ChatGPT

Preparing a Linux system for Kubernetes with cri-dockerd

This guide outlines the steps to configure a Linux system (for example Ubuntu 22.04) using Docker as the container runtime via the **cri-dockerd** adapter so that `kubeadm` can bootstrap a Kubernetes cluster. Docker is already installed; this guide focuses on installing `cri-dockerd`, enabling its service and socket, and tuning kernel parameters.

Prerequisites

- Docker Engine is installed and running on the host.
- The Debian package `cri-dockerd_0.3.9.3-0.ubuntu-jammy_amd64.deb` has been downloaded to your home directory (`~/`).
- You have root or `sudo` privileges.

1. Install the cri-dockerd adapter

The `cri-dockerd` package provides a Container Runtime Interface (CRI) shim that allows Kubernetes to communicate with Docker. Install the package with `dpkg` and fix any dependency issues:

```
sudo dpkg -i ~/cri-dockerd_0.3.9.3-0.ubuntu-jammy_amd64.deb

# If `dpkg` reports missing dependencies, fix them:
sudo apt-get install -f -y
```

2. Enable and start the cri-docker service and socket

The Debian package installs systemd unit files for `cri-docker.service` and `cri-docker.socket`. Reload systemd, enable the service and socket so they start automatically, and start them immediately. The DiveInto blog on CRI-dockerd illustrates using these commands [1]:

```
# Reload systemd to pick up the new unit files
sudo systemctl daemon-reload

# Enable and start both the service and the socket
sudo systemctl enable --now cri-docker.service cri-docker.socket
```

```
# Verify that the units are active
sudo systemctl status cri-docker.service cri-docker.socket
```

After these commands, the CRI socket should be listening at `/var/run/cri-dockerd.sock`. This path must be passed to kubeadm later via the `--cri-socket` flag when initializing the cluster.

3. Load kernel modules

Kubernetes networking requires the `overlay` and `br_netfilter` kernel modules. Load them now and ensure they load on every boot:

```
sudo modprobe overlay
sudo modprobe br_netfilter

# Persist module loading across reboots
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
```

4. Configure sysctl parameters

Kubernetes nodes must enable packet forwarding and ensure bridged network traffic passes through iptables. You must also increase the connection tracking limit to avoid dropping packets. Persist these values in `/etc/sysctl.d/k8s.conf` and apply them:

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables = 1
net.ipv6.conf.all.forwarding = 1
net.ipv4.ip_forward = 1
net.netfilter.nf_conntrack_max = 131072
EOF

# Reload sysctl settings
sudo sysctl --system
```

The file sets:

- `net.bridge.bridge-nf-call-iptables=1` – ensures that bridged network traffic is processed by iptables [2].
- `net.ipv6.conf.all.forwarding=1` – enables IPv6 forwarding [3].
- `net.ipv4.ip_forward=1` – enables IPv4 forwarding [2].

- `net.netfilter.nf_conntrack_max=131072` – increases the maximum number of tracked network connections [4].

The `sysctl --system` command applies all kernel parameter files under `/etc/sysctl.d/` as well as `/etc/sysctl.conf`.

5. Verify sysctl settings

Confirm that each parameter has the desired value:

```
sudo sysctl net.bridge.bridge-nf-call-iptables
sudo sysctl net.ipv6.conf.all.forwarding
sudo sysctl net.ipv4.ip_forward
sudo sysctl net.netfilter.nf_conntrack_max
```

Each command should report `= 1` (for the boolean flags) or `= 131072` for the connection tracking limit. If not, ensure the `k8s.conf` file is correctly formatted and run `sudo sysctl --system` again.

6. Initialize Kubernetes with kubeadm

With cri-dockerd running and the system tuned, you can bootstrap a Kubernetes control plane. Specify the CRI socket path so that kubelet uses Docker via the CRI. For example:

```
sudo kubeadm init \
  --pod-network-cidr=10.244.0.0/16 \
  --cri-socket unix:///var/run/cri-dockerd.sock
```

Be sure to include the `--cri-socket` flag; official guides and blogs demonstrate passing this argument when using cri-dockerd [1].

Following these steps will configure your Linux system to use Docker via the CRI interface and tune kernel parameters required by Kubernetes networking. Once initialized with `kubeadm`, you can proceed to install a network plugin (such as Calico or Flannel) and join additional worker nodes.

---

1 CRI-Dockerd: Kubernetes reunited with an old friend (Docker) | DiveInto.com
https://diveinto.com/blog/cri-dockerd

2 Kubernetes on Ubuntu 22.04 with CRI-Docker - Nathan Obert
https://www.nathanobert.com/posts/blog-kubernetes-on-ubuntu/

3 How to Enable IP forwarding on Linux (IPv4 / IPv6) - WHUK
https://www.webhosting.uk.com/kb/how-to-enable-ip-forwarding-on-linux-ipv4-ipv6/

4 nf_conntrack: table full, dropping packet. | Notes from a Nerd
https://paulroberts69.wordpress.com/2008/11/10/nf_conntrack-table-full-dropping-packet/