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 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-user@linux-host:~# uname -a
Linux ix-crpd-03 4.15.0-147-generic #151-Ubuntu SMP Fri Jun 18 19:21:19 UTC 2021 x86_64 x86_64 x86_64 GNU/
Linux
```

```
root-user@linux-host:lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.1 LTS
Release:        18.04
Codename:       bionic
```

2. Download the Docker software.

   a. Update your existing list of packages and download the necessary tools.

```
rootuser@linux-host:~# apt install apt-transport-https ca-certificates curl software-properties-common
[sudo] password for lab:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'apt' instead of 'apt-transport-https'
The following additional packages will be installed:………………………………….
```

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

```
rootuser@linux-host:~# add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
bionic stable"
Get:1 https://download.docker.com/linux/ubuntu bionic InRelease [64.4 kB]
Get:2 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages [18.8 kB]
Hit:3 http://archive.ubuntu.com/ubuntu bionic InRelease
Get:4 http://archive.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/main Translation-en [516 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic-security/main Translation-en [329 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic-updates/main Translation-en [422 kB]
Fetched 1,528 kB in 8s (185 kB/s)
Reading package lists... Done
```

c. Update the database with the Docker packages.

```
rootuser@linux-host:~# apt update
Hit:1 https://download.docker.com/linux/ubuntu bionic InRelease
Hit:2 http://archive.ubuntu.com/ubuntu bionic InRelease
Hit:3 http://archive.ubuntu.com/ubuntu bionic-security InRelease
Hit:4 http://archive.ubuntu.com/ubuntu bionic-updates InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

d. Update the apt package index, and install the latest version of Docker Engine.

```
rootuser@linux-host:~# 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-ce-cli docker-ce-rootless-extras docker-scan-plugin
  libltdl7 libseccomp2
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
Recommended packages:
  pigz slirp4netns
………………………………………………………………….
```

e. Check to see if the installation is successful.

```
rootuser@linux-host:~# 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.

```
rootuser@linux-host:~#  wget -O junos-routing-crpd-docker-21.2R1.10.tgz https://cdn.juniper.net/software/
crpd/21.2R1.10/junos-routing-crpd-docker-21.2R1.10.tgz?
SM_USER=user1&__gda__=1626246704_4cd5cfea47ebec7c1226d07e671d0186
Resolving cdn.juniper.net (cdn.juniper.net)... 23.203.176.210
Connecting to cdn.juniper.net (cdn.juniper.net)|23.203.176.210|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 127066581 (121M) [application/octet-stream]
Saving to: âjunos-routing-crpd-docker-21.2R1.10.tgzâ

junos-routing-crpd-docker-21.2R1.10.tgz   100%
[=====================================================================================>] 121.18M  4.08MB/
s    in 34s

2021-07-14 07:02:44 (3.57 MB/s) - âjunos-routing-crpd-docker-21.2R1.10.tgzâ saved [127066581/127066581]
```

4. Load the Junos cRPD software image to Docker.

```
rootuser@linux-host:~# docker load -i junos-routing-crpd-docker-21.2R1.10.tgz

6effd95c47f2: Loading layer [==================================================>]  65.61MB/65.61MB
.....................................................................................................
Loaded image: crpd:21.2R1.10
```

```
rootuser@linux-host:~# docker images
REPOSITORY   TAG         IMAGE ID       CREATED       SIZE
crpd         21.2R1.10   f9b634369718   3 weeks ago   374MB
```

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

```
rootuser@linux-host:~# docker volume create crpd01-config
crpd01-config
```

```
rootuser@linux-host:~# docker volume create crpd01-varlog
crpd01-varlog
```

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

```
rootuser@linux-host:~# docker run --rm --detach --name crpd01 -h crpd01 --net=bridge --privileged -v crpd01-
config:/config -v crpd01-varlog:/var/log -it crpd:21.2R1.10
e39177e2a41b5fc2147115092d10e12a27c77976c88387a694faa5cbc5857f1e
```

   Alternatively, you can allocate the amount of memory to the Junos cRPD instance while creating the instance.

