How To Install CRI-O On RHEL 10

By Pradeep Kumar / January 4, 2026 / 4 Minutes Of Reading

In this blog post, we will learn how to install CRI-O on RHEL 10 step by step and understand how to run a Pod and container using CRI-O.

CRI-O is a lightweight, OCI-compliant container runtime built exclusively for Kubernetes. It is designed to work natively with Kubernetes and provides only the features required by the Kubernetes Container Runtime Interface (CRI), making it efficient and production-ready.

With Docker no longer being the default runtime for Kubernetes, CRI-O has become a popular and recommended alternative. If you are running Kubernetes on RHEL 10, CRI-O is an excellent choice for a clean, Kubernetes-native container runtime.

Prerequisites

- A Running RHEL 10 System
- Local User with Sudo access
- Active Red Hat subscription (or valid developer subscription)
- Internet connectivity

Note: CRI-O versions must always match the Kubernetes minor version.

Without any further delay, let's jump into the CRI-O installation steps on RHEL 10.

1) Add CRI-O Repository

RHEL does not ship CRI-O by default, so you need to add the upstream repository.

First export the CRIO version **variable**,

```
$ CRIO_VERSION=v1.34
```

Next create the crio.repo file using following command.

```
cat <<EOF | sudo tee /etc/yum.repos.d/cri-o.repo
[cri-o]
name=CRI-O
baseurl=https://download.opensuse.org/repositories/isv:/cri-o
enabled=1
gpgcheck=1
gpgkey=https://download.opensuse.org/repositories/isv:/cri-o:
EOF
```

2) Set SELinux To Permissive Mode

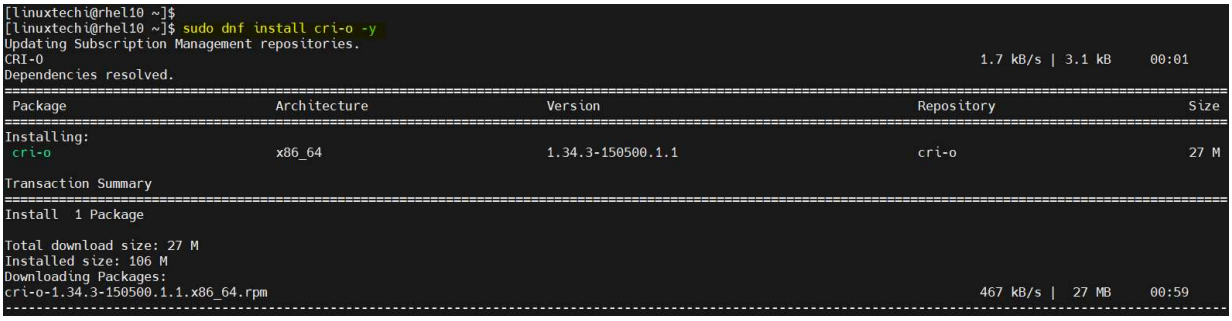
Run the following commands,

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

3) Install CRI-O On RHEL 10

As we have already setup the crio repository, so we are good to install crio installation, run the following dnf command.

```
$ sudo dnf install cri-o -y
```

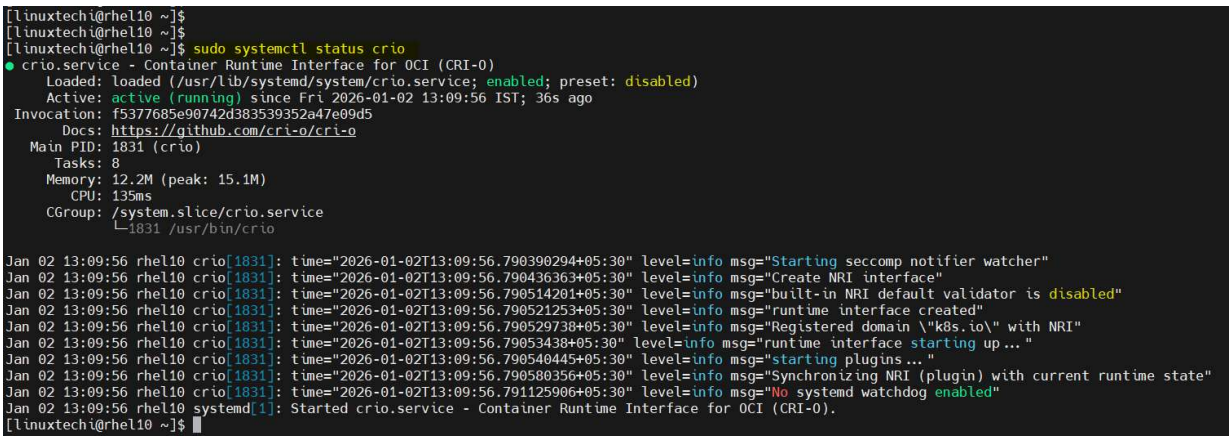


Once installed, start the crio service, run

```
$ sudo systemctl start crio && sudo systemctl enable crio
```

