Juniper Networks | Engineering Simplicity

vSRX Deployment Guide for KVM

Published
2021-01-04

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

vSRX Deployment Guide for KVM

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

About This Guide | vi

1 Overview

Unresolved topicref | 2

Understand vSRX with KVM | 2

Requirements for vSRX on KVM | 7

Unresolved topicref | 18

2 Installing vSRX in KVM

Prepare Your Server for vSRX Installation | 20

- Enable Nested Virtualization | 20
- Upgrade the Linux Kernel on Ubuntu | 21

Install vSRX with KVM | 22

- Install vSRX with virt-manager | 22
- Install vSRX with virt-install | 25

Example: Install and Launch vSRX on Ubuntu | 28

- Requirements | 28
- Overview | 29
- Quick Configuration - Install and Launch a vSRX VM on Ubuntu | 29 | 33
- Step by Step Configuration | 33

Load an Initial Configuration on a vSRX with KVM | 45

- Create a vSRX Bootstrap ISO Image | 46
- Provision vSRX with an ISO Bootstrap Image on KVM | 47

Unresolved topicref | 48

3 vSRX VM Management

Connect to the vSRX Management Console on KVM | 51

Add a Virtual Network to a vSRX VM with KVM | 51

Add a Virtio Virtual Interface to a vSRX VM with KVM | 53

Configure SR-IOV and PCI on KVM | 55

SR-IOV Overview | **56**

SR-IOV HA Support with Trust Mode Disabled (KVM only) | **56**

Understand SR-IOV HA Support with Trust Mode Disabled (KVM only) | **57**

Configure SR-IOV support with Trust Mode Disabled (KVM only) | **58**

Limitations | **58**

Configure an SR-IOV Interface on KVM | **59**

Upgrade a Multi-core vSRX | 63

Configure the Queue Value for vSRX VM with KVM | **64**

Shutdown the vSRX Instance with virt-manager | **64**

Upgrade vSRX with virt-manager | **65**

Monitor the vSRX VM in KVM | 66

Manage the vSRX Instance on KVM | 67

Power On the vSRX Instance with virt-manager | **68**

Power On the vSRX Instance with virsh | **68**

Pause the vSRX Instance with virt-manager | **69**

Pause the vSRX Instance with virsh | **69**

Rebooting the vSRX Instance with virt-manager | **69**

Reboot the vSRX Instance with virsh | **69**

Power Off the vSRX Instance with virt-manager | **70**

Power Off the vSRX Instance with virsh | **70**

Shutdown the vSRX Instance with virt-manager | **71**

Shutdown the vSRX Instance with virsh | **71**

Remove the vSRX Instance with virsh | 72

Recover the Root Password for vSRX in a KVM Environment | 72

4 Configuring and Managing vSRX

Unresolved topicref | 76

Unresolved topicref | 76

Unresolved topicref | 76

Unresolved topicref | 77

Unresolved topicref | 77

Unresolved topicref | 77

5 Configuring vSRX Chassis Clusters

Configure a vSRX Chassis Cluster in Junos OS | 79

Chassis Cluster Overview | 79

Enable Chassis Cluster Formation | 80

Chassis Cluster Quick Setup with J-Web | 81

Manually Configure a Chassis Cluster with J-Web | 83

vSRX Cluster Staging and Provisioning for KVM | 90

Chassis Cluster Provisioning on vSRX | 90

Creating the Chassis Cluster Virtual Networks with virt-manager | 92

Creating the Chassis Cluster Virtual Networks with virsh | 92

Configuring the Control and Fabric Interfaces with virt-manager | 94

Configuring the Control and Fabric Interfaces with virsh | 94

Configuring Chassis Cluster Fabric Ports | 95

Verify the Chassis Cluster Configuration | 96

6 Troubleshooting

Unresolved topicref | 99

About This Guide

Use this guide to install the vSRX Virtual Firewall in a kernel-based virtual machine (KVM) environment. This guide also includes basic vSRX configuration and management procedures.

After completing the installation and basic configuration procedures covered in this guide, refer to the Junos OS documentation for information about further software configuration.

1

CHAPTER

Overview

Unresolved topicref | 2

Understand vSRX with KVM | 2

Requirements for vSRX on KVM | 7

Unresolved topicref | 18

Unresolved topicref

SUMMARY

Unresolved topicref placeholder.

This is a placeholder for unresolved topicref links.

Understand vSRX with KVM

IN THIS SECTION

- vSRX on KVM | 2
- vSRX Scale Up Performance | 3

This section presents an overview of vSRX on KVM.

vSRX on KVM

The Linux kernel uses the kernel-based virtual machine (*KVM*) as a virtualization infrastructure. KVM is open source software that you can use to create multiple virtual machines (VMs) and to install security and networking appliances.

The basic components of KVM include:

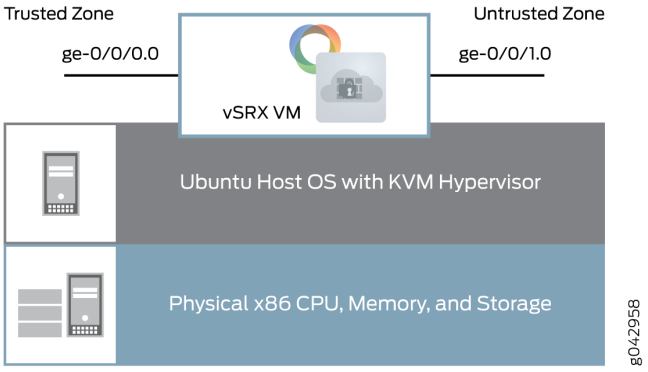
- A loadable kernel module included in the Linux kernel that provides the basic virtualization infrastructure
- A processor-specific module

When loaded into the Linux kernel, the KVM software acts as a *hypervisor*. KVM supports *multitenancy* and allows you to run multiple vSRX VMs on the *host* OS. KVM manages and shares the system resources between the host OS and the multiple vSRX VMs.

NOTE: vSRX requires you to enable hardware-based virtualization on a host OS that contains an Intel Virtualization Technology (VT) capable processor.

Figure 1 on page 3 illustrates the basic structure of a vSRX VM on an Ubuntu server.

Figure 1: vSRX VM on Ubuntu



vSRX Scale Up Performance

Table 1 on page 4 shows the vSRX scale up performance when deployed on KVM, based on the number of vCPUs and vRAM applied to a vSRX VM along with the Junos OS release in which a particular vSRX software specification was introduced.

Table 1: vSRX Scale Up Performance

vCPUs	vRAM	NICs	Release Introduced
2 vCPUs	4 GB	• Virtio • SR-IOV (Intel 82599, X520/540)	Junos OS Release 15.1X49-D15 and Junos OS Release 17.3R1
5 vCPUs	8 GB	• Virtio • SR-IOV (Intel 82599, X520/540)	Junos OS Release 15.1X49-D70 and Junos OS Release 17.3R1
5 vCPUs	8 GB	• SR-IOV (Intel X710/ XL710)	Junos OS Release 15.1X49-D90 and Junos OS Release 17.3R1

You can scale the performance and capacity of a vSRX instance by increasing the number of vCPUs and the amount of vRAM allocated to the vSRX. The multi-core vSRX automatically selects the appropriate vCPUs and vRAM values at boot time, as well as the number of Receive Side Scaling (RSS) queues in the NIC. If the vCPU and vRAM settings allocated to a vSRX VM do not match what is currently available, the vSRX scales down to the closest supported value for the instance. For example, if a vSRX VM has 3 vCPUs and 8 GB of vRAM, vSRX boots to the smaller vCPU size, which requires a minimum of 2 vCPUs. You can scale up a vSRX instance to a higher number of vCPUs and amount of vRAM, but you cannot scale down an existing vSRX instance to a smaller setting.

NOTE: The number of RSS queues typically matches with the number of data plane vCPUs of a vSRX instance. For example, a vSRX with 4 data plane vCPUs should have 4 RSS queues.

vSRX Session Capacity Increase

vSRX solution is optimized to increase the session numbers by increasing the memory.

With the ability to increase the session numbers by increasing the memory, you can enable vSRX to:

- Provide highly scalable, flexible and high-performance security at strategic locations in the mobile network.

- Deliver the performance that service providers require to scale and protect their networks.

Run the **show security flow session summary | grep maximum** command to view the maximum number of sessions.

Starting in Junos OS Release 18.4R1, the number of flow sessions supported on a vSRX instance is increased based on the vRAM size used.

Starting in Junos OS Release 19.2R1, the number of flow sessions supported on a vSRX 3.0 instance is increased based on the vRAM size used.

NOTE: Maximum of 28M sessions are supported on vSRX 3.0. You can deploy vSRX 3.0 with more than 64G memory, but the maximum flow sessions can still be only 28M.

Table 2 on page 5 lists the flow session capacity.

Table 2: vSRX and vSRX 3.0 Flow Session Capacity Details

vCPUs	Memory	Flow Session Capacity
2	4 GB	0.5 M
2	6 GB	1 M
2/5	8 GB	2 M
2/5	10 GB	2 M
2/5	12 GB	2.5 M
2/5	14 GB	3 M
2/5/9	16 GB	4 M
2/5/9	20 GB	6 M

Table 2: vSRX and vSRX 3.0 Flow Session Capacity Details *(Continued)*

vCPUs	Memory	Flow Session Capacity
2/5/9	24 GB	8 M
2/5/9	28 GB	10 M
2/5/9/17	32 GB	12 M
2/5/9/17	40 GB	16 M
2/5/9/17	48 GB	20 M
2/5/9/17	56 GB	24 M
2/5/9/17	64 GB	28 M

Release History Table

Release	Description
19.2R1	Starting in Junos OS Release 19.2R1, the number of flow sessions supported on a vSRX 3.0 instance is increased based on the vRAM size used.
18.4R1	Starting in Junos OS Release 18.4R1, the number of flow sessions supported on a vSRX instance is increased based on the vRAM size used.

RELATED DOCUMENTATION

Requirements for vSRX on KVM | 7

Upgrade a Multi-core vSRX | 63

Install vSRX with KVM | 22

Requirements for vSRX on KVM

IN THIS SECTION

- Software Specifications | 7
- Hardware Specifications | 12
- Best Practices for Improving vSRX Performance | 13
- Interface Mapping for vSRX on KVM | 15
- vSRX Default Settings on KVM | 17

This section presents an overview of requirements for deploying a vSRX instance on KVM;

Software Specifications

Table 3 on page 7 lists the system software requirement specifications when deploying vSRX in a KVM environment. The table outlines the Junos OS release in which a particular software specification for deploying vSRX on KVM was introduced. You will need to download a specific Junos OS release to take advantage of certain features.

CAUTION: A Page Modification Logging (PML) issue related to the KVM host kernel might prevent the vSRX from successfully booting. If you experience this behavior with the vSRX, we recommend that you disable the PML at the host kernel level. See "Prepare Your Server for vSRX Installation" on page 20 for details about disabling the PML as part of enabling nested virtualization.

Table 3: Specifications for vSRX

Component	Specification	Release Introduced
Linux KVM	Ubuntu 14.04.5, 16.04, 16.10, and 18.04	Junos OS Release 18.4R1

Table 3: Specifications for vSRX *(Continued)*

Component	Specification	Release Introduced
Hypervisor support	Red Hat Enterprise Linux (RHEL) 7.3	
	CentOS 7.2	
	CentOS 7.6 and 7.7	Junos OS Release 19.2R1
	Red Hat Enterprise Linux (RHEL) 7.6 and 7.7	Junos OS Release 19.2R1
Memory	4 GB	Junos OS Release 15.1X49-D15 and Junos OS Release 17.3R1
	8 GB	Junos OS Release 15.1X49-D70 and Junos OS Release 17.3R1
	16 GB	Junos OS Release 15.1X49-D90 and Junos OS Release 17.3R1
	32 GB	Junos OS Release 15.1X49-D100 and Junos OS Release 17.4R1
Disk space	16 GB IDE drive	Junos OS Release 15.1X49-D15 and Junos OS Release 17.3R1
vCPUs	2 vCPUs	Junos OS Release 15.1X49-D15 and Junos OS Release 17.3R1

Table 3: Specifications for vSRX *(Continued)*

Component	Specification	Release Introduced
	5 vCPUs	Junos OS Release 15.1X49-D70 and Junos OS Release 17.3R1
	9 vCPUs	Junos OS Release 15.1X49-D90 and Junos OS Release 17.3R1
	17 vCPUs	Junos OS Release 15.1X49-D100 and Junos OS Release 17.4R1
vNICs	2-8 vNICs. • Virtio • SR-IOV (Intel 82599, X520/X540) For SR-IOV limitations, see the *Known Behavior* section of the *vSRX Release Notes*.	Junos OS Release 15.1X49-D15 and Junos OS Release 17.3R1
	• SR-IOV (X710/XL710)	Junos OS Release 15.1X49-D90
	• SR-IOV (Mellanox ConnectX-3/ConnectX-3 Pro and Mellanox ConnectX-4 EN/ConnectX-4 Lx EN)	Junos OS Release 18.1R1

Table 3: Specifications for vSRX *(Continued)*

Component	Specification	Release Introduced
	Starting in Junos OS Release 19.4R1, DPDK version 18.11 is supported on vSRX. With this feature the Mellanox Connect Network Interface Card (NIC) on vSRX now supports OSPF Multicast and VLANs.	Junos OS Release 19.4R1
	vSRX3.0 supports LiquidIO DPDK driver with KVM hypervisor. If you use the LiquidIO II smart NICs, then you can use vSRX3.0 by the VF of SR-IOV.	Junos OS Release 20.4R1

NOTE: A vSRX on KVM deployment requires you to enable hardware-based virtualization on a host OS that contains an Intel Virtualization Technology (VT) capable processor. You can verify CPU compatibility here: http://www.linux-kvm.org/page/Processor_support

Table 4 on page 10 lists the specifications on the vSRX VM.

Table 4: Specifications for vSRX 3.0

vCPU	DPDK	Hugepage	vRAM	vDisk	vNIC	Supported Junos OS Release
2	17.05	2G	4G	20G	2-8	Junos OS Release 18.2R1

Table 4: Specifications for vSRX 3.0 *(Continued)*

vCPU	DPDK	Hugepage	vRAM	vDisk	vNIC	Supported Junos OS Release
5	17.05	6G	8G	20G	2-8 vSRX supports VIRTIO on KVM hypervisor.	Junos OS Release 18.4R1
9	17.5.02	12G	16G	20G	2-8 vSRX supports VIRTIO on KVM hypervisor.	Junos OS Release 19.1R1
17	17.5.02	24G	32G	20G	2-8 vSRX supports VIRTIO on KVM hypervisor.	Junos OS Release 19.1R1

Starting in Junos OS Release 19.1R1, the vSRX instance supports guest OS using 9 or 17 vCPUs with single-root I/O virtualization over Intel X710/XL710 on Linux KVM hypervisor for improved scalability and performance.

