JUNIPER NETWORKS | Engineering Simplicity

Quick Start

Junos Containerized Routing Protocol Daemon (cRPD)

IN THIS GUIDE

Step 1: Begin

IN THIS SECTION

In this guide, we walk you through how to install and configure Junos® containerized routing protocol process (cRPD) on a Linux host and access it using Junos CLI. Next, we show you how to connect and configure two Junos cRPD instances and establish an OSPF adjacency.

Meet Junos cRPD

Junos cRPD is a cloud-native, containerized routing engine that supports simple deployment throughout the cloud infrastructure. Junos cRPD decouples the RPD from Junos OS and packages the RPD as a Docker container that runs on any Linux-based system, including servers and whitebox routers. Docker is an open source software platform that makes it simple to create and manage a virtual container.

Junos cRPD supports multiple protocols such as OSPF, IS-IS, BGP, MP-BGP, and so on. Junos cRPD shares the same management functionality as Junos OS and Junos OS Evolved to deliver a consistent configuration and management experience in routers, servers, or any Linux-based device.

Get Ready

Before you begin deployment:

- Familiarize yourself with your Junos cRPD license agreement. See Flex Software License for cRPD and Managing cRPD Licenses.
- Set up a Docker hub account. You'll need an account to download Docker Engine. See Docker ID accounts for details.

Install and Configure Docker on a Linux Host

1. Verify that your host meets these minimum system requirements.
   - Linux OS support - Ubuntu 18.04
   - Linux Kernel - 4.15
   - Docker Engine- 18.09.1 or later versions
   - CPUs- 2 CPU core
   - Memory - 4 GB
   - Disk space - 10 GB
   - Host processor type - x86_64 multicore CPU

- Network Interface - Ethernet

```
root@crpd:~# uname -a
Linux crpd 5.4.0-193-generic #213-Ubuntu SMP Fri Aug 2 19:14:16 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
```

```
root@crpd:~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.5 LTS
Release:        20.04
Codename:       focal
```

2. Download the Docker software.

   a. Download the necessary tools.

```
root@crpd:~# apt install apt-transport-https ca-certificates curl software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20230311ubuntu0.20.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.68.0-1ubuntu2.23).
curl set to manually installed.
The following additional packages will be installed:
  python3-software-properties
The following NEW packages will be installed:
  apt-transport-https
The following packages will be upgraded:
  python3-software-properties software-properties-common
2 upgraded, 1 newly installed, 0 to remove and 58 not upgraded.
Need to get 33.8 kB of archives.
After this operation, 143 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 apt-transport-https all 2.0.10
[1,704 B]
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 software-properties-common all
0.99.9.12 [10.4 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-software-properties all
0.99.9.12 [21.7 kB]
Fetched 33.8 kB in 0s (73.4 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 72196 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.0.10_all.deb ...
```

```
Unpacking apt-transport-https (2.0.10) ...
Preparing to unpack .../software-properties-common_0.99.9.12_all.deb ...
Unpacking software-properties-common (0.99.9.12) over (0.99.9.8) ...
Preparing to unpack .../python3-software-properties_0.99.9.12_all.deb ...
Unpacking python3-software-properties (0.99.9.12) over (0.99.9.8) ...
Setting up apt-transport-https (2.0.10) ...
Setting up python3-software-properties (0.99.9.12) ...
Setting up software-properties-common (0.99.9.12) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for dbus (1.12.16-2ubuntu2.3) ...
```

b. Add the Docker repository to Advanced Packaging Tool (APT) sources.

```
root@crpd:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
OK
root@crpd:~# lsb_release -cs
focal
root@crpd:~# add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -
cs) stable"
Get:1 https://download.docker.com/linux/ubuntu focal InRelease [57.7 kB]
Hit:2 http://us.archive.ubuntu.com/ubuntu focal InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:5 http://us.archive.ubuntu.com/ubuntu focal-security InRelease
Get:6 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages [49.7 kB]
Fetched 107 kB in 1s (84.9 kB/s)
Reading package lists... Done
```

c. Update the list of available packages.

```
root@crpd:~# apt update
Hit:1 https://download.docker.com/linux/ubuntu focal InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu focal InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:5 http://us.archive.ubuntu.com/ubuntu focal-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

d. Install the latest version of Docker Engine.

