RedHat - Install a Kubernetes Cluster on RHEL 9.x | Rocky 9.x: A Step-by-Step Guide

> Last verified: October 2026 against the official Kubernetes, containerd, Docker, Calico and EPEL documentation (see **Sources** at the bottom). Versions used in this guide: Kubernetes **v1.37** (latest patch v1.37.1; supported minors are 1.37, 1.36 and 1.35), containerd **2.x** from Docker's `containerd.io` package, Calico **v3.33.0**, RHEL **9.8** / Rocky Linux **9.8**.

Original third-party references (these may be out of date; the official sources are listed at the bottom of the page):

- https://infotechys.com/install-a-kubernetes-cluster-on-rhel-9
- https://www.youtube.com/watch?v=_ELvCuXO6y4
- https://medium.com/weeklycloud/kubernetes-installation-on-rhel-9-d5629f2fa4f9
- https://www.youtube.com/watch?v=vX2n05t0AQg&t=1782s

Prerequisites

On **each node** (control plane and workers), as root:

- RHEL 9.x or Rocky Linux 9.x (9.8 is the current minor release of both).
- At least 2 GB of RAM per machine and 2 or more CPUs on control plane machines.
- Full network connectivity between all machines in the cluster.
- A unique hostname, MAC address and `product_uuid` on every node (watch out for cloned VMs). Check with `ip link` and `cat /sys/class/dmi/id/product_uuid`.

Update the System

You can choose to disable the firewall, or keep it and open the required ports. SELinux is set to **permissive** in both cases.

Set SELinux to permissive (this is the method in the official kubeadm guide, so it applies to both options):

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

Permissive mode effectively disables SELinux enforcement. Kubernetes requires this so containers can access the host filesystem, which some network plugins need. You can leave SELinux enforcing only if you know how to configure it yourself; kubeadm does not support that setup.

> Option A: disable the firewall

```
systemctl disable --now firewalld.service
```

Calico's system requirements recommend disabling firewalld (or any other iptables manager) because it can interfere with the rules Calico adds.

> Option B: keep firewalld and open the required ports

For Kubernetes components to communicate across nodes, open these ports. The Kubernetes ports come from the official *Ports and Protocols* reference. The Calico ports come from Calico's system requirements; the default Calico install in this guide uses BGP plus VXLAN.

- **6443/tcp**: Kubernetes API server
- **2379-2380/tcp**: etcd server client API
- **10250/tcp**: kubelet API
- **10257/tcp**: kube-controller-manager
- **10259/tcp**: kube-scheduler
- **179/tcp**: Calico BGP
- **5473/tcp**: Calico Typha (operator installs always deploy Typha)
- **4789/udp**: Calico VXLAN

Commands to open ports on the **Control Plane Node**:

```
firewall-cmd --permanent --add-port={6443,2379-2380,10250,10257,10259,179,5473}/tcp
firewall-cmd --permanent --add-port=4789/udp
firewall-cmd --reload
```

These ports handle node-to-node communication and pod access on the workers:

- **10250/tcp**: kubelet API
- **10256/tcp**: kube-proxy
- **30000-32767/tcp and udp**: NodePort Services
- **179/tcp**: Calico BGP
- **5473/tcp**: Calico Typha (operator installs always deploy Typha)
- **4789/udp**: Calico VXLAN

Commands to open ports on **Worker Nodes**:

```
firewall-cmd --permanent --add-port={10250,10256,30000-32767,179,5473}/tcp
firewall-cmd --permanent --add-port={30000-32767,4789}/udp
firewall-cmd --reload
```

> End of firewall options

Stop NetworkManager from managing Calico's interfaces. Calico's troubleshooting guide says to do this before you install Calico. Run this on every node; the reboot in Step 4 applies it.

```
cat > /etc/NetworkManager/conf.d/calico.conf << EOF
[keyfile]
unmanaged-devices=interface-name:cali*;interface-name:tunl*;interface-name:vxlan.calico;interface-name:vxlan-v6.calico;interface-name:wireguard.cali;interface-name:wg-v6.cali
EOF
```

EPEL Release (optional; Kubernetes itself does not need EPEL)

RHEL 9:

```
subscription-manager repos --enable codeready-builder-for-rhel-9-$(arch)-rpms
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
```

Rocky Linux 9:

```
dnf config-manager --set-enabled crb
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
```

After installing EPEL, rerun the upgrade to apply any available updates:

```
dnf upgrade -y
```

If you are running on a VMware virtual machine, run the following:

```
dnf install open-vm-tools -y
```

Install vim colors for scripting (optional):

```
dnf install git -y
git clone https://github.com/flazz/vim-colorschemes ~/.vim/
cp ~/.vim/colors/desert.vim /etc/vimrc.local
```

Step 1: Add Kernel Modules

Load the kernel modules used by the container runtime and pod networking (**on each node**):

```
modprobe br_netfilter
modprobe overlay
```

Next, create a configuration file (**as the root user on each node**) so these modules load at boot:

```
cat > /etc/modules-load.d/k8s.conf << EOF
br_netfilter
overlay
EOF
```

Step 2: Configure Sysctl

Set the kernel parameters that Kubernetes networking relies on (**on each node**):

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

These commands adjust the following kernel parameters:

Kernel Parameter	Description
net.ipv4.ip_forward	Enables IPv4 packet forwarding. This is the only sysctl the official Kubernetes container runtime guide requires.
net.bridge.bridge-nf-call-iptables	Enables iptables to process bridged IPv4 traffic. Not required by the official guide, but some network plugins expect it. Needs the `br_netfilter` module.

