

# Juniper Cloud-Native Router Deployment Guide

Published  
2022-12-28

Juniper Networks, Inc.  
1133 Innovation Way  
Sunnyvale, California 94089  
USA  
408-745-2000  
[www.juniper.net](http://www.juniper.net)

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

*Juniper Cloud-Native Router Deployment Guide*  
Copyright © 2022 Juniper Networks, Inc. All rights reserved.

The information in this document is current as of the date on the title page.

## YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

## END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at <https://support.juniper.net/support/eula/>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

# Table of Contents

1

## Overview

What Is the Juniper® Cloud-Native Router? | 2

System Resource Requirements | 6

2

## Deploy Juniper Cloud-Native Router

Install Juniper Cloud-Native Router | 14

Install Juniper Cloud-Native Router Using Helm Chart | 14

Verify Operation of Containers | 25

Troubleshoot Deployment Issues | 27

Troubleshoot Deployment Issues | 27

View Cloud-Native Router Controller Configuration | 28

View Log Files | 29

3

## Post Deployment

Manage Cloud-Native Router Controller and Cloud-Native Router vRouter | 31

Access the Cloud-Native Router CLIs | 31

Remove the Juniper Cloud-Native Router | 40

Sample Configuration Files | 40

# 1

CHAPTER

## Overview

---

[What Is the Juniper® Cloud-Native Router? | 2](#)

[System Resource Requirements | 6](#)

---

# What Is the Juniper® Cloud-Native Router?

## IN THIS SECTION

- [Overview | 2](#)
- [Benefits | 3](#)
- [Juniper Cloud-Native Router Components | 5](#)

## Overview

Juniper Cloud-Native Router is a container-based software solution that combines the JCNR-controller (cRPD-based control plane) and the JCNR-vRouter. With the cloud-native router, you can enable Junos OS-based routing or switching control with enhanced forwarding capabilities.

JCNR-controller running on a Kubernetes compute-host provides control plane management functionality and uses the routing or forwarding capabilities provided by either the Linux kernel or the JCNR-vRouter.

The Dataplane Development Kit (DPDK) is an open source set of libraries and drivers. DPDK enables fast packet processing by allowing network interface cards (NICs) to send direct memory access (DMA) packets directly into an application's address space. The applications poll for packets, to avoid the overhead of interrupts from the NIC. Integrating with DPDK allows a vRouter to process more packets per second than is possible when the vRouter runs as a kernel module.

In this integrated solution, JCNR-Controller uses gRPC-based services to exchange messages and to communicate with JCNR-vRouter, thus creating the fully functional Cloud-Native Router. This close communication allows you to:

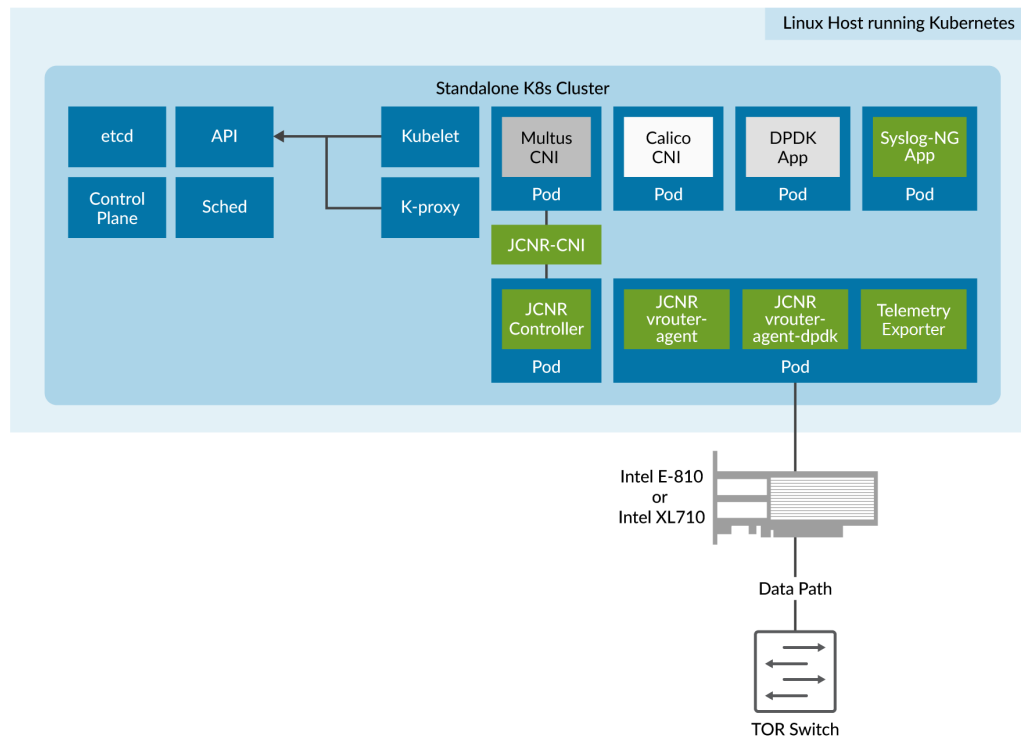
- Learn about fabric and workload interfaces
- Provision DPDK- or kernel-based interfaces for Kubernetes pods as needed
- Configure IPv4 and IPv6 address allocation for Pods
- Install routes into routing tables
- Run routing protocols such as ISIS, BGP, and OSPF

## Benefits

- Higher packet forwarding performance with DPDK-based JCNR-vRouter
- Easy deployment, removal, and upgrade on general purpose compute devices using Helm
- Full routing, switching, and forwarding stacks in software
- Basic L2 functionality, such as MAC learning, MAC aging, MAC limiting, and L2 statistics
- L2 reachability to Radio Units (RU) for management traffic
- L2 or L3 reachability to physical distributed units (DU) such as 5G millimeter wave DUs or 4G DUs
- VLAN tagging
- Bridge domains
- Trunk and access ports
- Supports multiple virtual functions (VF) on Ethernet NICs
- Support for bonded VF interfaces
- Configurable L2 access control lists (ACLs)
- Rate limiting of egress broadcast, unknown unicast, and multicast traffic on fabric interfaces
- IPv4 and IPv6 routing
- Out-of-the-box software-based open radio access network (O-RAN) support
- Quick spin up with containerized deployment

- Highly scalable solution

**Figure 1: Components of Juniper Cloud-Native Router**



## Kubernetes

**NOTE:** Juniper Networks refers to primary nodes and backup nodes. Kubernetes refers to master nodes and worker nodes. References in this guide to primary and backup correlate with master and worker in the Kubernetes world.

Kubernetes is an orchestration platform for running containerized applications in a clustered computing environment. It provides automatic deployment, scaling, networking, and management of containerized applications.

A Kubernetes pod consists of one or more containers, with each pod representing an instance of the application. A pod is the smallest unit that Kubernetes can manage. All containers in the pod share the same network name space.

We rely on Kubernetes to orchestrate the infrastructure that the cloud-native router needs to operate. However, we do not supply Kubernetes installation or management instructions in this documentation.

See <https://kubernetes.io> for Kubernetes documentation. Currently, Juniper Cloud-Native Router requires that the Kubernetes cluster be a standalone cluster, meaning that the Kubernetes primary and backup functions both run on a single node.

## Juniper Cloud-Native Router Components

### Juniper Cloud-Native Router Controller

The JCNR-Controller (cRPD) is the control-plane part of the cloud-native router solution. You use the controller to communicate with the other elements of the cloud-native router. Configuration, policies and rules that you set on the controller at deploy time are communicated to other components, primarily the JCNR-vRouter agent and JCNR-vRouter, for implementation.

For example, firewall filters (ACLs) are supported on cRPD to configure L2 access lists with deny rules. cRPD sends the configuration information to the JCNR-vRouter through the JCNR-vRouter agent.

### Juniper Cloud-Native Router Controller Functionality:

- Exposes Junos OS compatible CLI configuration and operation commands that are accessible to external automation and orchestration systems using the NETCONF protocol.
- Supports JCNR-vRouter as the high-speed forwarding plane. This enables applications that are built using the DPDK framework to send and receive packets directly to the application and the JCNR-vRouter without passing through the kernel
- Support for configuration of VLAN-tagged sub-interfaces on physical function (PF), virtual function (VF), virtio, access, and trunk interfaces managed by the DPDK-enabled JCNR-vRouter
- Supports configuration of bridge domains
- Advertises DPDK application reachability to core network using routing protocols primarily with BGP and IS-IS
- Distributes L3 network reachability information of the pods inside and outside a cluster

### Juniper Cloud-Native Router vRouter

JCNR-vRouter is an alternative to the Linux bridge or the Open vSwitch (OVS) module in the Linux kernel. The pod that houses the JCNR-vRouter container also houses the JCNR-vRouter agent container. JCNR-vRouter functions to:

- Perform L2 forwarding
- Perform L2 rate-limiting
- Allows the use of DPDK-based forwarding



- Enforce L2 access control lists (ACLs)
- Perform routing with Layer 3 virtual private networks

### Juniper Cloud-Native Router-Container Network Interface (JCNR-CNI)

JCNR-CNI is a new CNI developed by Juniper to handle Juniper-developed Pods like JCNR-vRouter agent and JCNR-vRouter agent DPDK, along with DPDK-enabled application Pods and the cloud-native router controller. JCNR-CNI is a Kubernetes CNI plugin installed on each node to provision network interfaces for application pods. During pod creation, Kubernetes delegates pod interface creation and configuration to JCNR-CNI. JCNR-CNI interacts with cRPD and JCNR-vrouter to setup DPDK interfaces. When a Pod is removed, JCNR-CNI is invoked to de-provision the pod interface, configuration, and associated state in Kubernetes and cloud-native router components. JCNR-CNI works with the Multus CNI to add and configure pod interfaces.

JCNR-CNI provides the following functionality:

- Manages the networking tasks in Kubernetes pods such as assigning IP addresses, allocating MAC addresses, and setting up interfaces between the Pod and host in a Kubernetes cluster
- Applies L2 ACLs – The policies are sent to JCNR-vRouter for applying in the data plane
- Acts on Pod events such as add and delete
- Generates cRPD configuration

### Syslog-NG

Juniper Cloud-Native Router uses a syslog-ng Pod to gather event logs from cRPD and vRouter and transform the logs into JSON-based notifications. The notifications are logged to a file and can be accessed from that file.