KVM Kernel Recommendations for vSRX

Table 5 on page 11 lists the recommended Linux kernel version for your Linux host OS when deploying vSRX on KVM. The table outlines the Junos OS release in which support for a particular Linux kernel version was introduced.

Table 5: Kernel Recommendations for KVM

Linux Distribution	Linux Kernel Version	Supported Junos OS Release
CentOS	3.10.0.229 Upgrade the Linux kernel to capture the recommended version.	Junos OS Release 15.1X49-D15 and Junos OS Release 17.3R1 or later release

Table 5: Kernel Recommendations for KVM *(Continued)*

Linux Distribution	Linux Kernel Version	Supported Junos OS Release
Ubuntu	3.16	
	4.4	
RHEL	3.10	

Additional Linux Packages for vSRX on KVM

Table 6 on page 12 lists the additional packages you need on your Linux host OS to run vSRX on KVM. See your host OS documentation for how to install these packages if they are not present on your server.

Table 6: Additional Linux Packages for KVM

Package	Version	Download Link
libvirt	0.10.0	libvirt download
virt-manager (Recommended)	0.10.0	virt-manager download

Hardware Specifications

Table 7 on page 13 lists the hardware specifications for the host machine that runs the vSRX VM.

Table 7: Hardware Specifications for the Host Machine

Component	Specification
Host processor type	Intel x86_64 multi-core CPU **NOTE**: DPDK requires Intel Virtualization VT-x/VT-d support in the CPU. See *About Intel Virtualization Technology.*
Physical NIC support for vSRX	• Virtio • SR-IOV (Intel X710/XL710, X520/540, 82599) • SR-IOV (Mellanox ConnectX-3/ConnectX-3 Pro and Mellanox ConnectX-4 EN/ConnectX-4 Lx EN) **NOTE**: If using SR-IOV with either the Mellanox ConnectX-3 or ConnectX-4 Family Adapters, on the Linux host, if necessary, install the latest MLNX_OFED Linux driver. See Mellanox OpenFabrics Enterprise Distribution for Linux (MLNX_OFED). **NOTE**: You must enable the Intel VT-d extensions to provide hardware support for directly assigning physical devices per guest. See "Configure SR-IOV and PCI on KVM" on page 55.
Physical NIC support for vSRX 3.0	Support SR-IOV on X710/XL710

Best Practices for Improving vSRX Performance

Review the following practices to improve vSRX performance.

NUMA Nodes

The x86 server architecture consists of multiple sockets and multiple cores within a socket. Each socket has memory that is used to store packets during I/O transfers from the NIC to the host. To efficiently read packets from memory, guest applications and associated peripherals (such as the NIC) should reside within a single socket. A penalty is associated with spanning CPU sockets for memory accesses, which

might result in nondeterministic performance. For vSRX, we recommend that all vCPUs for the vSRX VM are in the same physical non-uniform memory access (NUMA) node for optimal performance.

CAUTION: The Packet Forwarding Engine (PFE) on the vSRX will become unresponsive if the NUMA nodes topology is configured in the hypervisor to spread the instance's vCPUs across multiple host NUMA nodes. vSRX requires that you ensure that all vCPUs reside on the same NUMA node.

We recommend that you bind the vSRX instance with a specific NUMA node by setting NUMA node affinity. NUMA node affinity constrains the vSRX VM resource scheduling to only the specified NUMA node.

Mapping Virtual Interfaces to a vSRX VM

To determine which virtual interfaces on your Linux host OS map to a vSRX VM:

1. Use the **virsh list** command on your Linux host OS to list the running VMs.

```
hostOS# virsh list
Id    Name                           State
----------------------------------------------------
9     centos1                        running
15    centos2                        running
16    centos3                        running
48    vsrx                           running
50    1117-2                         running
51    1117-3                         running
```

2. Use the **virsh domiflist** ***vsrx-name*** command to list the virtual interfaces on that vSRX VM.

```
hostOS# virsh domiflist vsrx
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet1      bridge     brem2      virtio      52:54:00:8f:75:a5
vnet2      bridge     br1        virtio      52:54:00:12:37:62
vnet3      bridge     brconnect  virtio      52:54:00:b2:cd:f4
```

NOTE: The first virtual interface maps to the fxp0 interface in Junos OS.

Interface Mapping for vSRX on KVM

Each network adapter defined for a vSRX is mapped to a specific interface, depending on whether the vSRX instance is a standalone VM or one of a cluster pair for high availability. The interface names and mappings in vSRX are shown in Table 8 on page 15 and Table 9 on page 16.

Note the following:

- In standalone mode:
  - fxp0 is the out-of-band management interface.
  - ge-0/0/0 is the first traffic (revenue) interface.
- In cluster mode:
  - fxp0 is the out-of-band management interface.
  - em0 is the cluster control link for both nodes.
  - Any of the traffic interfaces can be specified as the fabric links, such as ge-0/0/0 for fab0 on node 0 and ge-7/0/0 for fab1 on node 1.

Table 8 on page 15 shows the interface names and mappings for a standalone vSRX VM.

Table 8: Interface Names for a Standalone vSRX VM

Network Adapter	Interface Name in Junos OS for vSRX
1	fxp0
2	ge-0/0/0
3	ge-0/0/1
4	ge-0/0/2
5	ge-0/0/3

Table 8: Interface Names for a Standalone vSRX VM *(Continued)*

Network Adapter	Interface Name in Junos OS for vSRX
6	ge-0/0/4
7	ge-0/0/5
8	ge-0/0/6

Table 9 on page 16 shows the interface names and mappings for a pair of vSRX VMs in a cluster (node 0 and node 1).

Table 9: Interface Names for a vSRX Cluster Pair

Network Adapter	Interface Name in Junos OS for vSRX
1	fxp0 (node 0 and 1)
2	em0 (node 0 and 1)
3	ge-0/0/0 (node 0) ge-7/0/0 (node 1)
4	ge-0/0/1 (node 0) ge-7/0/1 (node 1)
5	ge-0/0/2 (node 0) ge-7/0/2 (node 1)

Table 9: Interface Names for a vSRX Cluster Pair *(Continued)*

Network Adapter	Interface Name in Junos OS for vSRX
6	ge-0/0/3 (node 0) ge-7/0/3 (node 1)
7	ge-0/0/4 (node 0) ge-7/0/4 (node 1)
8	ge-0/0/5 (node 0) ge-7/0/5 (node 1)

vSRX Default Settings on KVM

vSRX requires the following basic configuration settings:

- Interfaces must be assigned IP addresses.
- Interfaces must be bound to zones.
- Policies must be configured between zones to permit or deny traffic.

Table 10 on page 17 lists the factory-default settings for security policies on the vSRX.

Table 10: Factory Default Settings for Security Policies

Source Zone	Destination Zone	Policy Action
trust	untrust	permit

Table 10: Factory Default Settings for Security Policies *(Continued)*

Source Zone	Destination Zone	Policy Action
trust	trust	permit
untrust	trust	deny

RELATED DOCUMENTATION

- About Intel Virtualization Technology
- DPDK Release Notes

Unresolved topicref

SUMMARY

Unresolved topicref placeholder.

This is a placeholder for unresolved topicref links.

2
CHAPTER

Installing vSRX in KVM

Prepare Your Server for vSRX Installation | 20

Install vSRX with KVM | 22

Example: Install and Launch vSRX on Ubuntu | 28

Load an Initial Configuration on a vSRX with KVM | 45

Unresolved topicref | 48

Prepare Your Server for vSRX Installation

IN THIS SECTION

- Enable Nested Virtualization | 20
- Upgrade the Linux Kernel on Ubuntu | 21

Enable Nested Virtualization

We recommend that you enable nested *virtualization* on your host OS or OpenStack compute node. Nested virtualization is enabled by default on Ubuntu but is disabled by default on *CentOS*.

Use the following command to determine if nested virtualization is enabled on your host OS. The result should be Y.

hostOS# **cat /sys/module/kvm_intel/parameters/nested**

hostOS# Y

NOTE: APIC virtualization (APICv) does not work well with nested VMs such as those used with KVM. On Intel CPUs that support APICv (typically v2 models, for example E5 v2 and E7 v2), you must disable APICv on the host server before deploying vSRX.

To enable nested virtualization on the host OS:

1. Depending on your host operating system, perform the following:
   - On CentOS, open the **/etc/modprobe.d/dist.conf** file in your default editor.

     ```
     hostOS# vi /etc/modprobe.d/dist.conf
     ```

   - On Ubuntu, open the **/etc/modprobe.d/qemu-system-x86.conf** file in your default editor.

     ```
     hostOS# vi /etc/modprobe.d/qemu-system-x86.conf
     ```

2. Add the following line to the file:

```
hostOS# options kvm-intel nested=y enable_apicv=n
```

> **NOTE**: A Page Modification Logging (PML) issue related to the KVM host kernel might prevent the vSRX from successfully booting. We recommend that you add the following line to the file *instead* of the line listed above in Step 2:
>
> ```
> hostOS# options kvm-intel nested=y enable_apicv=n
> pml=n
> ```

3. Save the file and reboot the host OS.
4. (Optional) After the reboot, verify that nested virtualization is enabled.

```
hostOS# cat /sys/module/kvm_intel/parameters/nested
hostOS# Y
```

5. On Intel CPUs that support APICv (for example, E5 v2 and E7 v2), disable APICv on the host OS.

```
root@host# sudo rmmod kvm-intel
root@host# sudo sh -c “echo ’options
kvm-intel enable_apicv=n’ >> /etc/modprobe.d/dist.conf”
root@host# sudo modprobe kvm-intel
```

6. Optionally, verify that APICv is now disabled.

```
root@host# cat /sys/module/kvm_intel/parameters/enable_apicv
N
```

Upgrade the Linux Kernel on Ubuntu

To upgrade to the latest stable Linux kernel on Ubuntu:

1. Get and install the available updated kernel.

   hostOS:$ **sudo apt-get install linux-image-generic-lts-utopic**

2. Reboot the host OS.

   hostOS:$ **reboot**

3. Optionally, type **uname -a** in a terminal on your host OS to verify that the host OS is using the latest kernel version.

   hostOS:$ **uname -a**

```
3.16.0-48-generic
```

Install vSRX with KVM

IN THIS SECTION

- Install vSRX with virt-manager | 22
- Install vSRX with virt-install | 25

You use **virt-manager** or **virt-install** to install vSRX VMs. See your *host* OS documentation for complete details on these packages.

NOTE: To upgrade an existing vSRX instance, see *Migration, Upgrade, and Downgrade* in the *vSRX Release Notes*.

Install vSRX with virt-manager

Ensure that sure you have already installed KVM, qemu, virt-manager, and libvirt on your host OS. You must also configure the required virtual networks and storage pool in the host OS for the vSRX VM. See your host OS documentation for details.

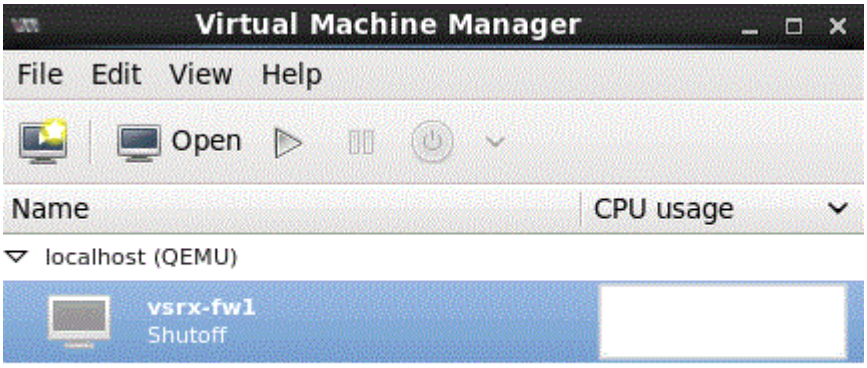
You can install and launch vSRX with the *KVM* **virt-manager** GUI package.

To install vSRX with **virt-manager**:

1. Download the vSRX QCOW2 image from the Juniper software download site.
2. On your host OS, type **virt-manager**. The Virtual Machine Manager appears. See Figure 2 on page 23.

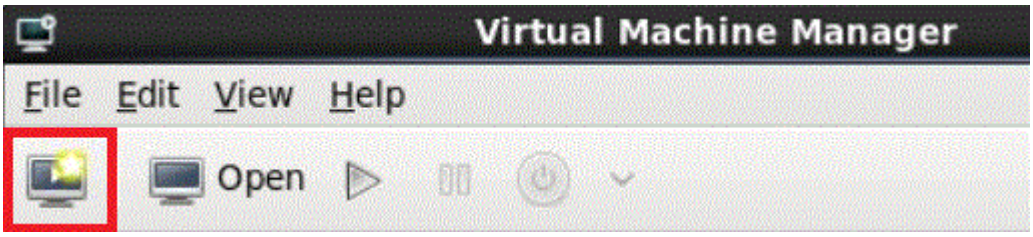
> **NOTE**: You must have admin rights on the host OS to use **virt-manager**.

Figure 2: virt-manager



3. Click **Create a new virtual machine** as seen in Figure 3 on page 23. The New VM wizard appears .

Figure 3: Create a New Virtual Machine



4. Select **Import existing disk image**, and click **Forward**.
5. Browse to the location of the downloaded vSRX QCOW2 image and select the vSRX image.
6. Select **Linux** from the OS type list and select **Show all OS options** from the Version list.
7. Select **Red Hat Enterprise Linux 7** from the expanded Version list and click **Forward**.
8. Set the RAM to 4096 MB and set CPUs to 2. Click **Forward**.
9. Set the disk image size to 16 GB and click **Forward**.
10. Name the vSRX VM, and select **Customize this configuration before install** to change parameters before you create and launch the VM. Click **Finish**. The Configuration dialog box appears.