```
root@crpd:~# apt install docker-ce
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin pigz
slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-
plugin pigz slirp4netns
0 upgraded, 8 newly installed, 0 to remove and 58 not upgraded.
Need to get 121 MB of archives.
After this operation, 437 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 pigz amd64 2.4-1 [57.4 kB]
Get:2 https://download.docker.com/linux/ubuntu focal/stable amd64 containerd.io amd64 1.7.21-1 [29.5 MB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 slirp4netns amd64 0.4.3-1 [74.3 kB]
Get:4 https://download.docker.com/linux/ubuntu focal/stable amd64 docker-buildx-plugin amd64
0.16.2-1~ubuntu.20.04~focal [29.9 MB]
Get:5 https://download.docker.com/linux/ubuntu focal/stable amd64 docker-ce-cli amd64
5:27.2.0-1~ubuntu.20.04~focal [14.8 MB]
Get:6 https://download.docker.com/linux/ubuntu focal/stable amd64 docker-ce amd64
5:27.2.0-1~ubuntu.20.04~focal [25.2 MB]
Get:7 https://download.docker.com/linux/ubuntu focal/stable amd64 docker-ce-rootless-extras amd64
5:27.2.0-1~ubuntu.20.04~focal [9,328 kB]
Get:8 https://download.docker.com/linux/ubuntu focal/stable amd64 docker-compose-plugin amd64
2.29.2-1~ubuntu.20.04~focal [12.5 MB]
Fetched 121 MB in 2s (54.7 MB/s)
Selecting previously unselected package pigz.
(Reading database ... 72197 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.4-1_amd64.deb ...
Unpacking pigz (2.4-1) ...
Selecting previously unselected package containerd.io.
Preparing to unpack .../1-containerd.io_1.7.21-1_amd64.deb ...
Unpacking containerd.io (1.7.21-1) ...
Selecting previously unselected package docker-buildx-plugin.
Preparing to unpack .../2-docker-buildx-plugin_0.16.2-1~ubuntu.20.04~focal_amd64.deb ...
Unpacking docker-buildx-plugin (0.16.2-1~ubuntu.20.04~focal) ...
Selecting previously unselected package docker-ce-cli.
Preparing to unpack .../3-docker-ce-cli_5%3a27.2.0-1~ubuntu.20.04~focal_amd64.deb ...
Unpacking docker-ce-cli (5:27.2.0-1~ubuntu.20.04~focal) ...
```

```
.....
```

e. Check to see if the installation is successful.

```
root@crpd:~# docker version
Client: Docker Engine - Community
 Version:           20.10.7
 API version:       1.41
 Go version:        go1.13.15
 Git commit:        f0df350
 Built:             Wed Jun  2 11:56:40 2021
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.7
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.13.15
  Git commit:       b0f5bc3
  Built:            Wed Jun  2 11:54:48 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.4.6
  GitCommit:        d71fcd7d8303cbf684402823e425e9dd2e99285d
 runc:
  Version:          1.0.0-rc95
  GitCommit:        b9ee9c6314599f1b4a7f497e1f1f856fe433d3b7
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

TIP: Use these commands to install the components you need for the Python environment and packages:

- `apt-add-repository universe`
- `apt-get update`
- `apt-get install python-pip`

- `python -m pip install grpcio`
- `python -m pip install grpcio-tools`

Download and Install Junos cRPD Software

Now that you've installed Docker on the Linux host and confirmed that the Docker Engine is running, let's download the Junos cRPD software from the Juniper Networks software download page.

NOTE: To download, install, and start using Junos cRPD without a license key, see Start your free trial today.

NOTE: You can open an Admin Case with Customer Care for privileges to download the software.