## System Resource Requirements

Read this section to understand the Linux host requirements for Juniper Cloud-Native Router.

The following tables list the [host system requirements on page 7](#) for installing cloud-native router in L2 mode, [cloud-native router resource requirements on page 11](#), and other [miscellaneous requirements on page 12](#).



Table 1: Cloud-Native Router Host System Requirements *(Continued)*

Component	Release 22.2		Release 22.3		Release 22.4	
	Value/ Version	Notes	Value/Version	Notes	Value/ Version	Notes
Kernel Version	4.18.X	The tested kernel version is 4.18.0-305.rt7.72.el8.x86_64	4.18.X	The tested kernel version is 4.18.0-305.rt7.72.el8.x86_64	RedHat Enterprise Linux (RHEL): 4.18.X Rocky Linux: 4.18.X	The tested kernel version for RHEL is 4.18.0-305.rt7.72.el8.x86_64 The tested kernel version for Rocky Linux is 4.18.0-372.19.1.rt7.176.el8_6.x86_64

Table 1: Cloud-Native Router Host System Requirements *(Continued)*

Component	Release 22.2		Release 22.3		Release 22.4	
	Value/ Version	Notes	Value/Version	Notes	Value/ Version	Notes
NIC	Intel E810 with Firmware 3.20 0x8000d853 1.3146.0		<ul style="list-style-type: none"> <li>Intel E810 with Firmware 3.20 0x8000d853 1.3146.0</li> <li>Intel XL710 with Firmware 8.60</li> </ul>		<ul style="list-style-type: none"> <li>Intel E810 with Firmware 4.00 0x80014411 1.3236.0</li> <li>Intel XL710 with Firmware 9.00 0x8000cead 1.3179.0</li> </ul>	
IAVF driver	Version 4.4.2		Version 4.4.2		Version 4.5.3.1	
ICE_COMMS	Version 1.3.35.0		Version 1.3.35.0		Version 1.3.35.0	
ICE	Version 1.8.3.1.2		Version 1.8.3.1.2	ICE driver is used only with the Intel E810 NIC	Version 1.9.11.9	ICE driver is used only with the Intel E810 NIC

Table 1: Cloud-Native Router Host System Requirements *(Continued)*

Component	Release 22.2		Release 22.3		Release 22.4	
	Value/ Version	Notes	Value/Version	Notes	Value/ Version	Notes
i40e			Version 2.18.9	i40e driver is used only with the Intel XL710 NIC	Version 2.18.9	i40e driver is used only with the Intel XL710 NIC
Kubernetes (K8s)	Version 1.22.2	The tested K8s version is 1.22.4, although 1.22.2 will also work. <b>NOTE:</b> The K8s cluster must be a standalone/all-in-one cluster	Version 1.22.2	The tested K8s version is 1.22.4, although 1.22.2 will also work. <b>NOTE:</b> The K8s cluster must be a standalone/all-in-one cluster	Version 1.22.2	The tested K8s version is 1.22.4, although 1.22.2 will also work. <b>NOTE:</b> The K8s cluster must be a standalone/all-in-one cluster
Calico	Version 3.22.0		Version 3.22.0			
Multus	Version 3.8		Version 3.8			
Helm	3.9.x		3.9.x			

Table 1: Cloud-Native Router Host System Requirements *(Continued)*

Component	Release 22.2		Release 22.3		Release 22.4	
	Value/ Version	Notes	Value/Version	Notes	Value/ Version	Notes
Container-RT	Docker CE 20.10.11		Docker CE 20.10.11			

Table 2: Cloud-Native Router Resource Requirements

Resource	Releases 22.2 and 22.3		Release 22.4	
	Value	Usage Notes	Value	Usage Notes
Data plane forwarding cores	2 physical cores (2p)		2 physical cores (2p)	
Service/Control Cores	0		0	
UIO Driver	VFIO-PCI		VFIO-PCI	
Hugepages (1G)	4 Gi	Add GRUB_CMDLINE_LINUX_D EFAULT values in <b>/etc/default/grub</b> and reboot the host. For example: GRUB_CMDLINE_LINUX_D EFAULT="console=tty1 console=ttyS0 default_hugepagesz=1 G hugepagesz=1G hugepages=64 intel_iommu=on iommu=pt"	4 Gi	Add GRUB_CMDLINE_LINUX_D EFAULT values in <b>/etc/default/grub</b> and reboot the host. For example: GRUB_CMDLINE_LINUX_D EFAULT="console=tty1 console=ttyS0 default_hugepagesz=1 G hugepagesz=1G hugepages=64 intel_iommu=on iommu=pt"
JCNR Controller cores	.5		.5	

Table 2: Cloud-Native Router Resource Requirements (*Continued*)

Resource	Releases 22.2 and 22.3		Release 22.4	
	Value	Usage Notes	Value	Usage Notes
JCNR vRouter Agent cores	.5		.5	

Table 3: Miscellaneous Requirements

Cloud-Native Router Release	Requirement
22.2, 22.3, and 22.4	Enable VLAN driver at system boot
	Enable VFIO-PCI driver at system boot
	Set IOMMU and IOMMU-PT in /etc/default/grub file. For example: GRUB_CMDLINE_LINUX_DEFAULT="console=tty1 console=ttyS0 default_hugepagesz=1G hugepagesz=1G hugepages=64 intel_iommu=on iommu=pt".
	Disable Spoofcheck on VFs allocated to JCNR. For example: ip link set <interfacename> vf 1 spoofcheck off.
	Set trust on VFs allocated to JCNR. For example: ip link set <interfacename> vf 1 trust on

# 2

CHAPTER

## Deploy Juniper Cloud-Native Router

---

[Install Juniper Cloud-Native Router](#) | 14

[Troubleshoot Deployment Issues](#) | 27

---



# Install Juniper Cloud-Native Router

## SUMMARY

The Juniper Cloud-Native Router (cloud-native router) uses the the JCNR-Controller (cRPD-based control plane) and JCNR-CNI to provide control plane capabilities and a container network interface. Juniper Cloud-Native Router uses the DPDK-enabled vRouter to provide high-performance data plane capabilities and Syslog-NG to provide notification functions. This section explains how you can install these components of the Cloud-Native Router.

## IN THIS SECTION

- [Install Juniper Cloud-Native Router Using Helm Chart | 14](#)
- [Verify Operation of Containers | 25](#)

The JCNR-Controller (cRPD) is an initialization container that provides control plane functionality for the cloud-native router. The control plane is responsible for provisioning of the workload and fabric interfaces used in Juniper Cloud-Native Router. It also manages communication with the vRouter-agent and the vRouter itself over a gRPC connection.

The JCNR-CNI is the container network interface that Juniper Cloud-Native Router uses to communicate with physical interfaces on the server and pod and container network interfaces within the installation.

The Juniper Cloud-Native Router Virtual Router (vRouter) is a container application set that provides advanced forwarding plane functionality. It extends the network from the physical routers and switches into a virtual overlay network hosted in the virtualized servers. The Data Plane Development Kit (DPDK) enables the vRouter to process more packets per second than is possible when the vRouter runs as a kernel module.

The Syslog-NG is a container application that allows Juniper Cloud-Native Router to provide notifications to users about events that occur in the cloud-native router deployment.

## Install Juniper Cloud-Native Router Using Helm Chart

Read this section to learn the steps required to load the cloud-native router image components into docker and install the cloud-native router components using Helm charts.

**NOTE:** In the installation sections of this guide, we do not, generally, specify version information when referring to file and directory names. When we do specify the version number in a file or directory name, we are referring to the current (latest) release.

**NOTE:** It is not recommended to deploy Juniper Cloud-Native Router version 22.4 if Kubernetes cpumanager is enabled in your Kubernetes cluster.

As mentioned in the "[System Resource Requirements](#)" on page 6, the Helm package manager for Kubernetes must be installed prior to installing Juniper Cloud-Native Router components.

**NOTE:** We do not provide a specific path into which you must download the package and install the software. Because of this you can copy the commands shown throughout this document and paste them into the CLI of your server.

The high-level overview of Juniper Cloud-Native Router installation is:

1. [Download the software installation package \(tarball\)](#)
2. [Expand the tarball](#)
3. [Change directory to Juniper\\_Cloud\\_Native\\_Router\\_<release number>](#)
4. [Load the image files into Docker](#)
5. [Enter the root password for your host server and your Juniper Cloud-Native Router](#)
6. [Apply the `secrets/jcni-secrets.yaml` to the Kubernetes system](#)

**NOTE:** For an L2 deployment: perform step 7 below and skip step 8.

For an L3 deployment: skip step 7 and perform step 8.

Perform only one of step 7 or step 8.

7. [Edit the `values.yaml` file to suit the needs of your installation](#)
8. [Edit the `values\_L3.yaml`](#)
9. [Install the Juniper Cloud-Native Router](#)

Each high-level procedure listed above is detailed below,

1. Download the tarball, **Juniper\_Cloud\_Native\_Router\_<release-number>.tgz**, to the directory of your choice.

How you get the tarball into a writeable directory on your server is up to you. You must perform the file transfer in binary mode so the compressed tar file will expand properly.