11. Select **Processor** and expand the **Configuration** list.
12. Select **Copy Host CPU Configuration**.
13. Set CPU Feature **invtsc** to disabled on CPUs that support that feature. Set **vmx** to require for optimal throughput. You can optionally set **aes** to require for improved cryptographic throughput

NOTE: If the CPU feature option is not present in your version of **virt-manager**, you need start and stop the VM once, and then edit the vSRX VM XML file, typically found in /etc/libvirt/qemu directory on your host OS. Use **virsh edit** to edit the VM XML file to configure <feature policy='require' name='vmx'/> under the <cpu mode> element. Also add <feature policy='disable' name='invtsc'/> if your host OS supports this CPU flag. Use the **virsh capabilities** command on your host OS to list the host OS and CPU virtualization capabilities.

The following example shows the relevant portion of the vSRX XML file on a CentOS host:

```
<cpu mode='custom' match='exact'>
    <model fallback='allow'>SandyBridge</model>
    <vendor>Intel</vendor>
    <feature policy='require' name='pbe'/>
    <feature policy='require' name='tm2'/>
    <feature policy='require' name='est'/>
    <feature policy='require' name='vmx'/>
    <feature policy='require' name='osxsave'/>
    <feature policy='require' name='smx'/>
    <feature policy='require' name='ss'/>
    <feature policy='require' name='ds'/>
    <feature policy='require' name='vme'/>
    <feature policy='require' name='dtes64'/>
    <feature policy='require' name='monitor'/>
    <feature policy='require' name='ht'/>
    <feature policy='require' name='dca'/>
    <feature policy='require' name='pcid'/>
    <feature policy='require' name='tm'/>
    <feature policy='require' name='pdcm'/>
    <feature policy='require' name='pdpe1gb'/>
    <feature policy='require' name='ds_cpl'/>
    <feature policy='require' name='xtpr'/>
    <feature policy='require' name='acpi'/>
    <feature policy='disable' name='invtsc'/>
   </cpu>
```

14. Select the disk and expand **Advanced Options**.
15. Select **IDE** from the Disk bus list.
16. Select the NIC, and select **virtio** from the Device model field. This first NIC is the fpx0 (management) interface for vSRX.
17. Click **Add Hardware** to add more virtual networks, and select **virtio** from the Device model list.
18. Click **Apply**, and click **x** to close the dialog box.
19. Click **Begin Installation**. The VM manager creates and launches the vSRX VM.

> **NOTE**: The default vSRX VM login ID is root with no password. By default, if a DHCP server is on the network, it assigns an IP address to the vSRX VM.

Install vSRX with virt-install

Ensure that sure you have already installed KVM, qemu, virt-install, and libvirt on your host OS. You must also configure the required virtual networks and storage pool in the host OS for the vSRX VM. See your host OS documentation for details.

> **NOTE**: You must have root access on the host OS to use the **virt-install** command.

The **virt-install** and **virsh** tools are CLI alternatives to installing and managing vSRX VMs on a Linux host.

To install vSRX with **virt-install**:

1. Download the vSRX QCOW2 image from the Juniper software download site.
2. On your host OS, use the **virt-install** command with the mandatory options listed in Table 11 on page 26.

> **NOTE**: See the official **virt-install** documentation for a complete description of available options.

Table 11: virt-install Options

Command Option	Description
--name *name*	Name the vSRX VM.
--ram *megabytes*	Allocate RAM for the VM, in megabytes.
--cpu *cpu-model*, *cpu-flags*	Enable the **vmx** feature for optimal throughput. You can also enable **aes** for improved cryptographic throughput. **NOTE**: CPU flag support depends on your host OS and CPU. Use **virsh capabilities** to list the virtualization capabilities of your host OS and CPU.
--vcpus *number*	Allocate the number of vCPUs for the vSRX VM.
--disk *path*	Specify disk storage media and size for the VM. Include the following options: • **size=*gigabytes*** • **device=disk** • **bus=ide** • **format=qcow2**
--os-type *os-type* --os-variant *os-type*	Configure the guest OS type and variant.
--import	Create and boot the vSRX VM from an existing image.

The following example creates a vSRX VM with 4096 MB RAM, 2 vCPUs, and disk storage up to 16 GB:

```
hostOS# virt-install --name vSRXVM --ram 4096
--cpu SandyBridge,+vmx,-invtsc --vcpus=2 --arch=x86_64 --disk path=/mnt/
```

```
vsrx.qcow2,size=16,device=disk,bus=ide,format=qcow2
--os-type linux --os-variant rhel7 --import
```

The following example shows the relevant portion of the vSRX XML file on a CentOS host:

```
<cpu mode='custom' match='exact'>
    <model fallback='allow'>SandyBridge</model>
    <vendor>Intel</vendor>
    <feature policy='require' name='pbe'/>
    <feature policy='require' name='tm2'/>
    <feature policy='require' name='est'/>
    <feature policy='require' name='vmx'/>
    <feature policy='require' name='osxsave'/>
    <feature policy='require' name='smx'/>
    <feature policy='require' name='ss'/>
    <feature policy='require' name='ds'/>
    <feature policy='require' name='vme'/>
    <feature policy='require' name='dtes64'/>
    <feature policy='require' name='monitor'/>
    <feature policy='require' name='ht'/>
    <feature policy='require' name='dca'/>
    <feature policy='require' name='pcid'/>
    <feature policy='require' name='tm'/>
    <feature policy='require' name='pdcm'/>
    <feature policy='require' name='pdpe1gb'/>
    <feature policy='require' name='ds_cpl'/>
    <feature policy='require' name='xtpr'/>
    <feature policy='require' name='acpi'/>
    <feature policy='disable' name='invtsc'/>
  </cpu>
```

NOTE: The default vSRX VM login ID is root with no password. By default, if a DHCP server is on the network, it assigns an IP address to the vSRX VM.

RELATED DOCUMENTATION

- Installing a virtual machine using virt-install
- Migration, Upgrade, and Downgrade

Linux CPU Flags

Example: Install and Launch vSRX on Ubuntu

IN THIS SECTION

- Requirements | 28
- Overview | 29
- Quick Configuration - Install and Launch a vSRX VM on Ubuntu | 29
- | 33
- Step by Step Configuration | 33

This example shows how to install and launch a vSRX instance on an Ubuntu server with KVM.

Requirements

This example uses the following hardware and software components:

- Generic x86 server
- Junos OS Release 15.1X49-D20 for vSRX
- Ubuntu version 14.04.2

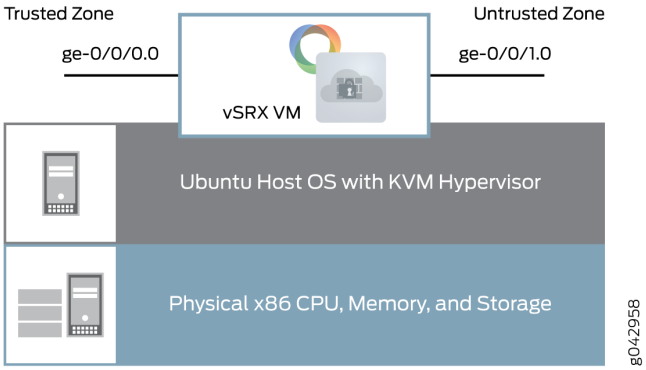
Before you begin:

- This example assumes a fresh install of the Ubuntu server software.
- Ensure that your host OS meets the requirements specified in "Requirements for vSRX on KVM" on page 7.

Overview

This example shows how to set up your Ubuntu host server and install and launch a vSRX VM. Figure 4 on page 29 shows the basic structure of a vSRX VM on an Ubuntu server.

Figure 4: vSRX VM on Ubuntu



NOTE: This example uses static IP addresses. If you are configuring the vSRX instance in an *NFV* environment, you should use DHCP.

Quick Configuration - Install and Launch a vSRX VM on Ubuntu

IN THIS SECTION

- CLI Quick Configuration | 30
- Procedure | 30

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and copy and paste the commands into the Ubuntu server terminal or vSRX console as specified.

Procedure

Step-by-Step Procedure

1. If the default virtual network does not already exist, copy the following commands and paste them into the Ubuntu server terminal to create the default virtual network.

```
cat <<EOF> /etc/libvirt/qemu/networks/default.xml
<network>
 <name>default</name>
 <forward mode='nat'/>
<nat>
 <port start ='1024' end='65535' />
 </nat>
<bridge name='virbr0' stp='on' delay='0' />
 <ip address='192.168.2.1' netmask='255.255.255.0'>
 <dhcp>
      <range start='192.168.2.2' end='192.168.2.254' />
 </dhcp>
 </ip>
</network>
 EOF
virsh net-define /etc/libvirt/qemu/networks/default.xml
virsh net-start default
virsh net-autostart default
```

2. Create the left, or trusted, virtual network on the Ubuntu server.

```
cat <<EOF> /etc/libvirt/qemu/networks/testleftnetwork.xml
<network>
 <name>TestLeft</name>
 <forward mode='route'/>
 <bridge name='virbr1' stp='on' delay='0' />
```

```
 <ip address='192.168.123.1' netmask='255.255.255.0'>
 <dhcp>
      <range start='192.168.123.100' end='192.168.123.250' />
 </dhcp>
 </ip>
</network>
 EOF
virsh net-define /etc/libvirt/qemu/networks/testleftnetwork.xml
virsh net-start TestLeft
virsh net-autostart TestLeft
```

3. Create the right, or untrusted, virtual network on the Ubuntu server.

```
cat <<EOF > /etc/libvirt/qemu/networks/testrightnetwork.xml
<network>
    <name>TestRight</name>
    <forward mode='nat'/>
<nat>
 <port start ='1024' end='65535' />
 </nat>
    <bridge name='virbr2' stp='on' delay='0' />
    <ip address='192.168.124.1' netmask='255.255.255.0'>
    <dhcp>
       <range start='192.168.124.100' end='192.168.124.250' />
   </dhcp>
 </ip>
</network>
EOF
virsh net-define /etc/libvirt/qemu/networks/testrightnetwork.xml
virsh net-start TestRight
virsh net-autostart TestRight
```

4. Download the vSRX KVM image from the Juniper Networks website at https://www.juniper.net/support/downloads/?p=vsrx#sw.

5. Copy the following commands and modify the **cpu** parameter and flags to match your Ubuntu server CPU. Paste the resulting commands into the Ubuntu server terminal to copy the image to a mount point and create the vSRX VM.

```
cp junos-vsrx-vmdisk-15.1X49-D20.2.qcow2 /mnt/vsrx20one.qcow2
virt-install  --name vSRX20One --ram 4096 --cpu SandyBridge,+vmx,-invtc, --vcpus=2 --arch=x86_64 --disk path=/mnt/vsrx20one.qcow2,size=16,device=disk,bus=ide,format=qcow2 --os-type linux --os-variant rhel7 --import --network=network:default,model=virtio --network=network:TestLeft,model=virtio --network=network:TestRight,model=virtio
```

NOTE: The CPU model and flags in the **virt-install** command might vary based on the CPU and features in the Ubuntu server.

6. To set the root password on the vSRX VM, copy and paste the command into the vSRX CLI at the **[edit]** hierarchy level.

```
set system root-authentication plain-text-password
```

7. To create a base configuration on the vSRX VM, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the following commands into the vSRX CLI at the **[edit]** hierarchy level, and then enter **commit** from configuration mode.

```
set interfaces fxp0 unit 0 family inet dhcp-client
set interfaces ge-0/0/0 unit 0 family inet address 192.168.123.254/24
set interfaces ge-0/0/1 unit 0 family inet dhcp-client
set security zones security-zone trust interfaces ge-0/0/0.0 host-inbound-traffic system-services all
set security zones security-zone untrust interfaces ge-0/0/1.0 host-inbound-traffic system-services dhcp
set routing-instances CUSTOMER-VR instance-type virtual-router
set routing-instances CUSTOMER-VR interface ge-0/0/0.0
set routing-instances CUSTOMER-VR interface ge-0/0/1.0
set security nat source rule-set source-nat from zone trust
set security nat source rule-set source-nat to zone untrust
set security nat source rule-set source-nat  rule nat1 match source-address 0.0.0.0/0
set security nat source rule-set source-nat rule nat1 then source-nat interface
```

IN THIS SECTION

- | 33

Step-by-Step Procedure

Step by Step Configuration

IN THIS SECTION

- Add Virtual Networks | 33
- Verify the Virtual Networks | 37
- Download and Installing the vSRX Image | 37
- Verify the vSRX Installation | 38
- Create a Base Configuration on the vSRX Instance | 40
- Verify the Basic Configuration on the vSRX Instance | 44

Use the following sections for a more detailed set of procedures to install and launch a vSRX VM.

Add Virtual Networks

Step-by-Step Procedure

You need to create virtual networks on the Ubuntu server to provide network connectivity to interfaces on the vSRX VM. Copy and paste these command into a terminal on the Ubuntu server.

This example uses three virtual networks:

- default— Connects the fxp0 management interface.

> **NOTE**: The default virtual network should already exist on the Ubuntu server. Use the **virsh net-list** command to verify that the default network is present and active.

- TestLeft— Connects the ge-0/0/0 interface to the trusted zone.
- TestRight— Connects the ge-0/0/1 interface to the untrusted zone.

1. If the default network does not exist, follow these steps:

Step-by-Step Procedure

a. Open a text editor on the Ubuntu server and create the default network XML (**default.xml**) file.

```
emacs /etc/libvirt/qemu/networks/default.xml
```

b. Set the forward mode to **nat**, configure an IP address and subnet mask, and a bridge interface, and configure DHCP to assign IP addresses to interfaces on this virtual network.

> **NOTE**: Use the XML format specified by libvirt.

```
<network>
 <name>default</name>
 <forward mode='nat'/>
<nat>
 <port start ='1024' end='65535' />
 </nat>
<bridge name='virbr0' stp='on' delay='0' />
 <ip address='192.168.2.1' netmask='255.255.255.0'>
 <dhcp>
      <range start='192.168.2.2' end='192.168.2.254' />
 </dhcp>
 </ip>
</network>
```

c. Define and start the default virtual network, based on the **default.xml** file you created.

```
virsh net-define /etc/libvirt/qemu/networks/default.xml
virsh net-start default
virsh net-autostart default
```

2. Remove any previously configured TestLeft virtual network.

```
virsh net-destroy TestLeft
virsh net-undefine TestLeft
```

3. Remove any previously configured TestRight virtual network.

```
virsh net-destroy TestRight
virsh net-undefine TestRight
```

4. Open a text editor on the Ubuntu server and create the TestLeft network XML (**testleftnetwork.xml**) file.

```
emacs /etc/libvirt/qemu/networks/testleftnetwork.xml
```

5. Set the forward mode to **route**, configure an IP address and subnet mask, and a bridge interface, and configure DHCP to assign IP addresses to interfaces on this virtual network.