1. Navigate to the Juniper Networks Support page for Junos cRPD: https://support.juniper.net/support/downloads/?p=crpd and click the latest version.
2. Enter your user ID and password and accept the Juniper end-user license agreement. You'll be guided to the software image download page.
3. Download the image directly on your host. Copy and paste the generated string as instructed on the screen.

```
root@crpd:~# wget "https://cdn.juniper.net/software/crpd/24.2R1/junos-routing-crpd-docker-
amd64-24.2R1.14.tgz?SM_USER=user1&__gda__=xxxxx" -O junos-routing-crpd-docker-amd64-24.2R1.14.tgz

--2024-08-29 18:12:48--  https://cdn.juniper.net/software/crpd/24.2R1/junos-routing-crpd-docker-
amd64-24.2R1.14.tgz?SM_USER=xxxxx&__gda__=xxxxx
Resolving cdn.juniper.net (cdn.juniper.net)... 104.92.231.114
Connecting to cdn.juniper.net (cdn.juniper.net)|104.92.231.114|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 167932289 (160M) [application/octet-stream]
Saving to: ‘junos-routing-crpd-docker-amd64-24.2R1.14.tgz’

junos-routing-crpd-docker-amd64-24.2R1. 100%
[=======================================================================>] 160.15M  67.2MB/s    in 2.4s

2024-08-29 18:12:51 (67.2 MB/s) - ‘junos-routing-crpd-docker-amd64-24.2R1.14.tgz’ saved [167932289/167932289]
```

4. Load the Junos cRPD software image to Docker.

```
root@crpd:~#  docker load -i junos-routing-crpd-docker-amd64-24.2R1.14.tgz
8716c4e476eb: Loading layer [==================================================>]  537.7MB/537.7MB
Loaded image: crpd:24.2R1.14
```

```
root@crpd:~# docker images
REPOSITORY   TAG         IMAGE ID       CREATED        SIZE
crpd         24.2R1.14   281363ff6dbf   2 months ago   530MB
```

5. Create a data volume for configuration and var logs.

```
root@crpd:~# docker volume create crpd01-config
crpd01-config
```

```
root@crpd:~# docker volume create crpd01-varlog
crpd01-varlog
```

6. Create a Junos cRPD instance. In this example, you'll name it crpd01.

```
root@crpd:~# docker run --rm --detach --name crpd01 -h crpd01 --net=none --privileged -v crpd01-config:/
config -v crpd01-varlog:/var/log -it crpd:24.2R1.14
e39177e2a41b5fc2147115092d10e12a27c77976c88387a694faa5cbc5857f1e
```

Check cRPD Resource Requirements for the details.

7. Verify the newly created container details.

```
root@crpd:~# docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED         STATUS         PORTS     NAMES
710b3b6ca824   crpd:24.2R1.14   "/sbin/runit-init.sh"   4 seconds ago   Up 4 seconds             crpd01
```

Step 2: Up and Running

IN THIS SECTION

Access the CLI

You configure Junos cRPD using Junos CLI commands for routing services. Here's how to access the Junos CLI:

1. Log in to the Junos cRPD container.

```
root@crpd:~# docker exec -it crpd01 cli
```

2. Check the Junos OS version.

```
root@crpd01> show version
Hostname: crpd01
Model: cRPD
Junos: 24.2R1.14
cRPD package version : 24.2R1.14 built by builder on 2024-06-22 01:04:40 UTC
```

3. Enter configuration mode.

```
root@crpd01> configure
Entering configuration mode
```

4. Add a password to the root administration user account. Enter a plain text password.

```
[edit]
root@crpd01# set system root-authentication plain-text-password
```

```
New password:
Retype new password:
```

5. Commit the configuration.

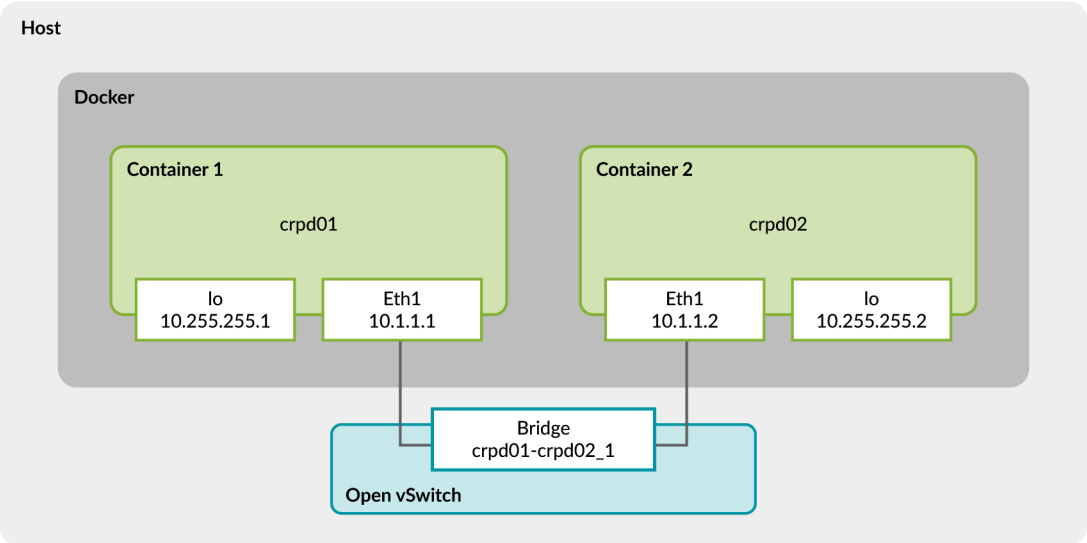
```
[edit]
root@crpd01# commit
commit complete
root@crpd01# exit
Exiting configuration mode

root@crpd01> exit
root@crpd:~#
```