2. Expand the file `Juniper_Cloud_Native_Router_<release-number>.tgz`.

```
tar xzvf Juniper_Cloud_Native_Router_<release-number>.tgz
```

```
x Juniper_Cloud_Native_Router_22.4/
x Juniper_Cloud_Native_Router_22.4/contrail-tools/
x Juniper_Cloud_Native_Router_22.4/contrail-tools/contrail-tools.yaml
x Juniper_Cloud_Native_Router_22.4/images/
x Juniper_Cloud_Native_Router_22.4/images/jcnr-images.tar.gz
x Juniper_Cloud_Native_Router_22.4/helmchart/
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-vrouter/
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-vrouter/values.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-vrouter/templates/
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-vrouter/templates/jcnr-secrets.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-vrouter/templates/
jcnrvrouter_cleanup.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-vrouter/templates/vrouterl3.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-vrouter/templates/_helpers.tpl
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-vrouter/templates/vrouter.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-vrouter/Chart.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-vrouter/README.md
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-vrouter/.helmignore
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-cni/
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-cni/values.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-cni/templates/
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-cni/templates/jcnrl3.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-cni/templates/jcnr-node-config.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-cni/templates/jcnr-
imagepullsecrets.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-cni/templates/jcnr-standalone.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-cni/templates/jcnr.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-cni/templates/jcnr-config.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-cni/templates/_helpers.tpl
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcnr-cni/templates/jcnr-nad.yaml
```

```

x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcni-cni/templates/jcni-cleanup.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcni-cni/templates/jcni-rbac.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcni-cni/files/
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcni-cni/files/jcni-cni-config-l3-
all.tpl
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcni-cni/files/jcni-cni-config-l2-
base.tpl
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcni-cni/files/jcni-cni-config-l3-
base.tpl
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcni-cni/Chart.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcni-cni/README.md
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/jcni-cni/.helmignore
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/syslog-ng/
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/syslog-ng/values.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/syslog-ng/templates/
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/syslog-ng/templates/jcni-secrets.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/syslog-ng/templates/_helpers.tpl
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/syslog-ng/templates/syslog-config.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/syslog-ng/templates/syslog.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/syslog-ng/files/
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/syslog-ng/files/syslog-ng.conf
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/syslog-ng/Chart.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/charts/syslog-ng/.helmignore
x Juniper_Cloud_Native_Router_22.4/helmchart/values.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/values_L3.yaml
x Juniper_Cloud_Native_Router_22.4/helmchart/Chart.yaml
x Juniper_Cloud_Native_Router_22.4/README.md
x Juniper_Cloud_Native_Router_22.4/secrets/
x Juniper_Cloud_Native_Router_22.4/secrets/jcni-secrets.yaml

```

### 3. Change directory to Juniper\_Cloud\_Native\_Router\_<version number>

```
cd Juniper_Cloud_Native_Router_<version number>
```

**NOTE:** All remaining steps in the installation assume that your current working directory is now **Juniper\_Cloud\_Native\_Router\_22.4**.

4. Load the image files into docker from the file, jcnr-images.tar.gz, located in the **Juniper\_Cloud\_Native\_Router\_22.4/images** directory relative to where you expanded the tarball in the previous step.

```
docker load -i images/jcnr-images.tar.gz
```

```
6139044550ba: Loading layer [=====>] 97.38MB/
97.38MB
fc995c5f2e0f: Loading layer [=====>] 10.24kB/
10.24kB
2cfd8c2d4f4a: Loading layer [=====>] 10.24kB/
10.24kB
ab09fd04eee8: Loading layer [=====>] 97.22MB/
97.22MB
Loaded image: enterprise-hub.juniper.net/jcnr-container-prod/contrail-k8s-deployer:R22.4-340
Loaded image: enterprise-hub.juniper.net/jcnr-container-prod/busybox:v4
e4cf89520b9e: Loading layer [=====>] 26.08MB/
26.08MB
fc995c5f2e0f: Loading layer [=====>] 10.24kB/
10.24kB
2cfd8c2d4f4a: Loading layer [=====>] 10.24kB/
10.24kB
74cfd132dd63: Loading layer [=====>] 122.9kB/
122.9kB
ac1d489bb50f: Loading layer [=====>] 46.51MB/
46.51MB
84ff92691f90: Loading layer [=====>] 10.24kB/
10.24kB
Loaded image: enterprise-hub.juniper.net/jcnr-container-prod/contrail-telemetry-
exporter:R22.4-340
a10c3144b352: Loading layer [=====>] 104.6MB/
104.6MB
Loaded image: enterprise-hub.juniper.net/jcnr-container-prod/contrail-init:R22.4-340
537deede1278: Loading layer [=====>] 77MB/
77MB
2531ae692599: Loading layer [=====>] 51.66MB/
51.66MB
Loaded image: enterprise-hub.juniper.net/jcnr-container-prod/contrail-k8s-applier:R22.4-340
24d78dfce51e: Loading layer [=====>] 2.56kB/
2.56kB
16ac99163d4e: Loading layer [=====>] 2.56kB/
```

```

2.56kB
5e8bf69f45ed: Loading layer [=====>] 3.076MB/
3.076MB
97cf204dd4fa: Loading layer [=====>] 60.42MB/
60.42MB
Loaded image: enterprise-hub.juniper.net/jcnr-container-prod/contrail-vrouter-kernel-init-
dpmk:R22.4-340
994393dc58e7: Loading layer [=====>] 5.827MB/
5.827MB
44cd6f21047a: Loading layer [=====>] 266.2kB/
266.2kB
7da972a0d779: Loading layer [=====>] 26.95MB/
26.95MB
d3942a0c8f2d: Loading layer [=====>] 7.68kB/
7.68kB
Loaded image: enterprise-hub.juniper.net/jcnr-container-prod/jcnr-cni:20221220-ce5cad7
Loaded image: enterprise-hub.juniper.net/jcnr-container-prod/syslog-ng:v6
fa2135dabe94: Loading layer [=====>] 94.78MB/
94.78MB
35dcfd80254d: Loading layer [=====>] 20.48kB/
20.48kB
fbb24fe116c3: Loading layer [=====>] 7.68kB/
7.68kB
1cdb02ca7491: Loading layer [=====>] 8.505MB/
8.505MB
ff669c9ec3ff: Loading layer [=====>] 24.58kB/
24.58kB
aaaf492d2211: Loading layer [=====>] 35.33kB/
35.33kB
ea5e63632e17: Loading layer [=====>] 26.62kB/
26.62kB
e6e9a27430e9: Loading layer [=====>] 13.82kB/
13.82kB
789a0773ab92: Loading layer [=====>] 95.23kB/
95.23kB
2a0091b226cc: Loading layer [=====>] 22.02kB/
22.02kB
8830cecede78: Loading layer [=====>] 13.82kB/
13.82kB
e03669bcbfaf: Loading layer [=====>] 9.216kB/
9.216kB
f825984009de: Loading layer [=====>] 461MB/
461MB

```

```

7f965397be5b: Loading layer [=====] 2.048kB/
2.048kB
e2f57c924e6b: Loading layer [=====] 2.56kB/
2.56kB
Loaded image: enterprise-hub.juniper.net/jcnr-container-prod/contrail-tools:R22.4-340
5cf0f971b440: Loading layer [=====] 43.68MB/
43.68MB
ecfd5896b4a5: Loading layer [=====] 68.1kB/
68.1kB
7b3522749e02: Loading layer [=====] 7.102MB/
7.102MB
4082f4b24edd: Loading layer [=====] 405.5kB/
405.5kB
95c3bf2c41a6: Loading layer [=====] 43.06MB/
43.06MB
67ee231fd6bf: Loading layer [=====] 20.22MB/
20.22MB
920d5c9189dd: Loading layer [=====] 204.3kB/
204.3kB
c7a102f767d9: Loading layer [=====] 6.144kB/
6.144kB
56bde89d62f0: Loading layer [=====] 140.2MB/
140.2MB
462729659615: Loading layer [=====] 9.604MB/
9.604MB
12701e87feb8: Loading layer [=====] 145.7MB/
145.7MB
002197a7a73a: Loading layer [=====] 9.216kB/
9.216kB
7f69527f070a: Loading layer [=====] 2.048kB/
2.048kB
ff32ce4577b1: Loading layer [=====] 2.56kB/
2.56kB
Loaded image: enterprise-hub.juniper.net/jcnr-container-prod/contrail-vrouter-
agent:R22.4-340
81518edc73dd: Loading layer [=====] 5.12kB/
5.12kB
d08901c424e8: Loading layer [=====] 721.4kB/
721.4kB
2cc14f52d561: Loading layer [=====] 45.29MB/
45.29MB
2f37cc7572c6: Loading layer [=====] 60.32MB/
60.32MB

```





**NOTE:** You must obtain your license file from your account team and install it in the **secrets.yaml** file as instructed above. Without the proper base64-encoded license file and root password in the **secrets.yaml** file, the cRPD Pod does not enter Running state, but remains in CrashLoopBackOff state.

You must copy the base64 outputs and paste them into the **secrets/jcnp-secrets.yaml** file in the appropriate locations.

6. Apply the **secrets/jcnp-secrets.yaml** to the Kubernetes system

```
kubectl apply -f secrets/jcnp-secrets.yaml
```

```
namespace/jcnp created
secret/jcnp-secrets created
```

**NOTE:** For an L2 deployment: perform step 7 below and skip step 8.  
For an L3 deployment: skip step 7 below and perform step 8.  
Perform only one of step 7 or step 8, not both.

7. For an L2 deployment, edit the **helmchart/jcnp/values.yaml** file.

You must customize the Helm chart for the Juniper Cloud-Native Router installation in L2 mode:

- Choose fabric interfaces—Use interface names from your host system
- Create the VLAN id list for trunk interfaces—Use VLAN ids that fit in your network
- Choose a fabric workload interface—Use interface names from your host system
- Set the VLAN id for traffic on the workload interface
- Set the severity level for JCNP-vRouter logging

**NOTE:** Leave the log\_level set to INFO unless instructed to change it by JTAC.

- Ensure that the mode option is set to "l2"
- Set the cpu core mask—physical cores, logical cores

- Choose the fabric interface–Use interface names from your host system
- Choose a workload interface–Use interface names from your host system
- (Optional) Set a rate limit for broadcast, multicast, and unknown unicast traffic in bytes per second by assigning storm control profiles
- (Optional) Set a core pattern to determine the generated names for core files. If you leave it blank, then cloud-native router pods do not overwrite the existing core pattern
- (Intel 810 NIC only) Enable QoS on the NIC by setting true or false (default is false)
- Set a writeable directory location for syslog-ng to store notifications
- (Optional) If you specify a bond interface as your `fabricInterface:`, provide `slaveInterface` names from your system under the `bondInterfaceConfigs:` section.
- By default `restoreInterface` is set to false. With this setting when vrouter pod crashes or is deleted the interfaces are not restored back to host.

**NOTE:** If you are using the Intel XL710 NIC, you must set `ddp=false` in the `values.yaml`

See ["Sample Configuration Files" on page 40](#) for a commented example of the default `helmchart/values.yaml` file.