NOTE: Use the XML format specified by libvirt.

```
<network>
 <name>TestLeft</name>
 <forward mode='route'/>
 <bridge name='virbr1' stp='on' delay='0' />
 <ip address='192.168.123.1' netmask='255.255.255.0'>
 <dhcp>
      <range start='192.168.123.100' end='192.168.123.250' />
 </dhcp>
```

```
 </ip>
</network>
```

6. Open a text editor on the Ubuntu server and create the TestRight network XML (**testrightnetwork.xml**) file.

```
emacs /etc/libvirt/qemu/networks/testrightnetwork.xml
```

7. Set the forward mode to **nat**, configure an IP address and subnet mask, and a bridge interface, and configure DHCP to assign IP addresses to interfaces on this virtual network.

> **NOTE**: Use the XML format specified by libvirt.

```
<network>
    <name>TestRight</name>
 <forward mode='nat'/>
<nat>
 <port start ='1024' end='65535' />
 </nat>
    <bridge name='virbr2' stp='on' delay='0' />
    <ip address='192.168.124.1' netmask='255.255.255.0'>
    <dhcp>
       <range start='192.168.124.100' end='192.168.124.250' />
   </dhcp>
 </ip>
</network>
```

8. Define and start the TestLeft virtual network, based on the **testleftnetwork.xml** file you created.

```
virsh net-define /etc/libvirt/qemu/networks/testleftnetwork.xml
virsh net-start TestLeft
virsh net-autostart TestLeft
```

9. Define and start the TestRight virtual network, based on the **testrightnetwork.xml** file you created.

```
virsh net-define /etc/libvirt/qemu/networks/testrightnetwork.xml
virsh net-start TestRight
virsh net-autostart TestRight
```

Verify the Virtual Networks

Purpose

Verify the new virtual network configuration on the Ubuntu server.

Action

Use the **virsh net-list** command on the Ubuntu server to verify that the new virtual interfaces are active and are set to autostart on reboot.

```
virsh net-list
 Name             State      Autostart     Persistent
----------------------------------------------------------
default                 active      yes            yes
TestLeft                active      yes            yes
TestRight               active      yes            yes
```

Download and Installing the vSRX Image

Step-by-Step Procedure

To download and install the vSRX image on the Ubuntu server:

1. Download the vSRX KVM image from the Juniper Networks website: https://www.juniper.net/support/downloads/?p=vsrx#sw

2. Copy the vSRX image to an appropriate mount point.

```
hostOS# cp junos-vsrx-vmdisk-15.1X49-D20.2.qcow2
/mnt/vsrx20one.qcow2
```

3. Use the **virt-install** command to create a vSRX VM. Modify the **cpu** parameter and flags to match your Ubuntu server CPU.

```
hostOS# virt-install  --name vSRX20One --ram
4096 --cpu SandyBridge,+vmx,-invtc, --vcpus=2 --arch=x86_64 --disk
path=/mnt/vsrx20one.qcow2,size=16,device=disk,bus=ide,format=qcow2
--os-type linux --os-variant rhel7 --import --
network=network:default,model=virtio
--network=network:TestLeft,model=virtio --
network=network:TestRight,model=virtio
```

NOTE: The CPU model and flags in the **virt-install** command might vary based on the CPU and features in the Ubuntu server.

Verify the vSRX Installation

Purpose

Verify the vSRX Installation.

Action

1. Use the **virsh console** command on the Ubuntu server to access the vSRX console and watch the progress of the installation. The installation can take several minutes to complete.

```
hostOS# virsh console vSRx200ne
Starting install...
ERROR    internal error: process exited while connecting to monitor:
libust[11994/11994]: Warning: HOME environment variable not set. Disabling
LTTng-UST per-user tracing. (in setup_local_apps() at lttng-ust-comm.c:305)
libust[11994/11995]: Error: Error opening shm /lttng-ust-wait-5 (in
get_wait_shm() at lttng-ust-comm.c:886)
libust[11994/11995]: Error: Error opening shm /lttng-ust-wait-5 (in
get_wait_shm() at lttng-ust-comm.c:886)

  Booting `Juniper Linux'

Loading Linux ...
```

```
Consoles: serial port
BIOS drive C: is disk0
BIOS drive D: is disk1
BIOS drive E: is disk2
BIOS drive F: is disk3
BIOS 639kB/999416kB available memory

FreeBSD/i386 bootstrap loader, Revision 1.2
(builder@example.com, Thu Jul 30 23:20:10 UTC 2015)
Loading /boot/defaults/loader.conf
/kernel text=0xa3a2c0 data=0x6219c+0x11f8e0 syms=[0x4+0xb2ed0+0x4+0x1061bb]
/boot/modules/libmbpool.ko text=0xce8 data=0x114
/boot/modules/if_em_vsrx.ko text=0x184c4 data=0x7fc+0x20
/boot/modules/virtio.ko text=0x2168 data=0x208 syms=[0x4+0x7e0+0x4+0x972]
/boot/modules/virtio_pci.ko text=0x2de8 data=0x200+0x8
syms=[0x4+0x8f0+0x4+0xb22]
/boot/modules/virtio_blk.ko text=0x299c data=0x1dc+0xc
syms=[0x4+0x960+0x4+0xa0f]
/boot/modules/if_vtnet.ko text=0x5ff0 data=0x360+0x10
syms=[0x4+0xdf0+0x4+0xf19]
/boot/modules/pci_hgcomm.ko text=0x12fc data=0x1a4+0x44
syms=[0x4+0x560+0x4+0x61d]
/boot/modules/chassis.ko text=0x9bc data=0x1d0+0x10 syms=[0x4+0x390+0x4+0x399]


Hit [Enter] to boot immediately, or space bar for command prompt.
Booting [/kernel]...
platform_early_bootinit: Early Boot Initialization
GDB: debug ports: sio
GDB: current port: sio
KDB: debugger backends: ddb gdb
KDB: current backend: ddb
Copyright (c) 1996-2015, Juniper Networks, Inc.
All rights reserved.
Copyright (c) 1992-2007 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
    The Regents of the University of California. All rights reserved.
FreeBSD is a registered trademark of The FreeBSD Foundation.
JUNOS 15.1X49-D15.4 #0: 2015-07-31 02:20:21 UTC

<output omitted>

The machine id is empty.
```

```
Cleaning up ...
Thu Aug 27 12:06:22 UTC 2015
Aug 27 12:06:22 init: exec_command: /usr/sbin/dhcpd (PID 1422) started
Aug 27 12:06:22 init: dhcp (PID 1422) started
Aug 27 12:06:23 init: exec_command: /usr/sbin/pppd (PID 1428) started

Amnesiac (ttyd0)

login:
```

2. On the vSRX console, log in and verify the vSRX version installed.

```
login: root
--- JUNOS 15.1X49-D15.4 built 2015-07-31 02:20:21 UTC

root@%

root@% cli
root>
root> show version
Model: vSRX
Junos: 15.1X49-D15.4
JUNOS Software Release [15.1X49-D15.4]
```

Create a Base Configuration on the vSRX Instance

Step-by-Step Procedure

To configure a base setup on the vSRX instance, enter the following steps in **edit** mode:

1. Create a root password.

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

2. Set the IP address family for the management interface, and enable the DHCP client for this interface.

```
set interfaces fxp0 unit 0 family inet dhcp-client
```

3. Set the IP address for the ge-0/0/0.0 interface.

```
set interfaces ge-0/0/0 unit 0 family inet address 192.168.123.254/24
```

4. Set the IP address family for the ge-0/0/1.0 interface, and enable the DHCP client for this interface.

```
set interfaces ge-0/0/1 unit 0 family inet dhcp-client
```

5. Add the ge-0/0/0.0 interface to the trust security zone and allow all system services from inbound traffic on that interface.

```
set security zones security-zone trust interfaces ge-0/0/0.0 host-inbound-traffic system-services all
```

6. Add the ge-0/0/1.0 interface to the untrust security zone and allow only DHCP system services from inbound traffic on that interface.

```
set security zones security-zone untrust interfaces ge-0/0/1.0 host-inbound-traffic system-services dhcp
```

7. Create a virtual router routing instance and add the two interfaces to that routing instance.

```
set routing-instances CUSTOMER-VR instance-type virtual-router
set routing-instances CUSTOMER-VR interface ge-0/0/0.0
set routing-instances CUSTOMER-VR interface ge-0/0/1.0
```

8. Create a source NAT rule set.

```
set security nat source rule-set source-nat from zone trust
set security nat source rule-set source-nat to zone untrust
```

9. Configure a rule that matches packets and translates the source address to the address of the egress interface.

```
set security nat source rule-set source-nat  rule nat1 match source-address 0.0.0.0/0
set security nat source rule-set source-nat rule nat1 then source-nat interface
```

Results

From configuration mode, confirm your configuration by entering the **show interfaces** command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
show interfaces

ge-0/0/0 {
    unit 0 {
        family inet {
            192.168.123.254/24;
        }
    }
}
ge-0/0/1 {
    unit 0 {
        family inet {
            ##
            ##              ##
            dhcp-client;
        }
    }
}
fxp0 {
    unit 0 {
        family inet {
            ##
            ##
            ##
            dhcp-client;
        }
    }
```

```
}
```

From configuration mode, confirm your security policies by entering the **show security policies** command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
show security policies
from-zone trust to-zone trust {
    policy default-permit {
        match {
            source-address any;
            destination-address any;
            application any;
        }
        then {
            permit;
        }
    }
}
from-zone trust to-zone untrust {
    policy default-permit {
        match {
            source-address any;
            destination-address any;
            application any;
        }
        then {
            permit;
        }
    }
}
from-zone untrust to-zone trust {
    policy default-deny {
        match {
            source-address any;
            destination-address any;
            application any;
        }
        then {
            deny;
        }
```

```
    }
}
```

If you are done configuring the device, enter **commit** from configuration mode.

> **NOTE**: As a final step, exit configuration mode and use the **request system reboot** command to reboot the vSRX VM. You can use the **virsh console** command on the Ubuntu server to reconnect to the vSRX after reboot.

Verify the Basic Configuration on the vSRX Instance

Purpose

Verify the basic configuration on the vSRX instance.

Action

Verify that the ge-0/0/0.0 interface has an assigned IP address from the TestLeft network DHCP address range, and that the ge-0/0/1.0 has an assigned IP address from the TestRight network DHCP address range.

```
root> show interfaces terse
Interface               Admin Link Proto    Local                 Remote
ge-0/0/0                up    up
ge-0/0/0.0              up    up   inet     192.168.123.254/24
gr-0/0/0                up    up
ip-0/0/0                up    up
lsq-0/0/0               up    up
lt-0/0/0                up    up
mt-0/0/0                up    up
sp-0/0/0                up    up
sp-0/0/0.0              up    up   inet
                                   inet6
sp-0/0/0.16383          up    up   inet
ge-0/0/1                up    up
ge-0/0/1.0              up    up   inet     192.168.124.238/24
dsc                     up    up
em0                     up    up
em0.0                   up    up   inet     128.0.0.1/2
em1                     up    up
```

```
em1.32768               up    up   inet     192.168.1.2/24
em2                     up    up
fxp0                    up    up
fxp0.0                  up    up   inet     192.168.2.1/24
ipip                    up    up
irb                     up    up
lo0                     up    up
lo0.16384               up    up   inet     127.0.0.1           --> 0/0
lo0.16385               up    up   inet     10.0.0.1            --> 0/0
                                    10.0.0.16           --> 0/0
                                    128.0.0.1           --> 0/0
                                    128.0.0.4           --> 0/0
                                    128.0.1.16          --> 0/0
lo0.32768               up    up
lsi                     up    up
mtun                    up    up
pimd                    up    up
pime                    up    up
pp0                     up    up
ppd0                    up    up
ppe0                    up    up
st0                     up    up
tap                     up    up
vlan                    up    down
```

RELATED DOCUMENTATION

- libvirt Network XML Format
- libvirt Command Reference

Load an Initial Configuration on a vSRX with KVM

IN THIS SECTION

- Create a vSRX Bootstrap ISO Image | 46

- Provision vSRX with an ISO Bootstrap Image on KVM | 47

Starting in Junos OS Release 15.1X49-D40 and Junos OS Release 17.3R1, you can use a mounted ISO image to pass the initial startup Junos OS configuration to a vSRX VM. This ISO image contains a file in the root directory called juniper.conf. This file uses the standard Junos OS command syntax to define configuration details, such as root password, management IP address, default gateway, and other configuration statements.

The process to bootstrap a vSRX VM with an ISO configuration image is as follows:

1. Create the **juniper.conf** configuration file with your Junos OS configuration.
2. Create an ISO image that includes the **juniper.conf** file.
3. Mount the ISO image to the vSRX VM.
4. Boot or reboot the vSRX VM. vSRX will boot using the **juniper.conf** file included in the mounted ISO image.
5. Unmount the ISO image from the vSRX VM.

NOTE: If you do not unmount the ISO image after the initial boot or reboot, all subsequent configuration changes to the vSRX are overwritten by the ISO image on the next reboot.

Create a vSRX Bootstrap ISO Image

This task uses a Linux system to create the ISO image.

To create a vSRX bootstrap ISO image:

1. Create a configuration file in plaintext with the Junos OS command syntax and save in a file called **juniper.conf**.
2. Create a new directory.

```
hostOS$ mkdir iso_dir
```

3. Copy **juniper.conf** to the new ISO directory.

```
hostOS$ cp juniper.conf iso_dir
```

NOTE: The **juniper.conf** file must contain the full vSRX configuration. The ISO bootstrap process overwrites any existing vSRX configuration.

4. Use the Linux **mkisofs** command to create the ISO image.

```
hostOS$ mkisofs -l -o test.iso iso_dir
I: -input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 0
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 0
175 extents written (0 MB)
```

NOTE: The **-l** option allows for a long filename.

