JUNIPER NETWORKS | Engineering Simplicity

vSRX Deployment Guide for Contrail

Published
2020-12-28

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

vSRX Deployment Guide for Contrail

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

Use this guide to install the vSRX Virtual Firewall on Juniper Networks Contrail, and to create a service chain that includes vSRX. 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

vSRX Overview

SUMMARY

In this topic you learn about vSRX architecture and its benefits.

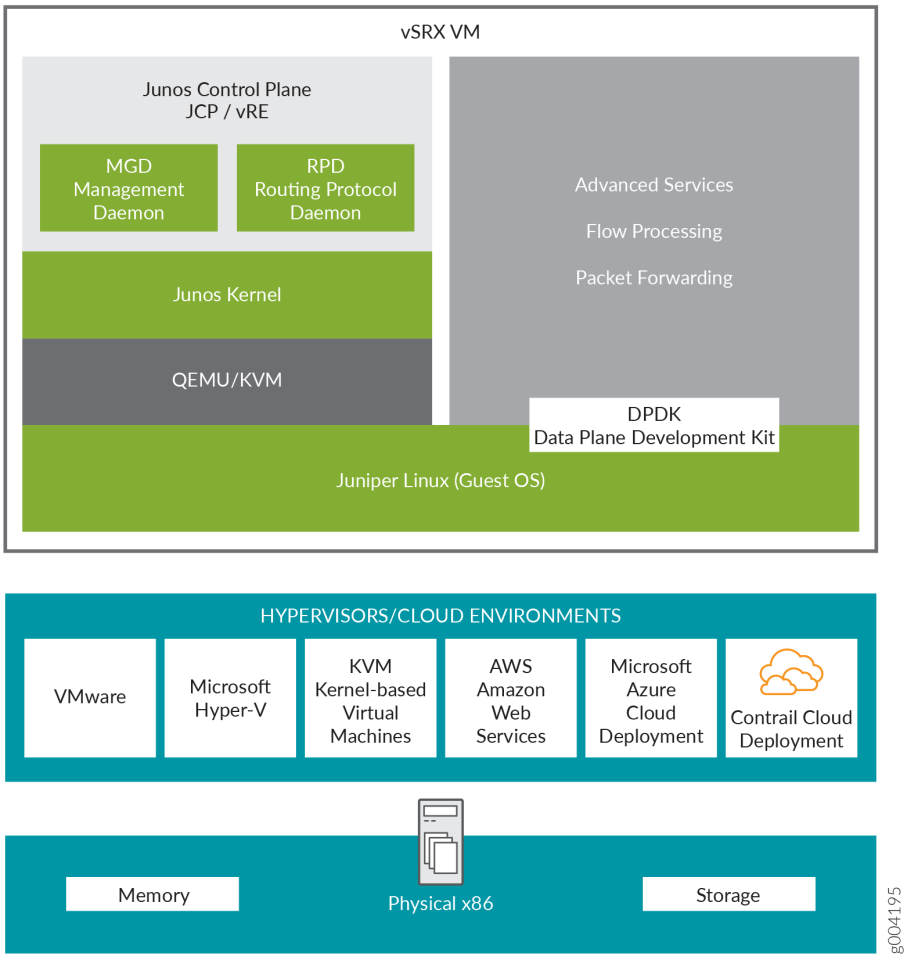
IN THIS SECTION

vSRX is a virtual security appliance that provides security and networking services at the perimeter or edge in virtualized private or public *cloud* environments. vSRX runs as a virtual machine (*VM*) on a standard x86 server. vSRX is built on the Junos operating system (Junos OS) and delivers networking and security features similar to those available on the software releases for the SRX Series Services Gateways.