8. For an L3 deployment, edit the `helmchart/jcnr/values_L3.yaml` file.

You must customize the Helm chart for the Juniper Cloud-Native Router installation in L3 mode:

- Assign IP addresses to interfaces that you configure in `values_L3.yaml`
- Set the severity level for JCNR-vRouter logging

**NOTE:** Leave the `log_level` set to INFO unless instructed to change it by JTAC.

- Ensure that the mode option is set to "l3"
- Set the cpu core mask–physical cores, logical cores
- (Optional) Set a core pattern to determine the generated names for core files. If you leave it blank, then cloud-native router pods do not overwrite the existing core pattern
- Set a writeable directory location for syslog-ng to store notifications

See ["Sample Configuration Files" on page 40](#) for a commented example of the default `helm_charts/jcnr/values_L3.yaml` file.

9. Deploy the Juniper Cloud-Native Router using Helm

**NOTE:** Starting with Juniper Cloud-Native Router Release 22.4, you must specify either the `values.yaml` or `values_L3.yaml` file when you deploy the cloud-native router. You specify the YAML file in the `helm install` command as shown in the examples below.

For an L2 installation, issue the command:

```
helm install jcnr -f values.yaml .
```

```
NAME: jcnr
LAST DEPLOYED: Mon Dec 19 17:04:12 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

For an L3 installation, issue the command:

```
helm install jcnr -f values_L3.yaml .
```

```
NAME: jcnr
LAST DEPLOYED: Tue Dec 20 11:35:40 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

## 10. Confirm Juniper Cloud-Native Router Deployment

```
helm ls
```

NAME	NAMESPACE	REVISION	UPDATED
STATUS	CHART	APP VERSION	
jcnr	default	1	2022-12-17 16:44:17.472130634 -0700 PDT
deployed	jcnr-22.4.0	22.4.0	

## Verify Operation of Containers

This task allows you to confirm that the Juniper Cloud-Native Router Pods are running.

### 1. `kubectl get pods -A`

The output of the `kubectl` command shows all of the pods in the Kubernetes cluster in all namespaces. Successful deployment means that all pods display that they are in the running state. In this example we have marked the Juniper Cloud-Native Router Pods in **bold**. For example:

```
kubectl get pods -A
```

NAMESPACE	NAME	READY	STATUS
RESTARTS	AGE		
<b>contrail-deploy</b>	<b>contrail-k8s-deployer-7b5dd699b9-nd7xf</b>	<b>1/1</b>	<b>Running</b>
<b>0</b>	<b>41m</b>		
<b>contrail</b>	<b>contrail-vrouter-masters-dfxgm</b>	<b>3/3</b>	<b>Running</b>
<b>0</b>	<b>41m</b>		
<b>default</b>	<b>delete-crpd-dirs--1-6jmxz</b>	<b>0/1</b>	<b>Completed</b>
<b>0</b>	<b>43m</b>		
<b>default</b>	<b>delete-vrouter-dirs--1-645dt</b>	<b>0/1</b>	<b>Completed</b>
<b>0</b>	<b>43m</b>		
<b>jcnr</b>	<b>kube-crpd-worker-ds-8tnf7</b>	<b>1/1</b>	<b>Running</b>
<b>0</b>	<b>41m</b>		
<b>jcnr</b>	<b>syslog-ng-54749b7b77-v24hq</b>	<b>1/1</b>	<b>Running</b>
<b>0</b>	<b>41m</b>		
kube-system	calico-kube-controllers-57b9767bdb-5wbj6	1/1	Running
ago) 129d			2 (92d

kube-system	calico-node-j4m5b	1/1	Running	2 (92d ago)
kube-system	coredns-8474476ff8-fpw78	1/1	Running	2 (92d ago)
kube-system	dns-autoscaler-7f76f4dd6-q5vdp	1/1	Running	2 (92d ago)
kube-system	kube-apiserver-5a5s5-node2	1/1	Running	3 (92d ago)
kube-system	kube-controller-manager-5a5s5-node2	1/1	Running	4 (92d ago)
kube-system	kube-multus-ds-amd64-4zm5k	1/1	Running	2 (92d ago)
kube-system	kube-proxy-l6xm8	1/1	Running	2 (92d ago)
kube-system	kube-scheduler-5a5s5-node2	1/1	Running	4 (92d ago)
kube-system	node-local-dns-6kmg5	1/1	Running	2 (92d ago)

## 2. kubectl get ds -A

Use the kubectl get ds -A command to get a list of daemonset containers.

```
kubectl get ds -A
```

NAMESPACE	NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE
node	SELECTOR					
	AGE					
contrail	contrail-vrouter-masters	1	1	1	1	1
	node-role.kubernetes.io/master=					
	43m					
jcnr	kube-crpd-worker-ds	1	1	1	1	1
	<none>					
	43m					
kube-system	calico-node	1	1	1	1	1
	kubernetes.io/os=linux					
	129d					
kube-system	kube-multus-ds-amd64	1	1	1	1	1
	kubernetes.io/arch=amd64					
	129d					
kube-system	kube-proxy	1	1	1	1	1
	kubernetes.io/os=linux					
	129d					
kube-system	node-local-dns	1	1	1	1	1
	kubernetes.io/os=linux					
	129d					

# Troubleshoot Deployment Issues

## SUMMARY

This topic provides information about how to troubleshoot deployment issues using Kubernetes commands and how to view the cloud-native router configuration files.

## IN THIS SECTION

- [Troubleshoot Deployment Issues | 27](#)
- [View Cloud-Native Router Controller Configuration | 28](#)
- [View Log Files | 29](#)

## Troubleshoot Deployment Issues

This topic provides information on some of the issues that might be seen during deployment of the cloud-native router components and provides a number of Kubernetes (K8s) and shell commands that you run on the host server to help determine the cause of deployment issues.

**Table 4: Investigate Deployment Issues**

Potential issue	What to check	Related Commands
Image not found	Check if registry is accessible, image tags are correct	<ul style="list-style-type: none"><li>● <code>kubectl -n kube-system describe pod &lt;crpd-pod-name&gt;</code></li></ul>
Initialization errors	Check if jcnr-secrets is loaded and has a valid license key	<code>cat /var/run/jcnr/juniper.conf</code>  Confirm that root password and license key are present

Table 4: Investigate Deployment Issues (*Continued*)

Potential issue	What to check	Related Commands
cRPD Pod in CrashLoopBackOff state	<ul style="list-style-type: none"> <li>• Check if startup/liveness probe is failing or vrouter pod not running</li> <li>• rpd-vrouter-agent gRPC connection not UP</li> <li>• Composed configuration is invalid or config template is invalid</li> </ul>	<ul style="list-style-type: none"> <li>• <code>kubectl get pods -A</code></li> </ul> <p><code>kubectl describe pod &lt;crpd-pod-name&gt;</code></p> <ul style="list-style-type: none"> <li>• See <a href="#">"Access the Cloud-Native Router CLIs" on page 31</a> to enter the cRPD CLI and run the following command:</li> </ul> <p><code>show krt state channel vrouter</code></p> <ul style="list-style-type: none"> <li>• <code>cat /var/run/jcnr/juniper.conf</code></li> </ul>
vRouter Pod in CrashLoopBackOff state	Check the contrail-k8s-deployer pod for errors	<p><code>kubectl logs contrail-k8s-deployer-&lt;pod-hash&gt; -n contrail-deploy</code></p>

## View Cloud-Native Router Controller Configuration

The cloud-native router deployment process creates a configuration file for the cloud-native router controller (cRPD) as a result of entries in the **values.yaml** file. You can view this configuration file to see the details of the cRPD configuration. To view the cRPD configuration:

1. Navigate to the **/var/run/jcnr** folder to access the configuration file details.

```
root@server:/var/run/jcnr#ls
```

```
config  containers  juniper.conf  jcnr-crpd-pod.conf
```

2. View the contents of the configuration file.

```
root@server:/var/run/jcnr#vi juniper.conf
```

## View Log Files

In this topic, we use the default `log_path` directory, `/var/log/jcnr/`, and the default `syslog_notifications` directory, `/var/log/jcnr/jcnr-notifications.json`. You can change the location of the log files by changing the value of the `log_path:` or `syslog_notifications:` keys in the `values.yaml` file prior to deployment.

Navigate to the following path and issue the `ls` command to list the log files for each of the cloud-native router components.

```
# cd /var/log/jcnr/
```

```
[root@host: /var/log/jcnr]# ls
```

```
contrail-vrouter-agent.log  contrail-vrouter-dpdk-init.log  contrail-vrouter-dpdk.log  vrouter-  
kernel-init.log  
calico                    containers                    cloud-init.log             contrail                    jcnr-  
cni.log  
cloud-init-output.log    crpd                        pods                      jcnr-notifications.json
```



# 3

CHAPTER

## Post Deployment

---

Manage Cloud-Native Router Controller and Cloud-Native Router vRouter | 31

Sample Configuration Files | 40

---

# Manage Cloud-Native Router Controller and Cloud-Native Router vRouter

## SUMMARY

This topic contains instructions for how to access the cloud-native router CLIs, how to run operational commands in cRPD and vRouter containers, and how to remove cloud-native router.

## IN THIS SECTION

- [Access the Cloud-Native Router CLIs | 31](#)
- [Remove the Juniper Cloud-Native Router | 40](#)

## Access the Cloud-Native Router CLIs

You can access the cloud-native router's CLI to monitor the router's status and to make configuration changes. In this section we provide the commands that you use to access the cRPD and vRouter CLIs and provide some examples of show commands.

Because the cloud-native router controller element runs as a Pod in a Kubernetes (K8s) cluster, you must use K8s commands to access the CLI. We provide an example below. We do not provide specific directory paths in our examples so you can copy and paste the commands into your server.

### Access the Cloud-Native Router Controller (cRPD) CLI

In this example we list all of the K8s Pods running on the K8s host server. We use that output to identify the cRPD Pod that hosts the cloud-native router controller container. We then connect to the CLI of the cloud-native router controller and run some show commands.