Provision vSRX with an ISO Bootstrap Image on KVM

To provision a vSRX VM from an ISO bootstrap image:

1. Use the **virsh edit** command on the KVM host server where the vSRX VM resides to add the bootstrap ISO image as a disk device.

```
<disk type='file' device='cdrom'>
      <driver name='qemu' type='raw'/>
      <source file='/home/test.iso'/>
      <target dev='hdc' bus='ide'/>
      <readonly/>
      <address type='drive' controller='0' bus='1' target='0' unit='0'/>
    </disk>
```

2. Boot or reboot the vSRX VM.

```
user@host# virsh start ixvSRX
Connected to domain ixvSRX
```

3. Optionally, use the **virsh domblklist** Linux command to verify that the bootstrap ISO image is part of the VM.

```
hostOS# virsh domblklist ixvSRX
Target     Source
------------------------------------------------
hda        /home/test/vsrx209.qcow2
hdc        /home/test/test.iso
```

4. Verify the configuration, then power down the vSRX VM to remove the ISO image.
5. Use the **virsh edit** command on the KVM host server to remove the ISO image xml statements added in step 1, and then reboot the vSRX VM.

Release History Table

Release	Description
15.1X49-D80	Starting in Junos OS Release 15.1X49-D40 and Junos OS Release 17.3R1, you can use a mounted ISO image to pass the initial startup Junos OS configuration to a vSRX VM. This ISO image contains a file in the root directory called juniper.conf. This file uses the standard Junos OS command syntax to define configuration details, such as root password, management IP address, default gateway, and other configuration statements.

RELATED DOCUMENTATION

Linux mkisofs command

Unresolved topicref

SUMMARY

Unresolved topicref placeholder.

This is a placeholder for unresolved topicref links.

3

CHAPTER

vSRX VM Management

Connect to the vSRX Management Console on KVM | 51

Add a Virtual Network to a vSRX VM with KVM | 51

Add a Virtio Virtual Interface to a vSRX VM with KVM | 53

Configure SR-IOV and PCI on KVM | 55

Upgrade a Multi-core vSRX | 63

Monitor the vSRX VM in KVM | 66

Manage the vSRX Instance on KVM | 67

Recover the Root Password for vSRX in a KVM Environment | 72

Connect to the vSRX Management Console on KVM

Ensure that you have the **virt-manager** package or **virsh** installed on your *host* OS.

To connect to the vSRX management console using **virt-manager**:

1. Launch **virt-manager**.
2. Highlight the vSRX VM you want to connect to from the list of VMs displayed.
3. Click **Open**.
4. Select **View>Text Consoles>Serial 1**. The vSRX console appears.

To connect to the vSRX VM with **virsh**:

1. Use the **virsh console** command on the Linux host OS.

```
user@host# virsh console vSRX-kvm-2
Connected to domain vSRX-kvm-2
```

2. The vSRX console appears.

RELATED DOCUMENTATION

virt-tools

Add a Virtual Network to a vSRX VM with KVM

You can extend an existing vSRX *VM* to use additional virtual networks.

To create a *virtual network* with **virt-manager**:

1. Launch **virt-manager** and select **Edit>Connection Details**. The Connection details dialog box appears.
2. Select **Virtual Networks**. The list of existing virtual networks appears.
3. Click **+** to create a new virtual network for the control link. The Create a new virtual network wizard appears.
4. Set the subnet for this virtual network and click **Forward**.
5. Optionally, select **Enable DHCP** and click **Forward**.
6. Select the network type from the list and click **Forward**.

7. Verify the settings and click **Finish** to create the virtual network.

To create a virtual network with **virsh**:

1. Use the **virsh net-define** command on the *host* OS to create an XML file that defines the new virtual network. Include the XML fields described in Table 12 on page 52 to define this network.

> **NOTE**: See the official **virsh** documentation for a complete description of available options, including how to configure IPv6 networks.

Table 12: virsh net-define XML Fields

Field	Description
<network>...</network>	Use this XML wrapper element to define a virtual network.
<name>*net-name*</name>	Specify the virtual network name.
<bridge name="*bridge-name*" />	Specify the name of the host bridge used for this virtual network.
<forward mode="*forward-option*" />	Specify routed or nat. Do not use the <forward> element for isolated mode.
<ip address="*ip-address*" netmask="*net-mask*" <dhcp range start="*start*" end="*end*" </ dhcp> </ip>	Specify the IP address and subnet mask used by this virtual network, along with the DHCP address range.

The following example shows a sample XML file that defines a new virtual network.

```
<network>
 <name>mgmt</name>
 <bridge name="vbr1" />
 <forward mode="nat" />
 <ip address="10.10.10.1" netmask="255.255.255.0" >
```

```
  <dhcp>
<range start="10.10.10.2" end="10.10.10.99" />
  </dhcp>
 </ip>
</network>
```

2. Use the **virsh net-start** command in the host OS to start the new virtual network.

   hostOS# **virsh net-start mgmt**

3. Use the **virsh net-autostart** command in the host OS to automatically start the new virtual network when the host OS boots.

   hostOS# **virsh net-autostart mgmt**

4. Optionally, use the **virsh net-list –all** command in the host OS to verify the new virtual network.

```
HostOS# # virsh net-list --all
 Name                 State      Autostart     Persistent
----------------------------------------------------------
 mgmt                 active     yes           yes
 default              active     yes           yes
```

RELATED DOCUMENTATION

virt tools

Add a Virtio Virtual Interface to a vSRX VM with KVM

You can add additional *virtio* virtual interfaces to an existing vSRX *VM* with KVM.

To add additional virtio virtual interfaces to a vSRX VM using **virt-manager**:

1. In **virt-manager**, double-click the vSRX VM and select **View>Details**. The vSRX Virtual Machine details dialog box appears.
2. Click **Add Hardware**. The Add Hardware dialog box appears.
3. Select **Network** from the left navigation panel.

4. Select the host device or virtual network on which you want this new virtual interface from the Network source list.
5. Select **virtio** from the Device model list and click **Finish**.
6. From the vSRX console, reboot the vSRX instance.

   vsrx# request system reboot.

   vSRX reboots both Junos OS and the vSRX guest VM.

To add additional virtio virtual interfaces to a vSRX VM using **virsh**:

1. Use the **virsh attach-interface** command on the host OS with the mandatory options listed in Table 13 on page 54.

   NOTE: See the official **virsh** documentation for a complete description of available options.

   Table 13: virsh attach-interface Options

Command Option	Description
--domain *name*	Specify the name of the guest VM.
--type	Specify the host OS connection type as **bridge** or **network**.
--source *interface*	Specify the physical or logical interface on the host OS to associate with this vNIC.
--target *vnic*	Specify the name for the new vNIC.
--model	Specify the vNIC model.

   The following example creates a new virtio vNIC from the host OS virbr0 bridge.

```
user@host# virsh attach-interface --domain vsrxVM
--type bridge --source virbr0 --target vsrx-mgmt --model virtio
Interface attached successfully

user@host# virsh dumpxml vsrxVM
<output omitted>
```

```
 <interface type='bridge'>
      <mac address='00:00:5e:00:53:e8'/>
      <source bridge='virbr0'/>
      <target dev='vsrx-mgmt'/>
      <model type='virtio'/>
      <alias name='net1'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x08'
function='0x0'/>
    </interface>
```

2. From the vSRX console, reboot the vSRX instance.

   vsrx# request system reboot.

   vSRX reboots both Junos OS and the VSRX guest VM.

RELATED DOCUMENTATION

- virt tools

Configure SR-IOV and PCI on KVM

IN THIS SECTION

- SR-IOV Overview | 56
- SR-IOV HA Support with Trust Mode Disabled (KVM only) | 56
- Configure an SR-IOV Interface on KVM | 59

This section includes the following topics on SR-IOV for a vSRX instance deployed on KVM:

SR-IOV Overview

vSRX on KVM supports single-root I/O virtualization (*SR-IOV*) interface types. SR-IOV is a standard that allows a single physical NIC to present itself as multiple vNICs, or virtual functions (VFs), that a *virtual machine* (VM) can attach to. SR-IOV combines with other virtualization technologies, such as Intel VT-d, to improve the I/O performance of the VM. SR-IOV allows each VM to have direct access to packets queued up for the VFs attached to the VM. You use SR-IOV when you need I/O performance that approaches that of the physical bare metal interfaces.

NOTE: SR-IOV in KVM does not remap interface numbers. The interface sequence in the vSRX VM XML file matches the interface sequence shown in the Junos OS CLI on the vSRX instance.

SR-IOV uses two PCI functions:

- Physical Functions (PFs)—Full PCIe devices that include SR-IOV capabilities. Physical Functions are discovered, managed, and configured as normal PCI devices. Physical Functions configure and manage the SR-IOV functionality by assigning Virtual Functions. When SR-IOV is disabled, the host creates a single PF on one physical NIC.
- Virtual Functions (VFs)—Simple PCIe functions that only process I/O. Each Virtual Function is derived from a Physical Function. The number of Virtual Functions a device may have is limited by the device hardware. A single Ethernet port, the Physical Device, may map to many Virtual Functions that can be shared to guests. When SR-IOV is enabled, the host creates a single PF and multiple VFs on one physical NIC. The number of VFs depends on the configuration and driver support.

SR-IOV HA Support with Trust Mode Disabled (KVM only)

IN THIS SECTION

- Understand SR-IOV HA Support with Trust Mode Disabled (KVM only) | **57**
- Configure SR-IOV support with Trust Mode Disabled (KVM only) | **58**
- Limitations | **58**

Understand SR-IOV HA Support with Trust Mode Disabled (KVM only)

A Redundant Ethernet Interface (RETH) is a virtual interface consisting of equal number of member interfaces from each participating node of an SRX cluster. All logical configurations such as IP address, QoS, zones, and VPNs are bound to this interface. Physical properties are applied to the member or child interfaces. A RETH interface has a virtual MAC address which is calculated using the cluster id. RETH has been implemented as an aggregated interface/LAG in Junos OS. For a LAG, the parent (logical) IFDs MAC address is copied to each of the child interfaces. When you configure the child interface under the RETH interface, the RETH interface's virtual MAC gets overwritten on the **current MAC address** field of the child physical interface. This also requires the virtual MAC address to be programmed on the corresponding NIC.

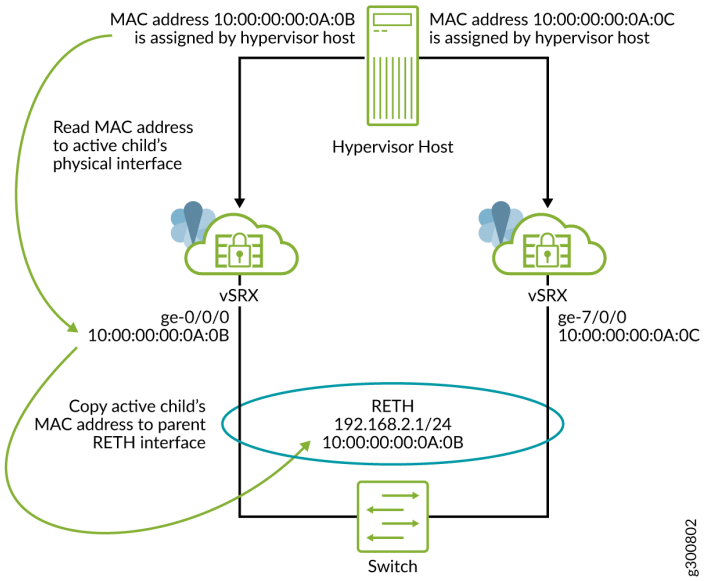
Junos OS runs as a VM on vSRX. Junos OS does not have direct access to the NIC and only has a virtual NIC access provided by the hypervisor which might be shared with other VMs running on the same host machine. This virtual access comes with certain restrictions such as a special mode called trust mode, which is required to program a virtual MAC address on the NIC. During deployments, providing the trust mode access might not be feasible because of possible security issues. To enable RETH model to work in such environments, MAC rewrite behavior is modified. Instead of copying the parent virtual MAC address to the children, we keep the children's physical MAC address intact and copy the physical MAC address of the child belonging to the active node of the cluster to the *current MAC* of the reth interface. This way, MAC rewrite access is not required when trust mode is disabled.

In case of vSRX, the DPDK reads the physical MAC address provided by the hypervisor and shares it with the Junos OS control plane. In standalone mode, this physical MAC address is programmed on the physical IFDs. But the support for the same is unavailable in cluster mode, because of which the MAC address for the physical interface is taken from the Juniper reserved MAC pool. In an environment where trust mode is not feasible, the hypervisor is unable to provide the physical MAC address.

To overcome this problem, we have added support to use the hypervisor provided physical MAC address instead of allocating it from the reserved MAC pool. See "Configure SR-IOV support with Trust Mode Disabled (KVM only)" on page 58.

Configure SR-IOV support with Trust Mode Disabled (KVM only)

Figure 5: Copying MAC address from active child interface to parent RETH



Starting in Junos OS Release 19.4R1, SR-IOV HA is supported with trust mode disabled. You can enable this mode by configuring the **use-active-child-mac-on-reth** and **use-actual-mac-on-physical-interfaces** configuration statements at the **[edit chassis cluster]** hierarchy level. If you configure commands in a cluster, the hypervisor assigns the child physical interface's MAC address and the parent RETH interface's MAC address is overwritten by the active child physical interface's MAC address

You need to reboot the vSRX instance to enable this mode. Both the nodes in the cluster need to be rebooted for the commands to take effect.

You need to configure the commands **use-active-child-mac-on-reth** and **use-actual-mac-on-physical-interfaces** together to enable this feature.

SEE ALSO

> *use-active-child-mac-on-reth*
>
> *use-actual-mac-on-physical-interfaces*

Limitations

SR-IOV HA support with trust mode disabled on KVM has the following limitations:

- SR-IOV HA support with trust mode disabled is only supported on KVM based systems.

- A reth interface can have maximum one port as a member on each vSRX cluster node.
- You cannot use **security nat proxy-arp** feature for NAT pools because no G-ARP is sent out on failover for the IP addresses in NAT pools. Instead, one can set the routes to the NAT pool range in the upstream router to point to the vSRX reth interface's IP address as the next-hop. Or, if directly connected hosts need to access the NAT pool addresses, these NAT pool addresses can be configured for proxy ARP under the reth interface.
- If the reth interface is configured with many VLANs, it might take some time to send all the G-ARPs on a failover. This might lead to a noticeable interruption in traffic.
- A dataplane failover will result in a change of the MAC address of the reth interface. Therefore the failover is not transparent to directly connected neighboring Layer 3 devices (routers or servers). The vSRX reth IP address must be mapped to a new MAC address in the ARP table on the neighboring devices. vSRX will send out a G-ARP which will help these devices. In case these neighboring devices do not act on the G-ARP received from the vSRX or show a slow response, the traffic might be interrupted until that device updates it's ARP table correctly.