Verify the crio service

```
$ sudo systemctl status crio
```



4) Install CNI Plugins For CRI-O

In this step, we will install CNI and plugins. These plugins are needed for networking foundation that allow crio to create and manage pod networks.

Run the following set of commands.

```
ugins/releases/download/${CNI_VERSION}/cni-plugins-linux-${ARCH}
```

Next create required folder (**/opt/cni/bin/**) and extract it using tar command.

```
$ sudo mkdir -p /opt/cni/bin
$ sudo tar -C /opt/cni/bin -xzf cni-plugins-linux-${ARCH}-${C
```

Move crio bridge conflist file using following mv command.

```
$ sudo mv /etc/cni/net.d/10-crio-bridge.conflist.disabled /et
```

Restart the ciro service to make above changes into the effect.

```
$ sudo systemctl restart crio
```

5) Install CRI-O Tools

Additionally, you must install the cri-tools package, which provides the crictl command-line utility. The crictl tool is essential for inspecting, managing, and troubleshooting pods and containers when working with CRI-O.

For a smooth and error-free setup, always ensure that the crictl version matches your CRI-O version, as version mismatches can lead to unexpected behavior and compatibility issues.

Execute the following set of commands:

```
$ export VERSION="v1.34.0"
$ wget https://github.com/kubernetes-sigs/cri-tools/releases/
$ sudo tar zxvf crictl-$VERSION-linux-amd64.tar.gz -C /usr/lo
$ sudo cp /usr/local/bin/crictl /usr/bin/
```

Verify the crictl version

```
$ sudo crictl --runtime-endpoint unix:///var/run/crio/crio.so
$ crictl --version
```

Also verify the output of crictl info command output, Runtime and Network should be in Ready State.

```
$ sudo crictl info | tail -20
```

On RHEL 10, CRI-O sandbox creation can fail due to systemd eBPF device filtering. The fix is to disable **enable_devices** in **/etc/crio/crio.conf.d/**.

```
$ sudo vi /etc/crio/crio.conf.d/99-disable-ebpf.conf
[crio.runtime]
enable_devices = false
```

save and close the file.

After that restart crio service

```
$ sudo systemctl restart crio
```

6) Test CRIO Installation

In order to test CRIO installation, we will spin up nginx pod. Create the pod configuration file with following content

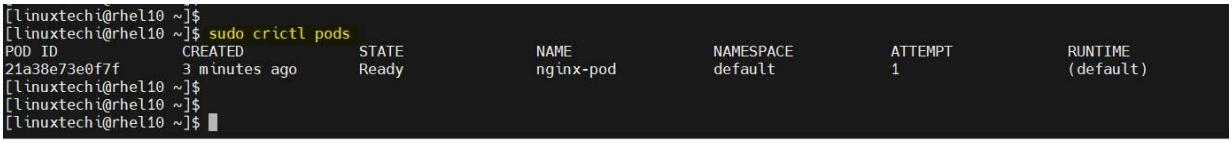
```
$ vi nginx-pod.json
{
  "metadata": {
    "name": "nginx-pod",
    "namespace": "default",
    "attempt": 1,
    "uid": "nginx-pod-uid"
  },
  "linux": {}
}
```

Create the pod sandbox:

```
$ POD_ID=$(sudo crictl runp nginx-pod.json)
```

Check Pod status, run

```
$ sudo crictl pods
```



Next, create the container config file:

```
$ vi nginx-container.json
```

Add following

```
{
  "metadata": {
    "name": "nginx"
  },
  "image": {
    "image": "docker.io/library/nginx:latest"
  },
  "log_path": "nginx.log",
  "linux": {
    "security_context": {
      "privileged": false
    }
  },
  "port_mappings": [
    {
      "container_port": 80,
      "protocol": "TCP"
```

```
        }
      ]
    }
```

Save and close the file.