### List the K8s Pods Running in the Cluster

```
kubectl get pods -A
```

NAMESPACE	NAME	READY	STATUS
RESTARTS	AGE		
contrail-deploy	contrail-k8s-deployer-7b5dd699b9-nd7xf	1/1	Running
0	41m		
contrail	contrail-vrouter-masters-dfxgm	3/3	Running
0	41m		

default	delete-crpd-dirs--1-6jmxz	0/1	Completed	
0	43m			
default	delete-vrouter-dirs--1-645dt	0/1	Completed	
0	43m			
jcnr	kube-crpd-worker-ds-8tnf7	1/1	Running	
0	41m			
jcnr	syslog-ng-54749b7b77-v24hq	1/1	Running	
0	41m			
kube-system	calico-kube-controllers-57b9767bdb-5wbj6	1/1	Running	2 (92d
ago) 129d				
kube-system	calico-node-j4m5b	1/1	Running	2 (92d
ago) 129d				
kube-system	coredns-8474476ff8-fpw78	1/1	Running	2 (92d
ago) 129d				
kube-system	dns-autoscaler-7f76f4dd6-q5vdp	1/1	Running	2 (92d
ago) 129d				
kube-system	kube-apiserver-5a5s5-node2	1/1	Running	3 (92d
ago) 129d				
kube-system	kube-controller-manager-5a5s5-node2	1/1	Running	4 (92d
ago) 129d				
kube-system	kube-multus-ds-amd64-4zm5k	1/1	Running	2 (92d
ago) 129d				
kube-system	kube-proxy-l6xm8	1/1	Running	2 (92d
ago) 129d				
kube-system	kube-scheduler-5a5s5-node2	1/1	Running	4 (92d
ago) 129d				
kube-system	node-local-dns-6kkg5	1/1	Running	2 (92d
ago) 129d				

The only Pod that has cRPD in its name is the **kube-crpd-worker-ds-npbjq**. Thus, this is the name of the Pod we will use to access the cRPD CLI.

### Connect to the cRPD CLI

The kubectl command that allows access to the controller's CLI has the following form:

```
kubectl exec -n <namespace> -it <cRPD worker Pod name> -- bash
```

In practice, you substitute values from your system for the values contained between angle brackets (<>). For example:

```
kubectl exec -n kube-system -it kube-crpd-worker-ds-8tnf7 -- bash
```

The result of the above command should appear similar to:

```
===>
      Containerized Routing Protocols Daemon (CRPD)
      Copyright (C) 2020-2021, Juniper Networks, Inc. All rights reserved.
                                                    <===

root@ix-jcncr-01:/#
```

At this point, you have connected to the shell of the cloud-native router. Just as with other Junos-based shells, you access the operational mode of the cloud-native router the same way as if you were connected to the console of a physical Junos OS device.

```
root@jcncr-01:/# cli
```

```
root@jcncr-01>
```

### Example Show Commands

In the following examples, we remove the prompt, **root@jcncr-01>**, so you can copy and paste the commands into your system without editing them.

```
show interfaces terse
```

```
__crpd-brd1: flags=67<UP,BROADCAST,RUNNING> mtu 1500
  inet6 fe80::205f:39ff:fe19:87b7 prefixlen 64 scopeid 0x20<link>
  ether 22:5f:39:19:87:b7 txqueuelen 1000 (Ethernet)
  RX packets 0 bytes 0 (0.0 B)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 7 bytes 746 (746.0 B)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

cali502530ac57f: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1480
  inet6 fe80::ecee:eeff:feee:eeee prefixlen 64 scopeid 0x20<link>
  ether ee:ee:ee:ee:ee:ee txqueuelen 0 (Ethernet)
  RX packets 9530538 bytes 816771272 (816.7 MB)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 11502794 bytes 11091296232 (11.0 GB)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```

caliae604977c78: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1480
    inet6 fe80::ecee:eeff:feee:eeee prefixlen 64 scopeid 0x20<link>
    ether ee:ee:ee:ee:ee:ee txqueuelen 0 (Ethernet)
    RX packets 10120320 bytes 904273274 (904.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 9242684 bytes 841165346 (841.1 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:b9:ad:64:ad txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.87.3.138 netmask 255.255.255.128 broadcast 10.87.3.255
    inet6 fe80::3eec:eff:fe4e:145c prefixlen 64 scopeid 0x20<link>
    ether 3c:ec:ef:4e:14:5c txqueuelen 1000 (Ethernet)
    RX packets 10432410 bytes 10076907508 (10.0 GB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3444445 bytes 3877176824 (3.8 GB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device memory 0xaae20000-aae3ffff

eno2: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 3c:ec:ef:4e:14:5d txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device memory 0xaae00000-aae1ffff

enp59s0f0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 40:a6:b7:2a:86:78 txqueuelen 1000 (Ethernet)
    RX packets 25596 bytes 5412132 (5.4 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6 bytes 660 (660.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp59s0f1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 40:a6:b7:2a:86:79 txqueuelen 1000 (Ethernet)

```

```

RX packets 554 bytes 116931 (116.9 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 7 bytes 770 (770.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp59s0f1v1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
ether 72:e4:ae:81:4a:b3 txqueuelen 1000 (Ethernet)
RX packets 7 bytes 2048 (2.0 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 612 bytes 107701 (107.7 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth0: flags=195<UP,BROADCAST,RUNNING,NOARP> mtu 1500
inet 169.254.93.210 netmask 255.255.255.255 broadcast 0.0.0.0
inet6 fe80::58b2:7fff:fea9:3adb prefixlen 64 scopeid 0x20<link>
ether 5a:b2:7f:a9:3a:db txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1 bytes 70 (70.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

gre0: flags=193<UP,RUNNING,NOARP> mtu 1476
unspec 00-00-00-00-30-30-30-3A-00-00-00-00-00-00-00-00 txqueuelen 1000 (UNSPEC)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ip6tnl0: flags=193<UP,RUNNING,NOARP> mtu 1452
inet6 fe80::d4c4:b5ff:fede:df84 prefixlen 64 scopeid 0x20<link>
unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 1000 (UNSPEC)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

irb: flags=67<UP,BROADCAST,RUNNING> mtu 1500
inet6 fe80::f4a0:c8ff:fee6:a28c prefixlen 64 scopeid 0x20<link>
ether f6:a0:c8:e6:a2:8c txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

```

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 4885871889 bytes 578917760800 (578.9 GB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4885871889 bytes 578917760800 (578.9 GB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lsi: flags=67<UP,BROADCAST,RUNNING> mtu 1500
    inet6 fe80::4fd:d0ff:fe4e:943b prefixlen 64 scopeid 0x20<link>
    ether 06:fd:d0:4e:94:3b txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4 bytes 440 (440.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

sit0: flags=193<UP,RUNNING,NOARP> mtu 1480
    inet6 ::127.0.0.1 prefixlen 96 scopeid 0x90<compat,host>
    inet6 ::172.17.0.1 prefixlen 96 scopeid 0x80<compat,global>
    inet6 ::169.254.93.210 prefixlen 96 scopeid 0x80<compat,global>
    inet6 ::10.87.3.138 prefixlen 96 scopeid 0x80<compat,global>
    sit txqueuelen 1000 (IPv6-in-IPv4)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

tunl0: flags=193<UP,RUNNING,NOARP> mtu 1480
    inet 10.233.90.0 netmask 255.255.255.255
    tunnel txqueuelen 1000 (IPIP Tunnel)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0

```

```
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
show configuration routing-instances
```

```
vswitch {
  instance-type virtual-switch;
  bridge-domains {
    bd100 {
      vlan-id 100;
    }
    bd200 {
      vlan-id 200;
    }
    bd300 {
      vlan-id 300;
    }
    bd700 {
      vlan-id 700;
      interface enp59s0f1v0;
    }
    bd701 {
      vlan-id 701;
    }
    bd702 {
      vlan-id 702;
    }
    bd703 {
      vlan-id 703;
    }
    bd704 {
      vlan-id 704;
    }
    bd705 {
      vlan-id 705;
    }
  }
  interface bond0;
}
```



### Access the Cloud-Native Router vRouter CLI

In this example we list all of the K8s Pods running on the K8s host server. We use that output to identify the vRouter Pod that hosts the cloud-native router vrouter-agent container. We then connect to the CLI of the vRouter-agent and run a show command to list the available interfaces.

#### List the K8s Pods Running in the Cluster

```
kubectl get pods -n contrail
```

NAME	READY	STATUS	RESTARTS	AGE
contrail-vrouter-masters-dfxgm	3/3	Running	0	79m

### Connect to the Cloud-Native Router vRouter CLI

The kubectl command that allows access to the controller's CLI has the following form:

```
kubectl exec -n contrail -it <contrail-vrouter-masters-pod> -- bash
```

In practice, you substitute values from your system for the values contained between angle brackets (<>). For example:

```
kubectl exec -n contrail -it contrail-vrouter-masters-xnwwp -- bash
```

The output of this command should look similar to:

```
root@jcnr-01:/#
```

At this point, you have connected to the vRouter's CLI. You can run commands in the CLI to learn about the state of the vRouter. For example, the command shown below allows you to see which interfaces are present on the vRouter.

```
root@jcnr-01:/# vif --list
```

```
Vrouter Operation Mode: PureL2
```

```
Vrouter Interface Table
```

```
Flags: P=Policy, X=Cross Connect, S=Service Chain, Mr=Receive Mirror
       Mt=Transmit Mirror, Tc=Transmit Checksum Offload, L3=Layer 3, L2=Layer 2
       D=DHCP, Vp=Vhost Physical, Pr=Promiscuous, Vnt=Native Vlan Tagged
```

Mnp=No MAC Proxy, Dpdk=DPDK PMD Interface, Rfl=Receive Filtering Offload,  
 Mon=Interface is Monitored  
 Uuf=Unknown Unicast Flood, Vof=VLAN insert/strip offload, Df=Drop New Flows, L=MAC  
 Learning Enabled  
 Proxy=MAC Requests Proxied Always, Er=Etree Root, Mn=Mirror without Vlan Tag,  
 HbsL=HBS Left Intf  
 HbsR=HBS Right Intf, Ig=Igmp Trap Enabled, Ml=MAC-IP Learning Enabled, Me=Multicast  
 Enabled

```
vif0/0      Socket: unix
             Type:Agent HWaddr:00:00:5e:00:01:00
             Vrf:65535 Flags:L2 QOS:-1 Ref:3
             RX queue errors to lcore 0 0 0 0 0 0 0 0 0 0 0
             RX packets:0 bytes:0 errors:0
             TX packets:11 bytes:4169 errors:0
             Drops:0

vif0/1      PCI: 0000:00:00.0 (Speed 25000, Duplex 1)
             Type:Physical HWaddr:46:37:1f:de:df:bc
             Vrf:65535 Flags:L2Vof QOS:-1 Ref:8
             RX queue errors to lcore 0 0 0 0 0 0 0 0 0 0 0
             Fabric Interface: eth_bond_bond0 Status: UP Driver: net_bonding
             Slave Interface(0): 0000:3b:02.0 Status: UP Driver: net_iavf
             Slave Interface(1): 0000:3b:02.1 Status: UP Driver: net_iavf
             Vlan Mode: Trunk Vlan: 100 200 300 700-705
             RX packets:0 bytes:0 errors:0
             TX packets:378 bytes:81438 errors:0
             Drops:0

vif0/2      PCI: 0000:3b:0a.0 (Speed 25000, Duplex 1)
             Type:Workload HWaddr:ba:69:c0:b7:1f:ba
             Vrf:0 Flags:L2Vof QOS:-1 Ref:7
             RX queue errors to lcore 0 0 0 0 0 0 0 0 0 0 0
             Fabric Interface: 0000:3b:0a.0 Status: UP Driver: net_iavf
             Vlan Mode: Access Vlan Id: 700 OVlan Id: 700
             RX packets:378 bytes:81438 errors:2
             TX packets:0 bytes:0 errors:0
             Drops:391
```

## Remove the Juniper Cloud-Native Router

We do not provide specific directory names for the commands in this topic. This allows you to copy and paste the commands from this document onto your server.

Uninstall the Juniper Cloud-Native Router.