6. Continue customizing the configuration, if required.

Interconnect cRPD Instances

Now let's learn how to build point-to-point links between two Junos cRPD containers.

In this example, we use two containers, crpd01 and crpd02, and connect them using eth1 interfaces that are connected to an OpenVswitch (OVS) bridge on the host. We're using an OVS bridge for Docker networking because it supports multiple host networking and provides secure communication. Refer to the following illustration:



1. Install the OVS switch utility.

```
root@crpd:~# apt install openvswitch-switch
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libunbound8 openvswitch-common python3-openvswitch python3-sortedcontainers
Suggested packages:
  openvswitch-doc python-sortedcontainers-doc
The following NEW packages will be installed:
  libunbound8 openvswitch-common openvswitch-switch python3-openvswitch python3-sortedcontainers
0 upgraded, 5 newly installed, 0 to remove and 58 not upgraded.
Need to get 3,167 kB of archives.
After this operation, 14.2 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libunbound8 amd64 1.9.4-2ubuntu1.6 [349
kB]
Get:2 http://us.archive.ubuntu.com/ubuntu focal/main amd64 python3-sortedcontainers all 2.1.0-2 [27.3 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 openvswitch-common amd64
2.13.8-0ubuntu1.4 [1,156 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-openvswitch all
2.13.8-0ubuntu1.4 [94.9 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 openvswitch-switch amd64
2.13.8-0ubuntu1.4 [1,539 kB]
Fetched 3,167 kB in 1s (3,989 kB/s)
Selecting previously unselected package libunbound8:amd64.
(Reading database ... 72451 files and directories currently installed.)
Preparing to unpack .../libunbound8_1.9.4-2ubuntu1.6_amd64.deb ...
```

2. Create another Junos cRPD container called crpd02.

```
root@crpd:~# docker volume create crpd02-config
crpd02-config
root@crpd:~# docker volume create crpd02-varlog
crpd02-varlog
root@crpd:~# docker run --rm --detach --name crpd02 -h crpd02 --net=none --privileged -v crpd02-config:/
config -v crpd02-varlog:/var/log -it crpd:24.2R1.14
c000db8998b708b6ff04581d53404d3164a1137c99c9adbe4c34ce3ea3f74671
```

3. Verify the newly created container details.

```
root@crpd:~# docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED          STATUS          PORTS     NAMES
c000db8998b7   crpd:24.2R1.14   "/sbin/runit-init.sh"   20 seconds ago   Up 20 seconds             crpd02
710b3b6ca824   crpd:24.2R1.14   "/sbin/runit-init.sh"   6 minutes ago    Up 6 minutes              crpd01
```

4. Log in to the Junos cRPD container.

```
root@crpd:~# docker exec -it crpd02 cli
root@crpd02> configure
Entering configuration mode
```

5. Add a password to the root administration user account. Enter a plain text password.

```
[edit]
root@crpd02# set system root-authentication plain-text-password
New password:
Retype new password:
```

6. Commit the configuration.

```
[edit]
root@crpd02# commit
commit complete
root@crpd02# exit
Exiting configuration mode

root@crpd01> exit
root@crpd:~#
```

7. Configure a new bridge named crpd01-crpd02_1 and add eth1 interface to crpd01 and crpd02 containers.

```
root@crpd:~# ovs-vsctl add-br crpd01-crpd02_1
root@crpd:~# ovs-docker add-port crpd01-crpd02_1 eth1 crpd01
root@crpd:~# ovs-docker add-port crpd01-crpd02_1 eth1 crpd02
```