Now, create the container inside the pod, run

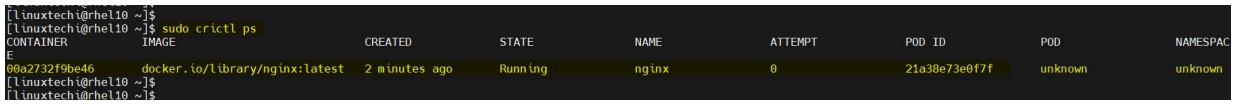
```
$ CONTAINER_ID=$(sudo crictl create --with-pull $POD_ID nginx
```

Next, start the container, run

```
$ sudo crictl start $CONTAINER_ID
```

Verify the container status

```
$ sudo crictl ps
```



Output above confirms that nginx container started successfully inside the pod.

View the container logs, run

```
$ sudo crictl logs $CONTAINER_ID
```

```
[linuxtechi@rhel10 ~]$
[linuxtechi@rhel10 ~]$ sudo crictl logs $CONTAINER_ID
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2026/01/02 08:33:53 [notice] 2#2: using the "epoll" event method
2026/01/02 08:33:53 [notice] 2#2: nginx/1.29.4
2026/01/02 08:33:53 [notice] 2#2: built by gcc 14.2.0 (Debian 14.2.0-19)
2026/01/02 08:33:53 [notice] 2#2: OS: Linux 6.12.0-55.9.1.el10_0.x86_64
2026/01/02 08:33:53 [notice] 2#2: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2026/01/02 08:33:53 [notice] 2#2: start worker processes
2026/01/02 08:33:53 [notice] 2#2: start worker process 25
2026/01/02 08:33:53 [notice] 2#2: start worker process 26
[linuxtechi@rhel10 ~]$
```

Get the pod IP address and try to access application, run the following command

```
$  sudo crictl inspectp --output table 21a38e73e0f7f | head
```

```
[linuxtechi@rhel10 ~]$
[linuxtechi@rhel10 ~]$  sudo crictl inspectp --output table 21a38e73e0f7f | head
ID: 21a38e73e0f7f72db800aee1d4ba1338ecffaf26b7973d62e1950d9d846e60c6
Name: nginx-pod
UID: nginx-pod-uid
Namespace: default
Attempt: 1
Status: SANDBOX_READY
Created: 2026-01-02 13:50:26.965649518 +0530 IST
IP Addresses: 10.85.0.2
Additional IP: 1100:200::2
Labels:
[linuxtechi@rhel10 ~]$
[linuxtechi@rhel10 ~]$
```

Now, run curl command.

```
$ curl -I 10.85.0.2
```

```
[linuxtechi@rhel10 ~]$
[linuxtechi@rhel10 ~]$ curl -I 10.85.0.2
HTTP/1.1 200 OK
Server: nginx/1.29.4
Date: Fri, 02 Jan 2026 08:44:21 GMT
Content-Type: text/html
Content-Length: 615
Last-Modified: Tue, 09 Dec 2025 18:28:10 GMT
Connection: keep-alive
ETag: "69386a3a-267"
Accept-Ranges: bytes

[linuxtechi@rhel10 ~]$
```

Great, output above confirms that we can reach nginx based application.

To clean up the Pod and container, run following commands.

```
$ sudo crictl stop $CONTAINER_ID
$ sudo crictl rm $CONTAINER_ID
$ sudo crictl stopp $POD_ID
$ sudo crictl rmp $POD_ID
```

That's all from this post, I hope you have found it informative and useful, feel free to post your feedback and comments in the below comments section.

About The Author

Pradeep Kumar

I am a Cloud Consultant with over 15 years of experience in Linux, Kubernetes, cloud technologies (AWS, Azure, OpenStack), automation (Ansible, Terraform), and DevOps. I hold certifications like RHCA, CKA, CKAD, CKS, AWS, and Azure.

← Previous Post

Leave A Comment

Your email address will not be published. Required fields are marked *

Type here..

Name*

Email*

Post Comment »

Search...

Join Our Newsletter!

*Name**

*Email Address**

Subscribe

Recent Posts

How to Install CRI-O on RHEL 10

K8s Deployments vs StatefulSets vs DaemonSets Explained

How to Setup Local Yum/DNF Repository on RHEL 10

How to Upgrade RHEL 9 to RHEL 10 Step-by-Step Guide

How to Install LMDE 7 Step-by-Step

Top 10 Things To Do After Installing Debian 13

How to Install Debian 13 Step by Step

How to Install RHEL 10: A Complete Step-by-Step Guide

Top 20 kubectl Commands Every Kubernetes Beginner Must Know

Kubernetes Networking: Services, Ingress and DNS Explained

How to Monitor Kubernetes Using Prometheus and Grafana

How to Make Your Linux Terminal Talk Using espeak-ng