Configure an SR-IOV Interface on KVM

If you have a physical NIC that supports SR-IOV, you can attach SR-IOV-enabled vNICs or virtual functions (VFs) to the vSRX instance to improve performance. We recommend that if you use SR-IOV, all revenue ports are configured as SR-IOV.

Note the following about SR-IOV support for vSRX on KVM:

- Starting in Junos OS Release 15.1X49-D90 and Junos OS Release 17.3R1, a vSRX instance deployed on KVM supports SR-IOV on an Intel X710/XL710 NIC in addition to Intel 82599 or X520/540.
- Starting in Junos OS Release 18.1R1, a vSRX instance deployed on KVM supports SR-IOV on the Mellanox ConnectX-3 and ConnectX-4 Family Adapters.

> **NOTE**: See the *vSRX Performance Scale Up* discussion in "Understand vSRX with KVM" on page 2 for the vSRX scale up performance when deployed on KVM, based on vNIC and the number of vCPUs and vRAM applied to a vSRX VM.

Before you can attach an SR-IOV enabled VF to the vSRX instance, you must complete the following tasks:

- Insert an SR-IOV-capable physical network adapter in the host server.

- Enable the Intel VT-d CPU virtualization extensions in BIOS on your host server. The Intel VT-d extensions provides hardware support for directly assigning a physical devices to guest. Verify the process with the vendor because different systems have different methods to enable VT-d.
- Ensure that SR-IOV is enabled at the system/server BIOS level by going into the BIOS settings during the host server boot-up sequence to confirm the SR-IOV setting. Different server manufacturers have different naming conventions for the BIOS parameter used to enable SR-IOV at the BIOS level. For example, for a Dell server ensure that the **SR-IOV Global Enable** option is set to **Enabled**.

NOTE: We recommend that you use **virt-manager** to configure SR-IOV interfaces. See the **virsh attach-device** command documentation if you want to learn how to add a PCI host device to a VM with the **virsh** CLI commands.

To add an SR-IOV VF to a vSRX VM using the **virt-manager** graphical interface:

1. In the Junos OS CLI, shut down the vSRX VM if it is running.

```
vsrx> request system power-off
```

2. In **virt-manager**, double-click the vSRX VM and select **View>Details**. The vSRX Virtual Machine details dialog box appears.
3. Select the Hardware tab, then click **Add Hardware**. The Add Hardware dialog box appears.
4. Select **PCI Host Device** from the Hardware list on the left.
5. Select the SR-IOV VF for this new virtual interface from the host device list.
6. Click **Finish** to add the new device. The setup is complete and the vSRX VM now has direct access to the device.
7. From the virt-manager icon bar at the upper-left side of the window, click the Power On arrow. The vSRX VM starts. Once the vSRX is powered on the Running status will display in the window.

   You can connect to the management console to watch the boot-up sequence.

   NOTE: After the boot starts, you need to select **View>Text Consoles>Serial 1** in **virt-manager** to connect to the vSRX console.

To add an SR-IOV VF to a vSRX VM using **virsh** CLI commands:

1. Define four virtual functions for eno2 interface, update the sriov_numvfs file with number 4.

```
root@LabHost:~# echo 4 > /sys/class/net/eno2/device/sriov_numvfs
root@LabHost:~# more /sys/class/net/eno2/device/sriov_numvfs
```

2. Identify the device.

   Identify the PCI device designated for device assignment to the virtual machine. Use the **lspci** command to list the available PCI devices. You can refine the output of **lspci** with **grep**.

   Use command **lspci** to check the VF number according to the VF ID.

```
root@ kvmsrv:~# lspci | grep Ether
……
83:00.0 Ethernet controller: Intel Corporation Ethernet Controller XL710 for
40GbE QSFP+ (rev 02)  -  Physical Function
83:00.1 Ethernet controller: Intel Corporation Ethernet Controller XL710 for
40GbE QSFP+ (rev 02)  -  Physical Function
83:02.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
83:02.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
83:02.2 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
83:02.3 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
83:02.4 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
83:02.5 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
83:02.6 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
83:02.7 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
83:0a.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
83:0a.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
83:0a.2 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
83:0a.3 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
```

```
83:0a.4 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
83:0a.5 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
83:0a.6 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
83:0a.7 Ethernet controller: Intel Corporation Ethernet Virtual Function 700
Series (rev 02)
.....
```

3. Add SR-IOV device assignment from a vSRX XML profile on KVM and review device information.

   The driver could use either vfio or kvm, depends on KVM server OS/kernel version and drivers for virtualization support. The address type references the unique PCI slot number for each SR-IOV VF (Virtual Function).

   Information on the domain, bus, and function are available from output of the **virsh nodedev-dumpxml** command.

```
<interface type="hostdev" managed="yes">
<driver name="vfio"/>
<source>
<address type="pci" domain="0x0000" bus="0x83" slot="0x02" function="0x3"/>
</source>
<address type="pci" domain="0x0000" bus="0x00" slot="0x05" function="0x0"/>
</interface>
```

4. Add PCI device in edit setting and select VF according to the VF number.

   NOTE: This operation should be done when VM is powered off. Also, do not clone VMs with PCI devices which might lead to VF or MAC conflict.

5. Start the VM using the **# virsh start *name of virtual machine*** command.

Release History Table

Release	Description
18.1R1	Starting in Junos OS Release 18.1R1, a vSRX instance deployed on KVM supports SR-IOV on the Mellanox ConnectX-3 and ConnectX-4 Family Adapters.

15.1X49-D90	Starting in Junos OS Release 15.1X49-D90 and Junos OS Release 17.3R1, a vSRX instance deployed on KVM supports SR-IOV on an Intel X710/XL710 NIC in addition to Intel 82599 or X520/540.

RELATED DOCUMENTATION

- Requirements for vSRX on KVM | 7
- Intel SR-IOV Explanation
- PCI-SIG SR-IOV Primer
- SR-IOV
- Intel - SR-IOV Configuration Guide
- Red Hat - SR–IOV - PCI Devices

Upgrade a Multi-core vSRX

IN THIS SECTION

- Configure the Queue Value for vSRX VM with KVM | 64
- Shutdown the vSRX Instance with virt-manager | 64
- Upgrade vSRX with virt-manager | 65

Starting in Junos OS Release 15.1X49-D70 and Junos OS Release 17.3R1, you can use **virt-manager** to scale the performance and capacity of a vSRX instance by increasing the number of vCPUs or the amount of vRAM allocated to the vSRX. See "Requirements for vSRX on KVM" on page 7 for the software requirement specifications for a vSRX VM.

See your *host* OS documentation for complete details on the **virt-manager** package

NOTE: You cannot scale down the number of vCPUs or decrease the amount of vRAM for an existing vSRX VM.

Configure the Queue Value for vSRX VM with KVM

Before you plan to scale up vSRX performance, modify the vSRX VM XML file to configure network multi-queuing as a means to support an increased number of dataplane vCPUs for the vSRX VM. This setting updates the libvirt driver to enable multi-queue virtio-net so that network performance can scale as the number of dataplane vCPUs increases. Multi-queue virtio is an approach that enables the processing of packet sending and receiving to be scaled to the number of available virtual CPUs (vCPUs) of a guest, through the use of multiple queues.

The configuration of multi-queue virtio-net, however, can only be performed in the XML file. OpenStack does not support multi-queue.

To update the queue, at the **<driver name='vhost' queues='x'/>** line in the vSRX VM XML file, match the number of queues with number of dataplane vCPUs you plan to configure for the vSRX VM. The default is 4 dataplane vCPUs, but you can scale that number to 4, 8, or 16 vCPUs.

The following XML file example configures 8 queues for a vSRX VM with 8 dataplane vCPUs:

```
<output omitted>

 <interface type='network'>
       <source network='net2'/>
     <model type='virtio'/>
     <driver name='vhost' queues='8'/>
     <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
   </interface>
```

Shutdown the vSRX Instance with virt-manager

In situations where you want to edit and modify the vSRX VM XML file, you need to completely shut down vSRX and the associated VM.

To gracefully shutdown the vSRX instance with **virt-manager**:

1. Launch **virt-manager**.
2. Check the vSRX instance you want to power off.
3. Select **Open** to open a console window to the vSRX instance.
4. From the vSRX console, reboot the vSRX instance.

vsrx# request system power-off.

5. From **virt-manager**, select **Shut Down** to completely shutdown the VM so you can edit the XML file.

> **NOTE**: Do not use **Force Reset** or **Force Off** on any active VM as it may create file corruptions.

Upgrade vSRX with virt-manager

You must shut down the vSRX VM before you can update vCPU or vRAM values for the VM.

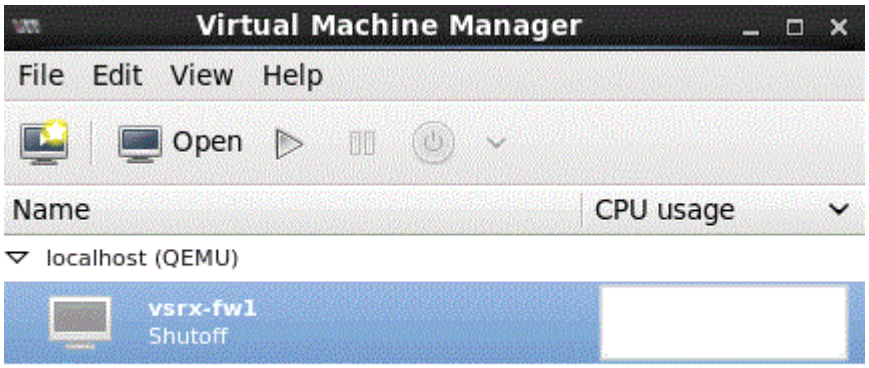
You can upgrade and launch vSRX with the *KVM* **virt-manager** GUI package.

To scale up a vSRX VM with **virt-manager** to a higher number of vCPUs or to an increased amount of vRAM:

1. On your host OS, type **virt-manager**. The Virtual Machine Manager appears. See Figure 6 on page 65.

> **NOTE**: You must have admin rights on the host OS to use **virt-manager**.

Figure 6: virt-manager



2. Select **Open** to open the powered down vSRX VM and select **Edit Hardware Details** to open the virtual machine details window.
3. Select **Processor** and set the number of vCPUs. Click **Apply**.
4. Select **Memory** and set the vRAM to the desired size. Click **Apply**.

5. Click **Power On**. The VM manager launches the vSRX VM with the new vCPU and vRAM settings.

> **NOTE**: vSRX scales down to the closest supported value if the vCPU or vRAM settings do not match what is currently available.

Release History Table

Release	Description
15.1X49-D70	Starting in Junos OS Release 15.1X49-D70 and Junos OS Release 17.3R1, you can use virt-manager to scale the performance and capacity of a vSRX instance by increasing the number of vCPUs or the amount of vRAM allocated to the vSRX

RELATED DOCUMENTATION

- Understand vSRX with KVM | 2
- Requirements for vSRX on KVM | 7
- Installing a virtual machine using virt-install

Monitor the vSRX VM in KVM

You can monitor the overall state of the vSRX *VM* with **virt-manager** or **virsh**.

To monitor the vSRX VM with **virt-manager**:

1. From the virt-manager GUI, select the vSRX VM you want to monitor.
2. Select **View>Graph** and select the statistics you want to monitor. Options include CPU, memory, disk I/O, and network interface statistics.

   The window updates with thumbnail graphs for the statistics you selected.
3. Optionally, double-click on the thumbnail graph to expand the view.

To monitor the vSRX VM with **virsh**, use the commands listed in Table 14 on page 67.

Table 14: virsh Monitor Commands

Command	Description
virsh cpu-stats *vm-name*	Lists the CPU statistics for the VM.
virsh domifstat *vm-name interface-name*	Displays the vNIC statistics for the VM.
virsh dommemstat *vm-name*	Displays memory statistics for the VM.
virsh vcpuinfo *vm-name*	Displays vCPU details for the VM.
virsh nodecpustats	Displays CPU statistics for the host OS.

RELATED DOCUMENTATION

- virt tools

Manage the vSRX Instance on KVM

IN THIS SECTION

- Power On the vSRX Instance with virt-manager | **68**
- Power On the vSRX Instance with virsh | **68**
- Pause the vSRX Instance with virt-manager | **69**
- Pause the vSRX Instance with virsh | **69**
- Rebooting the vSRX Instance with virt-manager | **69**
- Reboot the vSRX Instance with virsh | **69**
- Power Off the vSRX Instance with virt-manager | **70**
- Power Off the vSRX Instance with virsh | **70**

- Shutdown the vSRX Instance with virt-manager | 71
- Shutdown the vSRX Instance with virsh | 71
- Remove the vSRX Instance with virsh | 72

Each vSRX instance is an independent *VM* that you can power on, pause, or shut down. You can manage the vSRX VM with multiple tools, including **virt-manager** and **virsh**.

Power On the vSRX Instance with virt-manager

To power on the vSRX instance with **virt-manager**:

1. Launch **virt-manager**.
2. Check the vSRX instance you want to power on.
3. From the icon bar, select the power on arrow. The vSRX VM starts. You can connect to the management console to watch the boot-up sequence.

> **NOTE**: After the boot starts, you need to select **View>Text Consoles>Serial 1** in **virt-manager** to connect to the vSRX console.

Power On the vSRX Instance with virsh

To power on the vSRX instance with **virsh**:

Use the **virsh start** command on the *host* OS to start a vSRX VM.

```
user@host# virsh start vSRX-kvm-2
Domain vSRX-kvm-2 started
```

Pause the vSRX Instance with virt-manager

To pause the vSRX instance with **virt-manager**:

1. Launch **virt-manager**.
2. Check the vSRX instance you want to pause.
3. From the icon bar, select the power on pause icon. The vSRX VM pauses.

Pause the vSRX Instance with virsh

To pause the vSRX instance with **virsh**:

Use the **virsh suspend** command on the host OS to pause a vSRX VM.

```
user@host# virsh suspend vSRX-kvm-2
Domain vSRX-kvm-2 suspended
```

Rebooting the vSRX Instance with virt-manager

To reboot the vSRX instance with **virt-manager**:

1. Launch **virt-manager**.
2. Check the vSRX instance you want to reboot.
3. Select **Open** to open a console window to the vSRX instance.
4. From the vSRX console, reboot the vSRX instance.

   vsrx# request system reboot.

   vSRX reboots both Junos OS and the VSRX guest VM.