```
rootuser@linux-host:~# docker run --rm --detach --name crpd-01 -h crpd-01 --privileged -v crpd01-config:/
config -v crpd01-varlog:/var/log -m 2048MB --memory-swap=2048MB -it crpd:21.2R1.10
WARNING: Your kernel does not support swap limit capabilities or the cgroup is not mounted. Memory limited
without swap.
1125e62c9c639fc6fca87121d8c1a014713495b5e763f4a34972f5a28999b56c
```

   Check cRPD Resource Requirements for the details.

7. Verify the newly created container details.

```
rootuser@linux-host:~# docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED              STATUS
PORTS                                                                          NAMES
e39177e2a41b   crpd:21.2R1.10   "/sbin/runit-init.sh"   About a minute ago   Up About a minute   22/tcp, 179/
tcp, 830/tcp, 3784/tcp, 4784/tcp, 6784/tcp, 7784/tcp, 50051/tcp   crpd01
```

```
rootuser@linux-host:~# docker stats
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O          BLOCK I/O      PIDS
e39177e2a41b   crpd01    0.00%     147.1MiB / 3.853GiB   3.73%     1.24kB / 826B    4.1kB / 35MB   58
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O          BLOCK I/O      PIDS
e39177e2a41b   crpd01    0.00%     147.1MiB / 3.853GiB   3.73%     1.24kB / 826B    4.1kB / 35MB   58
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O          BLOCK I/O      PIDS
e39177e2a41b   crpd01    0.05%     147.1MiB / 3.853GiB   3.73%     1.24kB / 826B    4.1kB / 35MB   58
```

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.

```
rootuser@linux-host:~# docker exec -it crpd01 cli
```

2. Check the Junos OS version.

```
rootuser@crpd01> show version
root@crpd01> show version
Hostname: crpd01
Model: cRPD
Junos: 21.2R1.10
cRPD package version : 21.2R1.10 built by builder on 2021-06-21 14:13:43 UTC
```

3. Enter configuration mode.

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

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

```
[edit]
rootuser@crpd01# set system root-authentication plain-text-password
New password:
Retype new password:
```

5. Commit the configuration.

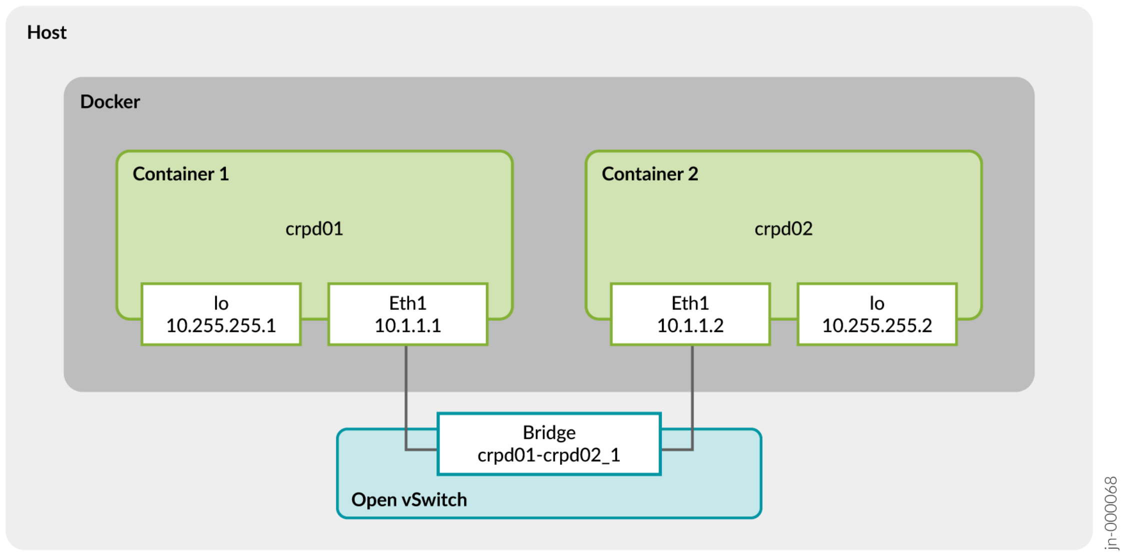
```
[edit]
rootuser@crpd01# commit
commit complete
```

6. Log in to the Junos cRPD instance with the CLI and continue customizing the configuration.

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.