```
helm uninstall jcnr
```

## Sample Configuration Files

Read this section to find sample YAML configuration files for use when you deploy Juniper Cloud-Native Router. These YAML files control the features and functions available to cloud-native router by affecting the deployment instructions. YAML files for workload configuration are also included. The workload configuration files control the workload functions.

We've included the following sample configuration files:

- **Juniper Cloud-Native Router Main Configuration File**
  - • main ["values.yaml" on page 41](#) file
- **Juniper Cloud-Native Router Main L3 Configuration File**
  - main ["values\\_L3.yaml" on page 43](#) file
- **Juniper Cloud-Native Router vRouter-Specific Configuration File**
  - ["jcnr-vrouter specific values.yaml file" on page 47](#)
- **Juniper Cloud-Native Router JCNR-CNI-Specific Configuration File**
  - ["jcnr-cni specific values.yaml file" on page 51](#)
- **L2 Workload Configuration Files**
  - • ["nad-dpdk\\_trunk\\_vlan\\_3002.yaml" on page 53](#)
  - • [" nad-kernel\\_access\\_vlan\\_3001.yaml" on page 54](#)
  - • [" nad-odu-bd3003-sub.yaml" on page 55](#)
  - • ["nad-odu-bd3004-sub.yaml" on page 56](#)
  - • ["odu-virtio-subinterface.yaml" on page 57](#)

- ["pod-dpdk-trunk-vlan3002.yaml " on page 58](#)
- ["pod-kernel-access-vlan-3001.yaml" on page 59](#)
- L3 Workload Configuration Files
  - ["L3\\_nad-net1.yaml" on page 60](#)
  - ["L3\\_odu1.yaml" on page 61](#)

Use these files to understand the configuration options available for deployment of Juniper Cloud-Native Router. The workload configuration files display how you can configure trunk and access interfaces and configure various VLANs for each type. Each of the files contain comments that start with a hash mark (#) and are highlighted in **bold** in these examples.

- values.yaml

This is the main **values.yaml** file. There are 3 other values.yaml files supplied in the TAR file. 1 **values.yaml** for each of the installation components: **jcnr-cni**, **jcnr-vrouter**, and **syslog-ng**.

If there are conflicting settings between the individual **values.yaml** files and the main **values.yaml** file, the settings in the main **values.yaml** file take precedence.

```
#####
#           Common Configuration (global vars)           #
#####
global:
  registry: svl-artifactory.juniper.net/
  # uncomment below if all images are available in the same path; it will
  # take precedence over "repository" paths under "common" section below
  #repository: path/to/allimages/

  # uncomment below if you are using a private registry that needs authentication
  # registryCredentials - Base64 representation of your Docker registry credentials
  # secretName - Name of the Secret object that will be created
  #imagePullSecret:
    #registryCredentials:
    #secretName: regcred

  common:
    vrouter:
      repository: atom-docker/cn2/bazel-build/dev/
      tag: R22.4-340
    crpd:
      repository: junos-docker-local/warthog/
```

```

    tag: 22.4R1.10
  jcnrcni:
    repository: junos-docker-local/warthog/
    tag: 20221220-ce5cad7

# defines the log severity. Possible options: DEBUG, INFO, WARN, ERR
log_level: "INFO"

# "log_path": this directory will contain various jcnr related descriptive logs
# such as contrail-vrouter-agent.log, contrail-vrouter-dpdk.log etc.
log_path: "/var/log/jcnr/"
# "syslog_notifications": absolute path to the file that will contain syslog-ng
# generated notifications in json format
syslog_notifications: "/var/log/jcnr/jcnr_notifications.json"

# mode in which jcnr will operate; possible options include "l2" or "l3"
mode: "l2"

#####
#                               L2 PARAMS                               #
#####

# fabricInterface: NGDU or tor side interface, expected all types
# of traffic; interface_mode is always trunk for this mode
fabricInterface:
- bond0:
    interface_mode: trunk
    vlan-id-list: [100, 200, 300, 700-705]
    storm-control-profile: rate_limit_pf1

# fabricWorkloadInterface: RU side interfaces, expected traffic is only
# management/control traffic; interface mode is always access for this mode
fabricWorkloadInterface:
- enp59s0f1v0:
    interface_mode: access
    vlan-id-list: [700]

jcnr-vrouter:
# restoreInterfaces: setting this to true will restore the interfaces
# back to their original state in case vrouter pod crashes or restarts
restoreInterfaces: false

# bond interface configurations

```

```

bondInterfaceConfigs:
  - name: "bond0"
    mode: 1          # ACTIVE_BACKUP MODE
    slaveInterfaces:
      - "enp59s0f0v0"
      - "enp59s0f0v1"

# MTU for all physical interfaces( all VF's and PF's)
mtu: "9000"

# vrouter fwd core mask
# if qos is enabled, you will need to allocate 4 CPU cores (primary and siblings)
cpu_core_mask: "2,3,22,23"

# rate limit profiles for bum traffic on fabric interfaces in bytes per second
stormControlProfiles:
  rate_limit_pf1:
    bandwidth:
    level: 0
  #rate_limit_pf2:
  # bandwidth:
  # level: 0

# Set ddp to true to enable Dynamic Device Personalization (DDP)
# It provides datapath optimization at NIC for traffic like GTPU, SCTP etc.
ddp: true

# Set true/false to Enable or Disable QOS, note: QOS is not supported on X710 NIC.
qosEnable: false

# core pattern to denote how the core file will be generated
# if left empty, JCNr pods will not overwrite the default pattern
corePattern: ""

# path for the core file; vrouter considers /var/crashes as default value if not specified
coreFilePath: /var/crash

```

- values\_L3.yaml

The file, values\_L3.yaml controls installation and operation parameters of the cloud-native router when you deploy in L3 mode.

Note that there are common configuration parameters that exist in both `values_L3.yaml` and the main `values.yaml`. Any values not set in `values_L3.yaml` are taken from the common configuration parameters section of the main `values.yaml` file.

```
# This is a sample values.yaml file to install JCNr in L3 mode
# Install by overriding values.yaml with this file e.g.
# helm install jcnr -f values_L3.yaml
# Please note the overriding file does not replace values.yaml i.e. any values
# that are not present in this file will be taken from the original values.yaml
# e.g. if global.repository is commented in values_L3.yaml and uncommented in
# values.yaml, then the value in values.yaml is still considered
#
#####
#           Common Configuration (global vars)           #
#####
global:
  registry: enterprise-hub.juniper.net/
  # uncomment below if all images are available in the same path; it will
  # take precedence over "repository" paths under "common" section below
  repository: jcnr-container-prod/

  # uncomment below if you are using a private registry that needs authentication
  # registryCredentials - Base64 representation of your Docker registry credentials
  # secretName - Name of the Secret object that will be created
  #imagePullSecret:
    #registryCredentials:
    #secretName: regcred

common:
  vrtrouter:
    #repository: atom-docker/cn2/bazel-build/dev/
    tag: R22.4-340
  crpd:
    #repository: junos-docker-local/warthog/
    tag: 22.4R1.10
  jcnrcni:
    #repository: junos-docker-local/warthog/
    tag: 20221220-ce5cad7

# defines the log severity. Possible options: DEBUG, INFO, WARN, ERR
log_level: "INFO"
```

```

# "log_path": this directory will contain various jcnr related descriptive logs
# such as contrail-vrouter-agent.log, contrail-vrouter-dpdk.log etc.
log_path: "/var/log/jcnr/"
# "syslog_notifications": absolute path to the file that will contain syslog-ng
# generated notifications in json format
syslog_notifications: "/var/log/jcnr/jcnr_notifications.json"

# mode in which jcnr will operate; possible options include "l2" or "l3"
mode: "l3"

jcnr-vrouter:
  # vrouter fwd core mask
  cpu_core_mask: "2,3"

  # set multinode to true if you have more than one node in your Kubernetes cluster
  # (master + worker) and you want to run vrouter in both master and worker nodes
  #multinode: false

  # nodeSelector can be given as a key value pair for vrouter to install on the specific
  # nodes, we can give multiple key value pair.
  # Example: nodeSelector: {key1: value1}
  #nodeSelector:
  #  key1: value1
  #  key2: value2

  #nodeSelector: {}

  # contrail vrouter vhost0 binding interface on the host
  vrouter_dpdk_physical_interface: "eth2"

  # uio driver will be vfio-pci or uio_pci_generic
  vrouter_dpdk_uio_driver: "vfio-pci"

  vhost_interface_ipv4: ""

  vhost_interface_ipv6: ""

  # vrouter gateway IP for IPv4
  vhost_gateway_ipv4: "" # if gateway IP is not provided vrouter will pickup the gateway
  # IP from kernel table

  # vrouter gateway IP for IPv6
  vhost_gateway_ipv6: "" # if gateway IP is not provided vrouter will pickup the gateway IP

```