Reboot the vSRX Instance with virsh

To reboot the vSRX VM with **virsh**:

1. Use the **virsh console** command on the host OS to connect to the vSRX VM.

2. On the vSRX console, use the **request system reboot** command to reboot Junos OS and the vSRX VM.

```
user@host# virsh console vSRX-kvm-2
Connected to domain vSRX-kvm-2
vsrx# request system reboot
```

Power Off the vSRX Instance with virt-manager

To power off the vSRX instance with **virt-manager**:

1. Launch **virt-manager**.
2. Check the vSRX instance you want to power off.
3. Select **Open** to open a console window to the vSRX instance.
4. From the vSRX console, power off the vSRX instance.

```
vsrx> request system power-off
```

vSRX powers off both Junos OS and the guest VM.

Power Off the vSRX Instance with virsh

To power off the vSRX instance with **virsh**:

1. Use the **virsh console** command on the host OS to connect to the vSRX VM.
2. On the vSRX console, use the **request system power-off** command to power off Junos OS and the vSRX VM.

```
user@host# virsh console vSRX-kvm-2
Connected to domain vSRX-kvm-2
vsrx# request system power-off
```

Shutdown the vSRX Instance with virt-manager

In situations where you want to edit and modify the vSRX VM XML file, you need to completely shut down vSRX and the associated VM.

To gracefully shutdown the vSRX instance with **virt-manager**:

1. Launch **virt-manager**.
2. Check the vSRX instance you want to power off.
3. Select **Open** to open a console window to the vSRX instance.
4. From the vSRX console, reboot the vSRX instance.

   vsrx# request system power-off.
5. From **virt-manager**, select **Shut Down** to completely shutdown the VM so you can edit the XML file.

NOTE: Do not use **Force Reset** or **Force Off** on any active VM as it may create file corruptions.

Shutdown the vSRX Instance with virsh

In situations where you want to modify the vSRX VM XML file, you need to completely shut down vSRX and the associated VM.

To gracefully shutdown the vSRX instance with **virsh**:

1. Use the **virsh console** command on the host OS to connect to the vSRX VM.
2. On the vSRX console, use the **request system power-off** command to power off Junos OS and the vSRX VM.
3. On the host OS, use the **virsh shutdown** command to shut down the VM after vSRX has powered off.

```
user@host# virsh console vSRX-kvm-2
Connected to domain vSRX-kvm-2
vsrx# request system power-off
user@host# virsh shutdown vSRX-kvm-2
```

NOTE: Do not use the **virsh destroy** command on any active VM as it may create file corruptions.

Remove the vSRX Instance with virsh

In situations where you want to completely remove the vSRX instance, you need to destroy the vSRX VM and undefine the associated XML file.

To completely remove the vSRX instance with **virsh**:

1. On the host OS, use the **virsh destroy** command to destroy the vSRX VM.
2. On the host OS, use the **virsh undefine** command to undefine the vSRX XML file.

```
user@host# virsh destroy vSRX-kvm-2
user@host# virsh undefine vSRX-kvm-2
```

RELATED DOCUMENTATION

virt tools

Recover the Root Password for vSRX in a KVM Environment

If you forget the root password for a vSRX instance deployed in a KVM environment, use this password recovery procedure to reset the root password. (KB article 31790).

NOTE: You need console access to recover the root password

To recover the root password for a vSRX instance:

1. Reboot the vSRX instance by entering the **virsh reboot** command, specifying either ***domain-id*** or ***domain-name***.
2. Immediately attempt to login to the vSRX instance by entering the **virsh console** ***domain-name*** command to access the vSRX console.

NOTE: We recommend that you specify *domain-name* when attempting the vSRX instance login. It is possible that *domain-id* might change after the vSRX instance reboot.

3. After login you will see a prompt similar to **Escape character is ^]**. Press **Enter** two or three times until the boot process begins. Continue with the boot process until you see the following prompt:

```
Hit [Enter] to boot immediately, or space bar for command prompt. Booting
[kernel] in 9 seconds...
```

4. Press the space bar two or three times to stop the boot sequence, and then enter **boot -s** to login to single-user mode.
5. Enter **recovery** to start the root password recovery procedure.

```
Enter full pathname of shell or 'recovery' for root password recovery  or
RETURN for /bin/sh: recovery
```

Once the script terminates you will be in vSRX operational mode.

6. Enter configuration mode in the CLI.
7. Set the root password.

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

8. Enter the new root password.

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

9. At the second prompt, reenter the new root password.
10. If you are finished configuring the new root password for the vSRX instance, commit the configuration.

```
root@host# commit
commit complete
```

11. Exit from configuration mode.
12. Exit from operational mode.

13. Enter **y** to reboot the device.

```
Reboot the system? [y/n] y
```

The start up messages display on the screen.

14. Once again, press the space bar two or three times to access the bootstrap loader prompt.
15. The vSRX instance starts up again and prompts you to enter a user name and password. Enter the newly configured password.

```
login: root
Password: xxxxxxxx
```

4

CHAPTER

Configuring and Managing vSRX

Unresolved topicref | 76

Unresolved topicref | 76

Unresolved topicref | 76

Unresolved topicref | 77

Unresolved topicref | 77

Unresolved topicref | 77

Unresolved topicref

SUMMARY

Unresolved topicref placeholder.

This is a placeholder for unresolved topicref links.

Unresolved topicref

SUMMARY

Unresolved topicref placeholder.

This is a placeholder for unresolved topicref links.

Unresolved topicref

SUMMARY

Unresolved topicref placeholder.

This is a placeholder for unresolved topicref links.

Unresolved topicref

SUMMARY

Unresolved topicref placeholder.

This is a placeholder for unresolved topicref links.

Unresolved topicref

SUMMARY

Unresolved topicref placeholder.

This is a placeholder for unresolved topicref links.

Unresolved topicref

SUMMARY

Unresolved topicref placeholder.

This is a placeholder for unresolved topicref links.

5

CHAPTER

Configuring vSRX Chassis Clusters

Configure a vSRX Chassis Cluster in Junos OS | 79

vSRX Cluster Staging and Provisioning for KVM | 90

Verify the Chassis Cluster Configuration | 96

Configure a vSRX Chassis Cluster in Junos OS

IN THIS SECTION

- Chassis Cluster Overview | 79
- Enable Chassis Cluster Formation | 80
- Chassis Cluster Quick Setup with J-Web | 81
- Manually Configure a Chassis Cluster with J-Web | 83

Chassis Cluster Overview

Chassis cluster groups a pair of the same kind of vSRX instances into a cluster to provide network node redundancy. The vSRX instances in a chassis cluster must be running the same Junos OS release, and each instance becomes a node in the chassis cluster. You connect the control virtual interfaces on the respective nodes to form a *control plane* that synchronizes the configuration and Junos OS kernel state on both nodes in the cluster. The control link (a *virtual network* or *vSwitch*) facilitates the redundancy of interfaces and services. Similarly, you connect the *data plane* on the respective nodes over the fabric virtual interfaces to form a unified data plane. The fabric link (a virtual network or vSwitch) allows for the management of cross-node flow processing and for the management of session redundancy.

The control plane software operates in active/passive mode. When configured as a chassis cluster, one node acts as the primary and the other as the secondary to ensure stateful failover of processes and services in the event of a system or hardware failure on the primary . If the primary fails, the secondary takes over processing of control plane traffic.

NOTE: If you configure a chassis cluster across two hosts, disable igmp-snooping on the bridge that each host physical interface belongs to and that the control virtual NICs (vNICs) use. This ensures that the control link heartbeat is received by both nodes in the chassis cluster.

The chassis cluster data plane operates in active/active mode. In a chassis cluster, the data plane updates session information as traffic traverses either node, and it transmits information between the nodes over the fabric link to guarantee that established sessions are not dropped when a failover occurs. In active/active mode, traffic can enter the cluster on one node and exit from the other node.

Chassis cluster functionality includes:

- Resilient system architecture, with a single active control plane for the entire cluster and multiple *Packet Forwarding Engine*s. This architecture presents a single device view of the cluster.
- Synchronization of configuration and dynamic runtime states between nodes within a cluster.
- Monitoring of physical interfaces, and failover if the failure parameters cross a configured threshold.
- Support for generic routing encapsulation (*GRE*) and IP-over-IP (IP-IP) tunnels used to route encapsulated IPv4 or *IPv6* traffic by means of two internal interfaces, gr-0/0/0 and ip-0/0/0, respectively. Junos OS creates these interfaces at system startup and uses these interfaces only for processing GRE and IP-IP tunnels.

At any given instant, a cluster node can be in one of the following states: hold, primary, secondary-hold, secondary, ineligible, or disabled. Multiple event types, such as interface monitoring, Services Processing Unit (SPU) monitoring, failures, and manual failovers, can trigger a state transition.

Enable Chassis Cluster Formation

You create two vSRX instances to form a chassis cluster, and then you set the cluster ID and node ID on each instance to join the cluster. When a vSRX instance joins a cluster, it becomes a node of that cluster. With the exception of unique node settings and management IP addresses, nodes in a cluster share the same configuration.

You can deploy up to 255 chassis clusters in a *Layer 2* domain. Clusters and nodes are identified in the following ways:

- The *cluster ID* (a number from 1 to 255) identifies the cluster.
- The *node ID* (a number from 0 to 1) identifies the cluster node.

Generally, on SRX Series devices, the cluster ID and node ID are written into EEPROM. On the vSRX instance, vSRX stores and reads the IDs from **boot/loader.conf** and uses the IDs to initialize the chassis cluster during startup.

Prerequisites

Ensure that your vSRX instances comply with the following prerequisites before you enable chassis clustering:

- You have committed a basic configuration to both vSRX instances that form the chassis cluster. See *Configure vSRX Using the CLI*.
- Use **show version** in Junos OS to ensure that both vSRX instances have the same software version.

- Use **show system license** in Junos OS to ensure that both vSRX instances have the same licenses installed.

You must set the same chassis cluster ID on each vSRX node and reboot the vSRX VM to enable chassis cluster formation.

1. In operational command mode, set the chassis cluster ID and node number on vSRX node 0.

```
user@vsrx0>set chassis cluster cluster-id number node 0 reboot
```

2. In operational command mode, set the chassis cluster ID and node number on vSRX node 1.

```
user@vsrx1>set chassis cluster cluster-id number node 1 reboot
```

NOTE: The vSRX interface naming and mapping to vNICs changes when you enable chassis clustering. See "Requirements for vSRX on KVM" on page 7 for a summary of interface names and mappings for a pair of vSRX VMs in a cluster (node 0 and node 1).

Chassis Cluster Quick Setup with J-Web

To configure chassis cluster from *J-Web*:

1. Enter the vSRX node 0 interface IP address in a Web browser.
2. Enter the vSRX username and password, and click **Log In**. The J-Web dashboard appears.
3. Click **Configuration Wizards> Cluster (HA) Setup** from the left panel. The Chassis Cluster Setup Wizard appears. Follow the steps in the setup wizard to configure the cluster ID and the two nodes in the cluster, and to verify connectivity.

   NOTE: Use the built-in Help icon in J-Web for further details on the Chassis Cluster Setup wizard.

   NOTE: Navigate to **Configure>Device Settings>Cluster (HA) Setup** from Junos OS release 18.1 and later to configure the chassis cluster setup.

4. Configure the secondary node Node1 by selecting **Yes, this is the secondary unit to be setup (Node 1)** using radio button.
5. Click **Next**.
6. Specify the settings such as **Enter password**, **Re-enter password**, **Node 0 FXP0 IP**, and **Node 1 FXP0 IP** for secondary node access.
7. Click **Next**.
8. Select the secondary unit's Control Port and Fabric Port.
9. Click **Next**.
10. (Optional) Select **Save a backup file before proceeding with shutdown** using check box to re-configure it for chassis cluster.
11. Click **Next**.
12. Click **Shutdown and continue** to connect to other unit.
13. Click **Refresh Browser**.
14. Configure the primary node Node0 by selecting **No, this is the primary unit to be setup (Node 0)** to configure primary unit and establish a chassis cluster configuration.
15. Click **Next**.
16. Specify the settings such as **Enter password**, **Re-enter password**, **Node 0 FXP0 IP**, and **Node 1 FXP0 IP** for primary node access.
17. Click **Next** to restart the primary unit.
18. (Optional) Select **Save a backup file before proceeding with shutdown** to save a backup file of current settings before proceeding.
19. Click **Reboot and continue**. After completing the reboot, power on the secondary unit to establish the chassis cluster connection.
20. Login to the device console and add static route to get the J-Web access.
21. Login to the J-Web and click **Configuration Wizards> Cluster (HA) Setup** from the left panel. The Chassis Cluster Setup Wizard appears.
22. Click **Next** to get the primary unit connected.
23. Configure the basic settings **DHCP Client**, **IP address**, **Default gateway**, **Member interface Node 0**, **Member interface Node 1**.
24. Click **Next** to complete the chassis cluster configuration.
25. Click **Finish** to exit the wizard. You can access the primary node using J-Web.

Manually Configure a Chassis Cluster with J-Web

You can use the *J-Web* interface to configure the primary node 0 vSRX instance in the cluster. Once you have set the cluster and node IDs and rebooted each vSRX, the following configuration will automatically be synced to the secondary node 1 vSRX instance.

Select **Configure>Chassis Cluster>Cluster Configuration**. The Chassis Cluster configuration page appears.

> **NOTE**: Navigate to **Configure>Device Settings>Cluster (HA) Setup** from Junos OS release 18.1 and later to configure the HA cluster setup.

Table 15 on page 83 explains the contents of the HA Cluster Settings tab.

Table 16 on page 85 explains how to edit the Node Settings tab.

Table 17 on page 86 explains how to add or edit the HA Cluster Interfaces table.

Table 18 on page 87 explains how to add or edit the HA Cluster Redundancy Groups table.

Table 15: Chassis Cluster Configuration Page

Field	Function
Node Settings	
Node ID	Displays the node ID.
Cluster ID	Displays the cluster ID configured for the node.
Host Name	Displays the name of the node.
Backup Router	Displays the router used as a gateway while the Routing Engine is in secondary state for redundancy-group 0 in a chassis cluster.
Management Interface	Displays the management interface of the node.