The vSRX provides you with a complete Next-Generation Firewall (NGFW) solution, including core firewall, VPN, NAT, advanced Layer 4 through Layer 7 security services such as Application Security, intrusion detection and prevention (IPS), and UTM features including Enhanced Web Filtering and Anti-Virus. Combined with Sky ATP, the vSRX offers a cloud-based advanced anti-malware service with dynamic analysis to protect against sophisticated malware, and provides built-in machine learning to improve verdict efficacy and decrease time to remediation.

Figure 1 on page 3 shows the high-level architecture.

Figure 1: vSRX Architecture



vSRX includes the Junos control plane (JCP) and the packet forwarding engine (PFE) components that make up the data plane. vSRX uses one virtual CPU (vCPU) for the JCP and at least one vCPU for the PFE. Starting in Junos OS Release 15.1X49-D70 and Junos OS Release 17.3R1, multi-core vSRX supports scaling vCPUs and virtual RAM (vRAM). Additional vCPUs are applied to the data plane to increase performance.

Junos OS Release 18.4R1 supports a new software architecture vSRX 3.0 that removes dual OS and nested virtualization requirement of existing vSRX architecture.

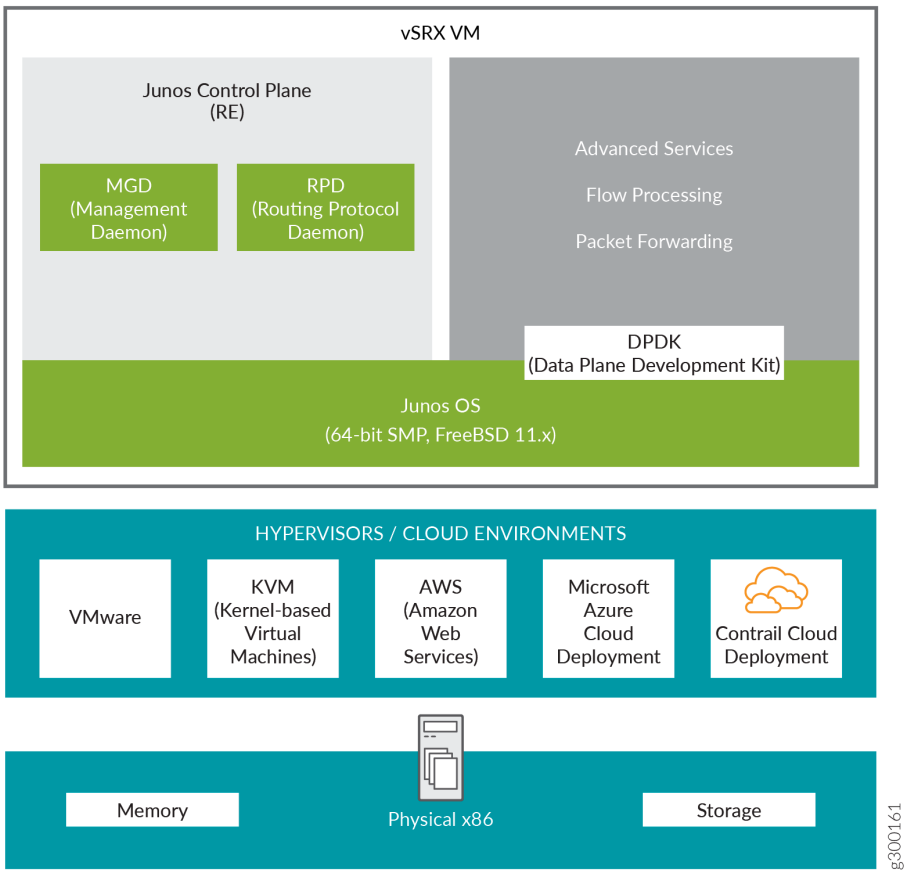
In vSRX 3.0 architecture, FreeBSD 11.x is used as the guest OS and the Routing Engine and Packet Forwarding Engine runs on FreeBSD 11.x as single virtual machine for improved performance and scalability. vSRX 3.0 uses DPDK to process the data packets in the data plane. A direct Junos upgrade from vSRX to vSRX 3.0 software is not supported.

vSRX 3.0 has the following enhancements compared to vSRX:

- Removed the restriction of requiring nested VM support in hypervisors.
- Removed the restriction of requiring ports connected to control plane to have Promiscuous mode enabled.
- Improved boot time and enhanced responsiveness of the control plane during management operations.
- Improved live migration.