```

from kernel table

# core pattern to denote how the core file will be generated
# if left empty, JCNr pods will not overwrite the default pattern
corePattern: ""

# path for the core file; vrouters considers /var/crashes as default value if not specified
coreFilePath: /var/crash

jcnr-cni:
  #data plane default is dpdk for vrouters case, linux for kernel module
  dataplane: dpdk

  # only for development environment where master and worker on a single node, then we need
  # to give true
  standalone: false

  # if cRPD needs to be running on the master node as RR (Route Reflector) then we need to
  # enable this file.
  cRPD_RR:
    enabled: false

  networkAttachmentDefinitionName: jcnr # default NAD name and VRF name will be Platter, if
  # we change the name, NAD and VRF will be created on the new Name
  # Pod yaml we need to give the NAD name and VRF name as above

  vrfTarget: 10:10 # vrfTarget used for the default NAD

  #JCNr case, Calico running with default BGP port 179, then for cRPD BGP port have to be
  #different, change the port to 178
  BGPListenPort: 178

  # if cRPD connects to MX or some other router, then we have to leave this port to 179 by
  # default, MX wants to connect to jcnr then MX to cRPD BGP port has to be configured as 178
  BGPConnectPort: 179

  # If master node is used as a RR, then this address should be matched with master node ipv4
  # loopback address.
  BGPIPV4Neighbor: 10.1.1.2

  # If master node is used as a RR, then this address should be matched with master node ipv6
  # loopback address.
  BGPIPV6Neighbor: 2001:db8:abcd::2

```

```
SRGBStartLabel: "400000"
```

```
SRGBIndexRange: "4000"
```

# we can add multiple master nodes configuration by copying the below node configuration as many times as nodes, have the unique name based on the node host name,

# Name format node-**<actual-node-name>**.json with unique IP Address

masterNodeConfig:

```
node-masternode1.json: |
{
  "ipv4LoopbackAddr": "100.1.1.2",
  "ipv6LoopbackAddr": "abcd::2",
  "isoLoopbackAddr": "49.0004.1000.0000.0000.00",
  "srIPv4NodeIndex": "2002",
  "srIPv6NodeIndex": "3002"
}
```

# we can add multiple worker nodes configuration by copying the below node configuration as many times as nodes, have the unique name based on the node host name,

# Name format node-**<actual-node-name>**.json with unique IP Address

workerNodeConfig:

```
node-workernode1.json: |
{
  "ipv4LoopbackAddr": "100.1.1.3",
  "ipv6LoopbackAddr": "abcd::3",
  "isoLoopbackAddr": "49.0004.1000.0000.0001.00",
  "srIPv4NodeIndex": "2003",
  "srIPv6NodeIndex": "3003"
}
```

- jcnr-vrouter specific values.yaml

This **values.yaml** file is specific to the jcnr-vrouter Pod. It is located under the **Juniper\_Cloud\_Native\_Router\_<release-number>/helmchartcharts/jcnr-vrouter** directory. If you enter any values in this file that conflict with values in the main **values.yaml** file, the values in the main **values.yaml** file take precedence.

```
# # This is a YAML-formatted file.
# # Declare variables to be passed into your templates.
```

```
common:
```

```

registry: svl-artifactory.juniper.net/
repository: atom-docker/cn2/bazel-build/dev/

# anchor tag for vrouter container images
vrouter-tag: &vrouter_tag JCNr-22.3-6

contrail_init:
  image: contrail-init
  tag: *vrouter_tag
  pullPolicy: IfNotPresent

contrail_vrouter_kernel_init_dpdk:
  image: contrail-vrouter-kernel-init-dpdk
  tag: *vrouter_tag
  pullPolicy: IfNotPresent

contrail_vrouter_agent:
  image: contrail-vrouter-agent
  tag: *vrouter_tag
  pullPolicy: IfNotPresent

contrail_vrouter_agent_dpdk:
  image: contrail-vrouter-dpdk
  tag: *vrouter_tag
  pullPolicy: IfNotPresent
  resources:
    limits:
      memory: 4Gi
      hugepages-1Gi: 4Gi          # Hugepages must be enabled with default size as 1G;
minimum 4Gi to be used
    requests:
      memory: 4Gi
      hugepages-1Gi: 4Gi

contrail_vrouter_telemetry_exporter:
  image: contrail-telemetry-exporter
  tag: *vrouter_tag
  pullPolicy: IfNotPresent

contrail_k8s_deployer:
  image: contrail-k8s-deployer
  tag: *vrouter_tag
  pullPolicy: IfNotPresent

```

```

contrail_k8s_crdloader:
  image: contrail-k8s-crdloader
  tag: *vrouter_tag
  pullPolicy: IfNotPresent

contrail_k8s_applier:
  image: contrail-k8s-applier
  tag: *vrouter_tag
  pullPolicy: IfNotPresent

busyBox:
  image: busybox
  tag: "latest"
  pullPolicy: IfNotPresent

vrouter_name: master

# uio driver will be vfio-pci or uio_pci_generic
vrouter_dpdk_uio_driver: "vfio-pci"

# MTU for all physical interfaces( all VF's and PF's)
mtu: "9000"

vrouter_log_path: "/var/log/jcnr/"

# Defines the log severity. Possible options: DEBUG, INFO, WARN, ERR
log_level: "INFO"

dpdkCommandAdditionalArgs: "--yield_option 0"

# Set ddp to true to enable Dynamic Device Personalization (DDP)
# It provides datapath optimization at NIC for traffic like GTPU, SCTP etc.
ddp: true

# vrouter fwd core mask
cpu_core_mask: "2,3"

# vrouter service thread mask
service_core_mask: ""

# vrouter control thread mask
dpdk_ctrl_thread_mask: ""

```

```

#
dpdk_mem_per_socket: "1024"

# L3 disabled for switching mode
jcnr_mode: "l2_only"

# global Mac table size - We recommend leaving this at the default value
mac_table_size: "10240"

# timeout (seconds) for aging Mac table entries (S)
mac_table_ageout: 60

# parameters for vRouter livenessProbe
livenessProbe:
  initialDelaySeconds: 10
  periodSeconds: 20
  timeoutSeconds: 5
  failureThreshold: 3
  successThreshold: 1

# parameters for vRouter startupProbe
startupProbe:
  initialDelaySeconds: 10
  periodSeconds: 20
  timeoutSeconds: 5
  failureThreshold: 3
  successThreshold: 1

# setting this to true will restore the interfaces back to
# their original state in case vrouter pod crashes or restarts
restoreInterfaces: false

# tor side interface, expected all types of traffic
fabricInterface:
- enp4s0f0vf0
- bond0

# RU side interfaces, expected traffic is only management/control traffic
fabricWorkloadInterface:
- enp4s0f1vf0

# bond interface configurations

```

```

bondInterfaceConfigs:
  - name: "bond0"
    mode: 1                # ACTIVE_BACKUP MODE
    slaveInterfaces:
      - "enp1s0f1"
      - "enp2s0f1"

# rate limit for broadcast/multicast traffic on fabric interfaces in bytes per second
fabricBMCastRateLimit: 0

```

- jcnr-cni specific values.yaml

This **values.yaml** file is specific to the jcnr-cni Pod. The jcnr-cni specific values.yaml file is located under the **Juniper\_Cloud\_Native\_Router\_<release-number>/helm\_charts/jcnr/charts/jcnr-cni** directory. If you enter any values in this file that conflict with values in the main **values.yaml** file, the values in the main **values.yaml** file take precedence.

```

# Default values for jcnr.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

common:
  registry: svl-artifactory.juniper.net/
  repository: junos-docker-local/warthog/

crpdImage:
  image: crpd
  tag: "22.3R1.8"
  pullPolicy: IfNotPresent

jcnrCNIImage:
  image: jcnr-cni
  tag: "20220918-fadf886"
  pullPolicy: IfNotPresent

crpdConfigGeneratorImage:
  image: crpdconfig-generator
  tag: "v3"
  pullPolicy: IfNotPresent

busyBox:
  image: busybox

```

```

tag: "latest"
pullPolicy: IfNotPresent

#data plane default is dpdk for vrouter case, linux for kernel module
dataplane: dpdk

networkAttachmentDefinitionName: vswitch

crpd_log_path: "/var/log/jcni/"

# Defines the log severity. Possible options: panic, fatal, error,
# warn or warning, info, debug, trace

log_level: "info"

# parameters for cRPD livenessProbe
livenessProbe:
  initialDelaySeconds: 5
  periodSeconds: 10
  timeoutSeconds: 5
  failureThreshold: 3
  successThreshold: 1

# parameters for cRPD startupProbe
startupProbe:
  initialDelaySeconds: 5
  periodSeconds: 10
  timeoutSeconds: 5
  failureThreshold: 3
  successThreshold: 1

crpdConfigs:
  interface_groups:
    fabricInterface: # TOR side interface, expected all types of traffic
      - bond0:
          interface_mode: trunk # interface mode is always trunk for fabricInterface
          vlan-id-list: [100, 200, 700] # vlan-id-lists
      - enp4s0f0vf0:
          interface_mode: trunk # interface mode is always trunk for fabricInterface
          vlan-id-list: [300, 500, 3001, 3002] # vlan-id-lists
      - enp4s0f0vf1:
          interface_mode: trunk # interface mode is always trunk for fabricInterface

```

```

    vlan-id-list: [3003, 3004, 3201-3250, 900] # vlan-id-lists
  - enp4s0f0vf2:
    interface_mode: trunk # interface mode is always trunk for fabricInterface
    vlan-id-list: [3251-3255] # vlan-id-lists
    fabricWorkloadInterface: # RU side interfaces, expected traffic is only management/
control traffic
  - enp4s0f1vf0:
    interface_mode: access # interface mode is always access for fabricWorkloadInterface
    vlan-id-list: [700] # vlan-id-list must always be a single value for
fabricWorkloadInterface
  - enp4s1f1vf0:
    interface_mode: access # interface mode is always access for fabricWorkloadInterface
    vlan-id-list: [900] # vlan-id-list must always be a single value for
fabricWorkloadInterface

routing_instances:
  - vswitch:
    instance-type: virtual-switch

```

- nad-dpdk\_trunk\_vlan\_3002.yaml