```
rootuser@linux-host:~# apt-get install openvswitch-switch
sudo] password for lab:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpython-stdlib libpython2.7-minimal libpython2.7-stdlib openvswitch-common python python-minimal python-
six python2.7 python2.7-minimal
```

2. Navigate to the usr/bin directory path and use the `wget` command to download and to install the OVS docker.

```
rootuser@linux-host:~# cd /usr/bin
```

```
rootuser@linux-host:~# wget "https://raw.githubusercontent.com/openvswitch/ovs/master/utilities/ovs-docker"

--2021-07-14 07:55:17--  https://raw.githubusercontent.com/openvswitch/ovs/master/utilities/ovs-docker
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.111.133,
185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8064 (7.9K) [text/plain]
Saving to: âovs-docker.1â

ovs-docker.1                              100%
[=======================================================================================>]   7.88K  --.-KB/
s    in 0s

2021-07-14 07:55:17 (115 MB/s) - âovs-docker.1â saved [8064/8064]
```

3. Change the permissions on the OVS bridge.

```
rootuser@linux-host:/usr/bin chmod a+rwx ovs-docker
```

4. Create another Junos cRPD container called crpd02.

```
rootuser@linux-host:~# docker run --rm --detach --name crpd02 -h crpd02 --net=bridge --privileged -v crpd02-
config:/config -v crpd02-varlog:/var/log -it crpd:21.2R1.10

e18aec5bfcb8567ab09b3db3ed5794271edefe553a4c27a3d124975b116aa02
```

5. Create a bridge called my-net. This step creates eth1 interfaces on crpd01 and crdp02.

```
rootuser@linux-host:~# docker network create --internal my-net

37ddf7fd93a724100df023d23e98a86a4eb4ba2cbf3eda0cd811744936a84116
```

6. Create an OVS bridge and add crpd01 and crpd02 containers with eth1 interfaces.

```
rootuser@linux-host:~# ovs-vsctl add-br crpd01-crpd02_1
rootuser@linux-host:~# ovs-docker add-port crpd01-crpd02_1 eth1 crpd01
rootuser@linux-host:~# ovs-docker add-port crpd01-crpd02_1 eth1 crpd02
```

7. Add IP addresses to the eth1 interfaces and to the loopback interfaces.

```
rootuser@linux-host:~# docker exec -d crpd01 ifconfig eth1 10.1.1.1/24
rootuser@linux-host:~# docker exec -d crpd02 ifconfig eth1 10.1.1.2/24
rootuser@linux-host:~# docker exec -d crpd01 ifconfig lo0 10.255.255.1 netmask 255.255.255.255
rootuser@linux-host:~# docker exec -d crpd02 ifconfig lo0 10.255.255.2 netmask 255.255.255.255
```

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

```
rootuser@linux-host:~# docker exec -it crpd01 bash

rootuser@crpd01:/# ifconfig
.....
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::42:acff:fe12:2  prefixlen 64  scopeid 0x20<link>
        ether 02:42:ac:12:00:02  txqueuelen 0  (Ethernet)
        RX packets 24  bytes 2128 (2.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 788 (788.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
........
```

9. 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.

```
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
```

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]
rootuser@crpd01# show policy-options
policy-statement adv {
    term 1 {
        from {
            route-filter 10.10.10.0/24 exact;
        }
        then accept;
    }
}

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

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

2. Commit the configuration.

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

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

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

[edit]
rootuser@crpd02# show routing-options
router-id 10.255.255.2;
static {
    route 10.20.20.0/24 reject;
}

[edit]
rootuser@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.

```
rootuser@crpd01> show ospf neighbor

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

```
rootuser@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.

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

2. List the crash files.

```
rootuser@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.

```
rootuser@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