Kernel Parameter	Description
net.bridge.bridge-nf-call-ip6tables	Enables ip6tables to process bridged IPv6 traffic. Not required by the official guide, but some network plugins expect it. Needs the `br_netfilter` module.

Apply the changes without rebooting and verify:

```
sysctl --system
sysctl net.ipv4.ip_forward
```

Step 3: Disable Swap

By default the kubelet fails to start if swap is detected. To disable swap on each server in your cluster:

```
swapoff -a
```

This command turns off all swap devices. To keep swap off after a reboot, comment out the swap entry in `/etc/fstab`:

```
sed -e '/swap/s/^/#/g' -i /etc/fstab
```

The `sed` command finds the line that contains the swap entry and comments it out by adding a `#` at the start, for example:

```
#/dev/mapper/vg00-swap   none                swap    defaults        0 0
```

Alternatively, newer Kubernetes releases can run with swap enabled. Set `failSwapOn: false` and a `swapBehavior` in the kubelet configuration; see the official *Swap memory management* documentation. This guide disables swap.

Step 4: Install Containerd

In this step, we'll install containerd on each node as the container runtime. Kubernetes talks to containerd through the CRI. Docker Engine itself is not a CRI runtime: dockershim was removed from the kubelet in v1.24.

> Add the Docker CE Repository

The `containerd.io` package comes from Docker's RHEL repository. First remove any conflicting packages; Docker's install guide lists these, and `dnf` may report that none are installed. Then add

the repository:

```
dnf remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate
docker-logrotate docker-engine podman runc
dnf -y install dnf-plugins-core
dnf config-manager --add-repo https://download.docker.com/linux/rhel/docker-ce.repo
```

Update the package cache, then install the `containerd.io` package. As of October 2026 it installs containerd 2.3.x.

```
dnf makecache
dnf -y install containerd.io
```

If prompted to accept the GPG key, verify that the fingerprint is `060A 61C5 1B55 8A7F 742B 77AA C52F EB6B 621E 9F35`.

> Configure Containerd

The `/etc/containerd/config.toml` shipped with the package contains `disabled_plugins = ["cri"]`, which disables the CRI plugin that Kubernetes needs. Replace it with the full default configuration:

```
containerd config default > /etc/containerd/config.toml
```

Next, set the `SystemdCgroup` option to `true`. The kubelet and the container runtime must use the same cgroup driver. The official docs recommend `systemd` on systemd-based hosts such as RHEL/Rocky and with cgroup v2. With containerd 2.x, the kubelet automatically detects the cgroup driver from containerd.

```
sed -i 's/SystemdCgroup = false/SystemdCgroup = true/g' /etc/containerd/config.toml
grep -n 'SystemdCgroup' /etc/containerd/config.toml
```

containerd 2.x uses a newer config format, so the setting is now under a different section. The `sed` command still works because it only changes the value:

```
[plugins.'io.containerd.cri.v1.runtime'.containerd.runtimes.runc.options]
  SystemdCgroup = true
```

(On containerd 1.x the section was `[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]`.)

Enable and start containerd so it also starts on boot:

```
systemctl enable --now containerd.service
```

Reboot the machine to apply the SELinux, module and NetworkManager changes:

```
systemctl reboot
```

After the reboot, check that containerd is up and running:

```
systemctl status containerd.service
```

Alternative runtime: CRI-O can be used instead of containerd. It is installed from the CRI-O project's own packages (see the cri-o/packaging README), and its socket is `unix:///var/run/crio/crio.sock`. Install only one container runtime per node. If kubeadm detects more than one, specify which to use with `--cri-socket`.

Step 5: Install Kubernetes Components

Install kubelet, kubeadm and kubectl on **every node** from the official community-owned repository at `pkgs.k8s.io`. Each Kubernetes minor version has its own repository; this one only has packages for **v1.37**. For another supported minor (1.36 or 1.35), change `v1.37` in both URLs.

Add Kubernetes Repository

The `exclude` line stops a normal `dnf update` from upgrading the Kubernetes packages. Kubernetes upgrades must follow the `kubeadm upgrade` procedure.

```
# This overwrites any existing configuration in /etc/yum.repos.d/kubernetes.repo
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.37/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.37/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

Install Kubernetes Packages

```
dnf makecache; dnf install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
```

The `--disableexcludes=kubernetes` flag lets this install use the Kubernetes packages that the repository's `exclude` line otherwise hides. This is the DNF4 syntax used on RHEL/Rocky 9.

> Enable the kubelet Service

```
systemctl enable --now kubelet
```

To verify the installation so far, run:

```
kubeadm version
kubelet --version
kubectl version --client
```

Don't worry about kubelet errors at this point. The kubelet restarts in a loop until `kubeadm init` (control plane) or `kubeadm join` (workers) gives it a configuration.

> Continued in RedHat - Kubernetes Cluster on RHEL 9.x | Rocky 9.x (Part 2): Control Plane, Workers & Test.

Revision #2
Created 11 June 2025 03:26:58 by Admin
Updated 24 October 2026 14:10:46 by Admin