter = 1
net.ipv4.conf.lo.rp_filter = 1
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
[root@k8master ~]#
```

All prerequisites for **worker and master both** are fulfilled now we are moving towards installing and configuring the Kubernetes worker node …

Worker Node Configurations

Run this command to install yum utilities on the worker node

[root@worker1 ~]# yum install yum-utils -y

Removing podman
[root@worker1 ~]# yum remove podman -y

Adding docker repo to install container runtime in our case we are using containerd
[root@worker1 ~]# yum-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

After adding the docker repo and installing containerd for installing containerd run this command

[root@worker1 ~]# yum install containerd.io -y

To configure containerd run this command

This command is for renaming the containerd config file
[root@worker1 ~]# mv /etc/containerd/config.toml /etc/containerd/config.toml.backup

This command is for generating a new config file for containerd
[root@worker1 ~]# containerd config default > /etc/containerd/config.toml

Verify file is generated or not
[root@worker1 ~]# ls /etc/containerd
config.toml config.toml.backup

Now open /etc/containerd/config.toml file search SystemdCgroup line and add
SystemdCgroup = true
[root@worker1 ~]# vim /etc/containerd/config.toml

```
124             ShimCgroup = ""
125             SystemdCgroup = true
```

Enabling and starting containerd

[root@worker1 ~]#systemctl enable --now containerd.service
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
[root@worker1 ~]# systemctl restart containerd.service
[root@worker1 ~]# systemctl is-enabled containerd.service
enabled
[root@worker1 ~]# systemctl is-active containerd.service
active

Now configure the k8s repo for v1.28 to configure and run this command

For reference [k8s docs]

```
[root@worker1 ~]# cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

After adding k8s repo installing kubelet,kubeadm,kubectl

[root@worker1 ~]# yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

Enabled kubelet

[root@worker1 ~]# systemctl enable kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.

==========Worker node configuration finishd==========

[root@k8master ~]# kubeadm init

After completing this command you will get this type of output on your terminal

```
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 192.168.43.156:6443 --token epiv7r.q6nkchryz72ar9jl \
        --discovery-token-ca-cert-hash sha256:e14c31f6d21c9495b4ddf941d9cacbb63ac7220e
2d364d8baee624af422d0713
```

Save this output to your notepad file this is very important of do not forget to save this

Before joining worker nodes run this commands on master node as kiosk user

```
[kiosk@k8master ~]$ mkdir -p $HOME/.kube

[kiosk@k8master ~]$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

[kiosk@k8master ~]$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

-- TBC --

After finishing worker node configuration go to your master node and setup calico (CNI Plugin) for network

For setting calico plugin you need to download calico release v3.26.3

To download calico v3.26.3 Go to this link and download release-v3.26.3.tgz this file

https://github.com/projectcalico/calico/releases/tag/v3.26.3

Or

Run this command on your master node with kiosk user

[kiosk@k8master ~]$ wget https://github.com/projectcalico/calico/releases/download/v3.26.3/release-v3.26.3.tgz

To initialize cluster run this command on master node

After downloading this file extract this file,for extracting run this command

[kiosk@k8master ~]$ tar -xvf /home/kiosk/release-v3.26.3

Verify file is extracted or not to verify run this command

[kiosk@k8master ~]$ ls

release-v3.26.3 release-v3.26.3.tgz

After this go to this path

[kiosk@k8master ~]$ cd release-v3.26.3/

[kiosk@k8master release-v3.26.3]$ ls

bin images manifests

[kiosk@k8master release-v3.26.3]$ cd manifests/

[kiosk@k8master manifests]$

Verify that your in right location

[kiosk@k8master manifests]$ pwd

/home/kiosk/release-v3.26.3/manifests

After this run this command on master node only
[do not run this command on worker node]

[kiosk@k8master manifests]$ kubectl apply -f /home/kiosk/release-v3.26.3/manifests/calico.yaml

After setting Calico we are going to initialize our k8s cluster

After we need to join our worker node to master node

For joining our run this command to your worker node as root in my case my token is different from yours

```
[root@worker1 ~]# kubeadm join 192.168.43.156:6443 --token epiv7r.q6nkchryz72ar9jl \
        --discovery-token-ca-cert-hash
sha256:e14c31f6d21c9495b4ddf941d9cacbb63ac7220e 2d364d8baee624af422d0713
```

Run this commands to verify you cluster

[kiosk@k8master]$ kubectl get pods --all-namespaces

[kiosk@k8master]$ kubectl get nodes

```
NAME                    STATUS     ROLES        AGE   VERSION
k8master.example.com    Ready      control-plane   72m   v1.28.4
worker1.example.com     Ready      <none>          45m   v1.28.4
```

==========END==========