Table 15: Chassis Cluster Configuration Page *(Continued)*

Field	Function
IP Address	Displays the management IP address of the node.
Status	Displays the state of the redundancy group. • **Primary**–Redundancy group is active. • **Secondary**–Redundancy group is passive.
Chassis Cluster>HA Cluster Settings>Interfaces	
Name	Displays the physical interface name.
Member Interfaces/IP Address	Displays the member interface name or IP address configured for an interface.
Redundancy Group	Displays the redundancy group.
Chassis Cluster>HA Cluster Settings>Redundancy Group	
Group	Displays the redundancy group identification number.
Preempt	Displays the selected preempt option. • **True**–Primary Role can be preempted based on priority. • **False**–Primary Role cannot be preempted based on priority.
Gratuitous ARP Count	Displays the number of gratuitous Address Resolution Protocol (*ARP*) requests that a newly elected primary device in a chassis cluster sends out to announce its presence to the other network devices.

Table 15: Chassis Cluster Configuration Page *(Continued)*

Field	Function
Node Priority	Displays the assigned priority for the redundancy group on that node. The eligible node with the highest priority is elected as primary for the redundant group.

Table 16: Edit Node Setting Configuration Details

Field	Function	Action
Node Settings		
Host Name	Specifies the name of the host.	Enter the name of the host.
Backup Router	Displays the device used as a gateway while the Routing Engine is in the secondary state for redundancy-group 0 in a chassis cluster.	Enter the IP address of the backup router.
Destination		
IP	Adds the destination address.	Click **Add**.
Delete	Deletes the destination address.	Click **Delete**.
Interface		
Interface	Specifies the interfaces available for the router. **NOTE**: Allows you to add and edit two interfaces for each fabric link.	Select an option.
IP	Specifies the interface IP address.	Enter the interface IP address.

Table 16: Edit Node Setting Configuration Details *(Continued)*

Field	Function	Action
Add	Adds the interface.	Click **Add**.
Delete	Deletes the interface.	Click **Delete**.

Table 17: Add HA Cluster Interface Configuration Details

Field	Function	Action
Fabric Link > Fabric Link 0 (fab0)		
Interface	Specifies fabric link 0.	Enter the interface IP fabric link 0.
Add	Adds fabric interface 0.	Click **Add**.
Delete	Deletes fabric interface 0.	Click **Delete**.
Fabric Link > Fabric Link 1 (fab1)		
Interface	Specifies fabric link 1.	Enter the interface IP for fabric link 1.
Add	Adds fabric interface 1.	Click **Add**.
Delete	Deletes fabric interface 1.	Click **Delete**.
Redundant Ethernet		
Interface	Specifies a logical interface consisting of two physical Ethernet interfaces, one on each chassis.	Enter the logical interface.

Table 17: Add HA Cluster Interface Configuration Details *(Continued)*

Field	Function	Action
IP	Specifies a redundant Ethernet IP address.	Enter a redundant Ethernet IP address.
Redundancy Group	Specifies the redundancy group ID number in the chassis cluster.	Select a redundancy group from the list.
Add	Adds a redundant Ethernet IP address.	Click **Add**.
Delete	Deletes a redundant Ethernet IP address.	Click **Delete**.

Table 18: Add Redundancy Groups Configuration Details

Field	Function	Action
Redundancy Group	Specifies the redundancy group name.	Enter the redundancy group name.
Allow preemption of primaryship	Allows a node with a better priority to initiate a failover for a redundancy group. **NOTE**: By default, this feature is disabled. When disabled, a node with a better priority does not initiate a redundancy group failover (unless some other factor, such as faulty network connectivity identified for monitored interfaces, causes a failover).	–
Gratuitous ARP Count	Specifies the number of gratuitous Address Resolution Protocol requests that a newly elected primary sends out on the active redundant Ethernet interface child links to notify network devices of a change in primary role on the redundant Ethernet interface links.	Enter a value from 1 to 16. The default is 4.

Table 18: Add Redundancy Groups Configuration Details *(Continued)*

Field	Function	Action
node0 priority	Specifies the priority value of node0 for a redundancy group.	Enter the node priority number as 0.
node1 priority	Specifies the priority value of node1 for a redundancy group.	Select the node priority number as 1.
Interface Monitor		
Interface	Specifies the number of redundant Ethernet interfaces to be created for the cluster.	Select an interface from the list.
Weight	Specifies the weight for the interface to be monitored.	Enter a value from 1 to 125.
Add	Adds interfaces to be monitored by the redundancy group along with their respective weights.	Click **Add**.
Delete	Deletes interfaces to be monitored by the redundancy group along with their respective weights.	Select the interface from the configured list and click **Delete**.
IP Monitoring		
Weight	Specifies the global weight for IP monitoring.	Enter a value from 0 to 255.
Threshold	Specifies the global threshold for IP monitoring.	Enter a value from 0 to 255.
Retry Count	Specifies the number of retries needed to declare reachability failure.	Enter a value from 5 to 15.

Table 18: Add Redundancy Groups Configuration Details *(Continued)*

Field	Function	Action
Retry Interval	Specifies the time interval in seconds between retries.	Enter a value from 1 to 30.
IPV4 Addresses to Be Monitored		
IP	Specifies the IPv4 addresses to be monitored for reachability.	Enter the IPv4 addresses.
Weight	Specifies the weight for the redundancy group interface to be monitored.	Enter the weight.
Interface	Specifies the logical interface through which to monitor this IP address.	Enter the logical interface address.
Secondary IP address	Specifies the source address for monitoring packets on a secondary link.	Enter the secondary IP address.
Add	Adds the IPv4 address to be monitored.	Click **Add**.
Delete	Deletes the IPv4 address to be monitored.	Select the IPv4 address from the list and click **Delete**.

SEE ALSO

Chassis Cluster Feature Guide for Security Devices

vSRX Cluster Staging and Provisioning for KVM

IN THIS SECTION

- Chassis Cluster Provisioning on vSRX | 90
- Creating the Chassis Cluster Virtual Networks with virt-manager | 92
- Creating the Chassis Cluster Virtual Networks with virsh | 92
- Configuring the Control and Fabric Interfaces with virt-manager | 94
- Configuring the Control and Fabric Interfaces with virsh | 94
- Configuring Chassis Cluster Fabric Ports | 95

You can provision the vSRX VMs and virtual networks to configure chassis clustering.

The staging and provisioning of the vSRX *chassis cluster* includes the following tasks:

Chassis Cluster Provisioning on vSRX

Chassis cluster requires the following direct connections between the two vSRX instances:

- Control link, or *virtual network*, which acts in active/passive mode for the control plane traffic between the two vSRX instances
- Fabric link, or virtual network, which acts in active/active mode for the data traffic between the two vSRX instances

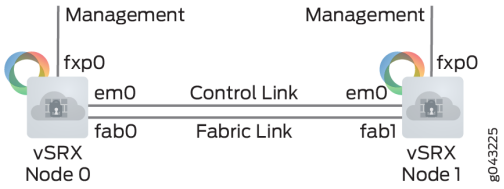
> **NOTE**: You can optionally create two fabric links for more redundancy.

The vSRX cluster uses the following interfaces:

- Out-of-band Management interface (fxp0)
- Cluster control interface (em0)
- Cluster fabric interface (fab0 on node0, fab1 on node1)

NOTE: The control interface must be the second *vNIC*. You can optionally configure a second fabric link for increased redundancy.

Figure 7: vSRX Chassis Cluster



vSRX supports chassis cluster using the virtio driver and interfaces, with the following considerations:

- When you enable chassis cluster, you must also enable jumbo frames (MTU size = 9000) to support the fabric link on the virtio network interface.
- If you configure a chassis cluster across two physical hosts, disable igmp-snooping on each host physical interface that the vSRX control link uses to ensure that the control link heartbeat is received by both nodes in the chassis cluster.

```
hostOS# echo 0 > /sys/devices/virtual/net/<bridge-name>/bridge/
multicast_snooping
```

- After you enable chassis cluster, the vSRX instance maps the second vNIC to the control link, em0. You can map any other vNICs to the fabric link.

NOTE: For virtio interfaces, link status update is not supported. The link status of virtio interfaces is always reported as Up. For this reason, a vSRX instance using virtio and chassis cluster cannot receive link up and link down messages from virtio interfaces.

The virtual network MAC aging time determines the amount of time that an entry remains in the MAC table. We recommend that you reduce the MAC aging time on the virtual networks to minimize the downtime during failover.

For example, you can use the **brctl setageing bridge 1** command to set aging to 1 second for the Linux bridge.

You configure the virtual networks for the control and fabric links, then create and connect the control interface to the control virtual network and the fabric interface to the fabric virtual network.

Creating the Chassis Cluster Virtual Networks with virt-manager

In KVM, you create two virtual networks (control and fabric) to which you can connect each vSRX instance for chassis clustering.

To create a virtual network with **virt-manager**:

1. Launch **virt-manager** and select **Edit>Connection Details**. The Connection details dialog box appears.
2. Select **Virtual Networks**. The list of existing virtual networks appears.
3. Click **+** to create a new virtual network for the control link. The Create a new virtual network wizard appears.
4. Set the subnet for this virtual network and click **Forward**.
5. Select **Enable DHCP** and click **Forward**.
6. Select **Isolated virtual network** and click **forward**.
7. Verify the settings and click **Finish** to create the virtual network.

Creating the Chassis Cluster Virtual Networks with virsh

In KVM, you create two virtual networks (control and fabric) to which you can connect each vSRX for chassis clustering.

To create the control network with **virsh**:

1. Use the **virsh net-define** command on the *host* OS to create an XML file that defines the new virtual network. Include the XML fields described in Table 19 on page 93 to define this network.

   NOTE: See the official **virsh** documentation for a complete description of available options.

Table 19: virsh net-define XML Fields

Field	Description
<network>...</network>	Use this XML wrapper element to define a virtual network.
<name>*net-name*</name>	Specify the virtual network name.
<bridge name="*bridge-name*" />	Specify the name of the host bridge used for this virtual network.
<forward mode="*forward-option*" />	Specify routed or nat. Do not use the <forward> element for isolated mode.
<ip address="*ip-address*" netmask="*net-mask*" <dhcp range start="*start*" end="*end*" </ dhcp> </ip>	Specify the IP address and subnet mask used by this virtual network, along with the DHCP address range.

The following example shows a sample XML file that defines a control virtual network.

```
<network>
 <name>control</name>
 <bridge name="controlvbr0" />
 <ip address="10.10.10.1" netmask="255.255.255.0" >
  <dhcp>
<range start="10.10.10.2" end="10.10.10.99" />
  </dhcp>
 </ip>
</network>
```

2. Use the **virsh net-start** command to start the new virtual network.

   hostOS# **virsh net-start control**

3. Use the **virsh net-autostart** command to automatically start the new virtual network when the host OS boots.

   hostOS# **virsh net-autostart control**

4. Optionally, use the **virsh net-list –all** command in the host OS to verify the new virtual network.

```
hostOS# # virsh net-list --all
 Name                 State      Autostart     Persistent
----------------------------------------------------------
 control              active     yes           yes
 default              active     yes           yes
```

5. Repeat this procedure to create the fabric virtual network.

Configuring the Control and Fabric Interfaces with virt-manager

To configure the control and fabric interfaces for chassis clustering with **virt-manager**:

1. In **virt-manager**, double-click the vSRX VM and select **View>Details**. The vSRX Virtual Machine details dialog box appears.
2. Select the second *vNIC* and select the control *virtual network* from the Source device list.
3. Select **virtio** from the Device model list and click **Apply**.
4. Select a subsequent vNIC, and select the fabric virtual network from the Source device list.
5. Select **virtio** from the Device model list and click **Apply**.
6. For the fabric interface, use the **ifconfig** command on the host OS to set the MTU to 9000.

   hostOS# **ifconfig vnet1 mtu 9000**

Configuring the Control and Fabric Interfaces with virsh

To configure control and fabric interfaces to a vSRX VM with **virsh**:

1. Type **virsh attach-interface --domain** ***vsrx-vm-name*** **--type network --source** ***control-vnetwork*** **--target control --model virtio** on the host OS.

   This command creates a virtual interface called control and connects it to the control virtual network.
2. Type **virsh attach-interface --domain** ***vsrx-vm-name*** **--type network --source** ***fabric-vnetwork*** **--target fabric --model virtio** on the host OS.

   This command creates a virtual interface called fabric and connects it to the fabric virtual network.
3. For the fabric interface, use the **ifconfig** command on the host OS to set the MTU to 9000.

   hostOS# **ifconfig vnet1 mtu 9000**

Configuring Chassis Cluster Fabric Ports

After the chassis cluster is formed, you must configure the interfaces that make up the fabric (data) ports.

Ensure that you have configured the following:

- Set the chassis cluster IDs on both vSRX instances and rebooted the vSRX instances.
- Configured the control and fabric links.

1. On the vSRX node 0 console in configuration mode, configure the fabric (data) ports of the cluster that are used to pass real-time objects (RTOs). The configuration will be synchronized directly through the control port to vSRX node 1.

   NOTE: A fabric port can be any unused revenue interface.

```
user@vsrx0# set interfaces fab0 fabric-options member-interfaces ge-0/0/0
user@vsrx0# set interfaces fab1 fabric-options member-interfaces ge-7/0/0
user@vsrx0# set chassis cluster reth-count 2
user@vsrx0# set chassis cluster redundancy-group 0 node 0 priority 100
user@vsrx0# set chassis cluster redundancy-group 0 node 1 priority 10
user@vsrx0# set chassis cluster redundancy-group 1 node 0 priority 100
user@vsrx0# set chassis cluster redundancy-group 1 node 1 priority 10
user@vsrx0# commit
```

2. Reboot vSRX node 0.

RELATED DOCUMENTATION

virt tools

Verify the Chassis Cluster Configuration

IN THIS SECTION

- Purpose | 96
- Action | 96

Purpose

Verify that the chassis cluster is operational after you set up the vSRX instances for clustering and set the cluster ID and the node ID.

Action

After reboot, the two nodes are reachable on interface fxp0 with SSH. If the configuration is operational, the **show chassis cluster status** command displays output similar to that shown in the following sample output.

vsrx> **show chassis cluster status**

```
Cluster ID: 1
Node                  Priority          Status    Preempt  Manual Monitor-
failures

Redundancy group: 0 , Failover count: 1
    node0                   100         secondary      no       no     None
    node1                   10          primary        no       no     None

Redundancy group: 1 , Failover count: 2
    node0                   100         secondary      no       no     None
    node1                   10          primary        no       no     None
```

A cluster is healthy when both the primary and the secondary nodes are present and when both have a priority greater than 0.

6
CHAPTER

Troubleshooting

Unresolved topicref | 99

Unresolved topicref

SUMMARY

Unresolved topicref placeholder.

This is a placeholder for unresolved topicref links.