```

apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: nad-vswitch-bd3002
spec:
  config: '{
    "cniVersion": "0.4.0",
    "name": "nad-vswitch-bd3002",
    "capabilities": {"ips": true},
    "plugins": [
      {
        "type": "jcnr",
        "args": {
          "instanceName": "vswitch",
          "instanceType": "virtual-switch",
          "bridgeDomain": "bd3002",
          "bridgeVlanId": "3002",
          "dataplane": "dpdk",
          "mtu": "9000"
        }
      }
    ]
  },

```



```

    "ipam": {
      "type": "static",
      "capabilities":{"ips":true},
      "addresses":[
        {
          "address":"2001:db8:3002::10.2.0.1/64",
          "gateway":"2001:db83002::10.2.0.254"
        },
        {
          "address":"10.2.0.1/24",
          "gateway":"10.2.0.254"
        }
      ]
    },
    "kubeConfig":"/etc/kubernetes/kubelet.conf"
  }
]
}'

```

- nad-kernel\_access\_vlan\_3001.yaml

```

apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: pod1-vswitch-bd3001-1
spec:
  config: '{
    "cniVersion":"0.4.0",
    "name": "pod1-vswitch-bd3001-1",
    "capabilities":{"ips":true},
    "plugins": [
      {
        "type": "jcnr",
        "args": {
          "instanceName": "vswitch",
          "instanceType": "virtual-switch",
          "bridgeDomain": "bd3001",
          "bridgeVlanId": "3001",
          "dataplane": "dpdk",
          "mtu": "9000",
          "interfaceType": "veth"

```

```

    },
    "ipam": {
      "type": "static",
      "capabilities":{"ips":true},
      "addresses":[
        {
          "address":"2001:db8:3001::10.1.0.1/64",
          "gateway":"2001:db8:3001::10.1.0.254"
        },
        {
          "address":"10.1.0.1/24",
          "gateway":"10.1.0.254"
        }
      ]
    },
    "kubeConfig":"/etc/kubernetes/kubelet.conf"
  }
]
}'

```

- nad-odu-bd3003-sub.yaml

```

apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: vswitch-bd3003-sub
spec:
  config: '{
    "cniVersion":"0.4.0",
    "name": "vswitch-bd3003-sub",
    "capabilities":{"ips":true},
    "plugins": [
      {
        "type": "jcnr",
        "args": {
          "instanceName": "vswitch",
          "instanceType": "virtual-switch",
          "bridgeDomain": "bd3003",
          "bridgeVlanId": "3003",
          "parentInterface": "net1",
          "interface": "net1.3003",

```

```

        "dataplane": "dpdk"
    },
    "ipam": {
        "type": "static",
        "capabilities": {"ips": true},
        "addresses": [
            {
                "address": "10.3.0.1/24",
                "gateway": "10.3.0.254"
            },
            {
                "address": "2001:db8:3003::10.3.0.1/120",
                "gateway": "2001:db8:3003::10.3.0.1"
            }
        ]
    },
    "kubeConfig": "/etc/kubernetes/kubelet.conf"
}
]
}'

```

- nad-odu-bd3004-sub.yaml

```

apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: vswitch-bd3004-sub
spec:
  config: '{
    "cniVersion": "0.4.0",
    "name": "vswitch-bd3004-sub",
    "capabilities": {"ips": true},
    "plugins": [
      {
        "type": "jcnr",
        "args": {
          "instanceName": "vswitch",
          "instanceType": "virtual-switch",
          "bridgeDomain": "bd3004",
          "bridgeVlanId": "3004",
          "parentInterface": "net1",

```

```

        "interface": "net1.3004",
        "dataplane": "dpdk"

    },
    "ipam": {
        "type": "static",
        "capabilities": {"ips": true},
        "addresses": [
            {
                "address": "30.4.0.1/24",
                "gateway": "30.4.0.254"
            },
            {
                "address": "2001:db8:3004::10.4.0.1/120",
                "gateway": "2001:db8:3004::10.4.0.1"
            }
        ]
    },
    "kubeConfig": "/etc/kubernetes/kubelet.conf"
}
]
}'

```

- odu-virtio-subinterface.yaml

```

apiVersion: v1
kind: Pod
metadata:
  name: odu-subinterface-1
  annotations:
    k8s.v1.cni.cncf.io/networks: |
      [
        {
          "name": "vswitch-bd3003-sub"
        },
        {
          "name": "vswitch-bd3004-sub"
        }
      ]
spec:
  affinity:

```

```

nodeAffinity:
  requiredDuringSchedulingIgnoredDuringExecution:
    nodeSelectorTerms:
      - matchExpressions:
          - key: kubernetes.io/hostname
            operator: In
            values:
              - 5d7s39.englab.juniper.net
containers:
  - name: odu-subinterface
    image: svl-artifactory.juniper.net/junos-docker-local/warthog/pktgen19116:subint
    imagePullPolicy: IfNotPresent
    securityContext:
      privileged: false
    resources:
      requests:
        memory: 2Gi
      limits:
        hugepages-1Gi: 2Gi
    env:
      - name: KUBERNETES_POD_UID
        valueFrom:
          fieldRef:
            fieldPath: metadata.uid
    volumeMounts:
      - name: dpdk
        mountPath: /dpdk
        subPathExpr: $(KUBERNETES_POD_UID)
      - mountPath: /dev/hugepages
        name: hugepage
volumes:
  - name: dpdk
    hostPath:
      path: /var/run/jcnr/containers
  - name: hugepage
    emptyDir:
      medium: HugePages

```

- pod-dpdk-trunk-vlan3002.yaml

```
apiVersion: v1
```

```

kind: Pod
metadata:
  name: odu-trunk-1
  annotations:
    k8s.v1.cni.cncf.io/networks: nad-vswitch-bd3002
spec:
  containers:
    - name: odu-trunk
      image: svl-artifactory.juniper.net/junos-docker-local/warthog/pktgen19116:trunk
      imagePullPolicy: IfNotPresent
      securityContext:
        privileged: true
      resources:
        requests:
          memory: 2Gi
        limits:
          hugepages-1Gi: 2Gi
      env:
        - name: KUBERNETES_POD_UID
          valueFrom:
            fieldRef:
              fieldPath: metadata.uid
      volumeMounts:
        - name: dpdk
          mountPath: /dpdk
          subPathExpr: ${KUBERNETES_POD_UID}
        - mountPath: /dev/hugepages
          name: hugepage
      volumes:
        - name: dpdk
          hostPath:
            path: /var/run/jcnr/containers
        - name: hugepage
          emptyDir:
            medium: HugePages

```

- pod-kernel-access-vlan-3001.yaml

```

apiVersion: v1
kind: Pod
metadata:

```

```

name:   odu-kenel-pod-bd3001-1
annotations:
  k8s.v1.cni.cncf.io/networks: pod1-vswitch-bd3001-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: kubernetes.io/hostname
                operator: In
                values:
                  - 5d8s7.englab.juniper.net
  containers:
    - name: odu-kenel-pod-bd3001-1
      image: vinod-iperf3:latest
      imagePullPolicy: IfNotPresent
      command: ["/bin/bash", "-c", "sleep infinity"]
      securityContext:
        privileged: false
      env:
        - name: KUBERNETES_POD_UID
          valueFrom:
            fieldRef:
              fieldPath: metadata.uid
      volumeMounts:
        - name: dpdk
          mountPath: /dpdk
          subPathExpr: ${KUBERNETES_POD_UID}
  volumes:
    - name: dpdk
      hostPath:
        path: /var/run/jcnr/containers

```

- L3\_nad-net1.yaml

```

apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: net1
spec:
  config: '{

```

```

    "cniVersion": "0.4.0",
    "name": "net1",
    "type": "jcnr",
    "args": {
      "vrfName": "net1",
      "vrfTarget": "1:11"
    },
    "kubeConfig": "/etc/kubernetes/kubelet.conf"
  }'

```

- l3\_odu1.yaml

```

apiVersion: v1
kind: Pod
metadata:
  name: L3-pktgen-odu1
  annotations:
    k8s.v1.cni.cncf.io/networks: |
      [
        {
          "name": "net1",
          "interface": "net1",
          "cni-args": {
            "mac": "aa:bb:cc:dd:ee:51",
            "dataplane": "vrouter",
            "ipConfig": {
              "ipv4": {
                "address": "10.1.51.2/30",
                "gateway": "10.1.51.1",
                "routes": [
                  "10.1.51.0/30"
                ]
              },
              "ipv6": {
                "address": "2001:db8::10:1:51:2/126",
                "gateway": "2001:db8::10:1:51:1",
                "routes": [
                  "2001:db8::1:1:51:0/126"
                ]
              }
            }
          }
        }
      ]

```



```

    },
    "name": "net2",
    "interface": "net2",
    "cni-args": {
      "mac": "aa:bb:cc:dd:ee:52",
      "dataplane": "vrouter",
      "ipConfig": {
        "ipv4": {
          "address": "10.1.52.2/30",
          "gateway": "10.1.52.1",
          "routes": [
            "10.1.52.0/30"
          ]
        },
        "ipv6": {
          "address": "2001:db8::10:1:52:2/126",
          "gateway": "2001:db8::10:1:52:1",
          "routes": [
            "2001:db8::10:1:52:0/126"
          ]
        }
      }
    }
  }
]
spec:
  containers:
    - name: L3-pktgen-odu1
      image: svl-artifactory.juniper.net/blr-data-plane/dpdk-app/dpdk:21.11
      imagePullPolicy: IfNotPresent
      command: ["/bin/bash", "-c", "sleep infinity"]
      securityContext:
        privileged: false
      env:
        - name: KUBERNETES_POD_UID
          valueFrom:
            fieldRef:
              fieldPath: metadata.uid
      resources:
        requests:
          memory: 4Gi
        limits:
          hugepages-1Gi: 4Gi
      name: hugepages

```

```
command: ["sleep"]
args: ["infinity"]
volumeMounts:
  - name: dpdk
    mountPath: /dpdk
    subPathExpr: $(KUBERNETES_POD_UID)
  - name: hugepages
    mountPath: /hugepages
volumes:
  - name: dpdk
    hostPath:
      path: /var/run/jcnr/containers
  - name: hugepages
    emptyDir:
      medium: HugePages
```