8. Log in to the crpd01 container and verify the interface configuration.

```
root@crpd:~# docker exec -it crpd01 bash
===>
           Containerized Routing Protocols Daemon (CRPD)
 Copyright (C) 2020-2023, Juniper Networks, Inc. All rights reserved.
                                                                    <===
root@crpd01:/# ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::300a:aeff:fe25:2eb2  prefixlen 64  scopeid 0x20<link>
        ether 32:0a:ae:25:2e:b2  txqueuelen 1000  (Ethernet)
        RX packets 15  bytes 1402 (1.4 KB)
        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
root@crpd01:/# exit
exit
```

9. Log in to the crpd02 container and verify the interface configuration.

```
root@crpd:~# docker exec -it crpd02 bash
===>
           Containerized Routing Protocols Daemon (CRPD)
 Copyright (C) 2020-2023, Juniper Networks, Inc. All rights reserved.
                                                                    <===
root@crpd02:/# ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::48d4:b1ff:fe30:9694  prefixlen 64  scopeid 0x20<link>
        ether 4a:d4:b1:30:96:94  txqueuelen 1000  (Ethernet)
        RX packets 8  bytes 656 (656.0 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

root@crpd02:/# exit
exit
root@crpd:~#
```

10. Configure IP addresses to the eth1 interfaces.

```
root@crpd:~# docker exec -d crpd01 ifconfig eth1 10.1.1.1/24
root@crpd:~# docker exec -d crpd02 ifconfig eth1 10.1.1.2/24
```

11. Log in to the Junos cRPD container crpd01.

```
root@crpd:~# docker exec -it crpd01 bash
===>
           Containerized Routing Protocols Daemon (CRPD)
 Copyright (C) 2020-2023, Juniper Networks, Inc. All rights reserved.
                                                                    <===
root@crpd01:/# ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.1.1  netmask 255.255.255.0  broadcast 10.1.1.255
        inet6 fe80::300a:aeff:fe25:2eb2  prefixlen 64  scopeid 0x20<link>
        ether 32:0a:ae:25:2e:b2  txqueuelen 1000  (Ethernet)
        RX packets 17  bytes 1542 (1.5 KB)
        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
```

12. Send a ping to the crpd02 container to confirm connectivity between the two containers. Use the IP address of eth1 of crpd02 (10.1.1.2) to ping the container.

```
root@crpd01:/# ping 10.1.1.2 -c 2
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.
64 bytes from 10.1.1.2: icmp_seq=1 ttl=64 time=0.323 ms
64 bytes from 10.1.1.2: icmp_seq=2 ttl=64 time=0.042 ms

--- 10.1.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1018ms
rtt min/avg/max/mdev = 0.042/0.182/0.323/0.141 ms
root@crpd01:/# exit
exit
```

The output confirms that the two containers can communicate with each other.

13. Log in to the Junos cRPD container crpd02.

```
root@crpd:~# docker exec -it crpd02 bash
```

```
===>
           Containerized Routing Protocols Daemon (CRPD)
 Copyright (C) 2020-2023, Juniper Networks, Inc. All rights reserved.
                                                                    <===
root@crpd02:/# ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.1.2  netmask 255.255.255.0  broadcast 10.1.1.255
        inet6 fe80::48d4:b1ff:fe30:9694  prefixlen 64  scopeid 0x20<link>
        ether 4a:d4:b1:30:96:94  txqueuelen 1000  (Ethernet)
        RX packets 16  bytes 1244 (1.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 1124 (1.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

14. Send a ping to the crpd01 container to confirm connectivity between the two containers. Use the IP address of eth1 of crpd01 (10.1.1.1) to ping the container.

```
root@crpd02:/# ping 10.1.1.1
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_seq=1 ttl=64 time=0.509 ms
64 bytes from 10.1.1.1: icmp_seq=2 ttl=64 time=0.074 ms
64 bytes from 10.1.1.1: icmp_seq=3 ttl=64 time=0.074 ms
^C
--- 10.1.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2048ms
rtt min/avg/max/mdev = 0.074/0.219/0.509/0.205 ms
```

The output confirms that the two containers can communicate with each other.

Configure Open Shortest Path First (OSPF)

Now you have two containers, crpd01 and crpd02, that are connected and communicating. The next step is to establish neighbor adjacencies for the two containers. OSPF-enabled routers must form adjacencies with their neighbor before they can share information with that neighbor.

1. Configure OSPF on the crpd01 container.

```
[edit]
root@crpd01# show policy-options
policy-statement adv {
    term 1 {
```

```
        from {
            route-filter 10.10.10.0/24 exact;
        }
        then accept;
    }
}

[edit]
root@crpd01# show protocols
ospf {
    area 0.0.0.0 {
        interface eth1;
        interface lo0.0;
    }
    export adv;
}

[edit]
root@crpd01# show routing-options
router-id 10.255.255.1;
static {
    route 10.10.10.0/24 reject;
}
```

2. Commit the configuration.

```
[edit]
root@crpd01# commit
commit complete
```

3. Repeat steps 1 and 2 to configure OSPF on the crpd02 container.

```
root@crpd02# show policy-options
policy-statement adv {
    term 1 {
        from {
            route-filter 10.20.20.0/24 exact;
        }
        then accept;
    }
}

[edit]
root@crpd02# show routing-options
router-id 10.255.255.2;
```

```
static {
    route 10.20.20.0/24 reject;
}

[edit]
root@crpd02# show protocols ospf
area 0.0.0.0 {
    interface eth1;
    interface lo0.0;
}
export adv;
```

4. Use show commands to verify OSPF neighbors that have an immediate adjacency.

```
root@crpd01> show ospf neighbor

Address          Interface              State           ID               Pri  Dead
10.1.1.2         eth1                   Full            10.255.255.2     128    38
```

```
root@crpd01> show ospf route
Topology default Route Table:

Prefix             Path  Route      NH       Metric NextHop       Nexthop
                   Type  Type       Type            Interface     Address/LSP
10.255.255.2       Intra AS BR      IP            1 eth1          10.1.1.2
10.1.1.0/24        Intra Network    IP            1 eth1
10.20.20.0/24      Ext2  Network    IP            0 eth1          10.1.1.2
10.255.255.1/32    Intra Network    IP            0 lo0.0
10.255.255.2/32    Intra Network    IP            1 eth1          10.1.1.2
```

The output shows the container's own loopback address and the loopback addresses of any containers which it is immediately adjacent to. The output confirms that the Junos cRPD has established an OSPF neighbor relationship and has learned their addresses and interfaces.

View Junos cRPD Core Files

When a core file is generated, you can find the output in the `/var/crash` folder. The generated core files are stored on the system that is hosting the Docker containers.

1. Change to the directory where crash files are stored.

```
root@linux-host:~# cd /var/crash
```

2. List the crash files.

```
root@linux-host:/var/crash#
 ls -l

total 32
-rw-r----- 1 root root 29304 Jul 14 15:14 _usr_bin_unattended-upgrade.0.crash
```

3. Identify the location of the core files.

```
root@linux-host:/var/crash# sysctl kernel.core_pattern
kernel.core_pattern = |/bin/bash -c "$@" -- eval /bin/gzip > /var/crash/%h.%e.core.%t-%p-%u.gz
```

Step 3: Keep Going

IN THIS SECTION

Congratulations! You've now completed the initial configuration for Junos cRPD!

What's Next?

Now that you've configured Junos cRPD containers and established communication between two containers, here are some things you might want to configure next.

If you want to	Then
Download, activate, and manage your software licenses to unlock additional features for your Junos cRPD	See Flex Software License for cRPD and Managing cRPD Licenses
Find more in-depth information about installing and configuring Junos cRPD	See Day One: Cloud Native Routing with cRPD
Check out blog posts about Junos cRPD with Docker Desktop.	See Juniper cRPD 20.4 on Docker Desktop
Configure routing and network protocols	See Routing and Network Protocols
Learn about Juniper Networks cloud-native routing solution	Watch the video Cloud-Native Routing Overview

General Information

Here are some excellent resources that will help you take your Junos cRPD knowledge to the next level:

If you want to	Then
Find in-depth product documentation for Junos cRPD	See cRPD Documentation
Explore all documentation available for Junos OS	Visit Junos OS Documentation
Stay up to date on new and changed features and known See the Junos OS Release Notes and resolved issues	Check out Junos OS Release Notes