Figure 2 on page 4 shows the high-level architecture for vSRX 3.0

Figure 2: vSRX 3.0 Architecture



Benefits

vSRX on standard x86 servers enables you to quickly introduce new services, deliver customized services to customers, and scale security services based on dynamic needs. vSRX is ideal for public, private, and hybrid cloud environments.

Some of the key benefits of vSRX in a virtualized private or public cloud multitenant environment include:

- *Stateful firewall* protection at the tenant edge
- Faster deployment of virtual firewalls into new sites
- Ability to run on top of various hypervisors and public cloud infrastructures
- Full routing, *VPN*, core security, and networking capabilities
- Application security features (including IPS and App-Secure)
- Content security features (including Anti Virus, Web Filtering, Anti Spam, and Content Filtering)
- Centralized management with Junos Space Security Director and local management with J-Web Interface
- Juniper Networks Sky Advanced Threat Prevention (Sky ATP) integration

Release History Table

Release	Description
15.1X49-D70	Starting in Junos OS Release 15.1X49-D70 and Junos OS Release 17.3R1, multi-core vSRX supports scaling vCPUs and virtual RAM (vRAM). Additional vCPUs are applied to the data plane to increase performance.

Understand vSRX with Contrail

IN THIS SECTION

This section presents an overview of vSRX on Contrail

vSRX on Juniper Networks Contrail

Juniper Networks Contrail is an open, standards-based software solution that delivers network *virtualization* and service automation for federated cloud networks. It provides self-service provisioning, improves network troubleshooting and diagnostics, and enables service chaining for dynamic application environments across enterprise virtual private cloud (VPC), managed Infrastructure as a Service (IaaS), and Networks Functions Virtualization (*NFV*) use cases.

You can use Contrail with open cloud orchestration systems such as OpenStack or CloudStack to instantiate instances of vSRX in a virtual environment. Contrail with vSRX provides network services such as *firewall*, *NAT*, and load balancing to virtual networks.

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

vSRX Scale Up Performance

Table 1 on page 6 shows the vSRX scale up performance 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 X520/X540)	Junos OS Release 15.1X49-D20
5 vCPUs	8 GB	• Virtio • SR-IOV (Intel X520/X540)	Junos OS Release 15.1X49-D70 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.

RELATED DOCUMENTATION

Contrail Overview

Requirements for vSRX on Contrail

IN THIS SECTION

Software Requirements

Table 2 on page 8 lists the system software requirement specifications when deploying vSRX on Juniper Networks Contrail. 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.

Table 2: Specifications for vSRX on Juniper Networks Contrail

Component	Specification	Junos OS Release Introduced
Hypervisor support	Linux KVM	Junos OS Release 15.1X49-D20 and Junos OS Release 17.3R1
Memory	4 GB	Junos OS Release 15.1X49-D20 and Junos OS Release 17.3R1
	8 GB	Junos OS Release 15.1X49-D70 and Junos OS Release 17.3R1
Disk space	20 GB IDE drive	Junos OS Release 15.1X49-D20 and Junos OS Release 17.3R1
vCPUs	2 vCPUs **NOTE**: The Contrail compute node must bare metal since vSRX as a VNF does not support nested virtualization.	Junos OS Release 15.1X49-D20 and Junos OS Release 17.3R1
	5 vCPUs **NOTE**: The Contrail compute node must bare metal since vSRX as a VNF does not support nested virtualization.	Junos OS Release 15.1X49-D70 and Junos OS Release 17.3R1
vNICs	Up to 16 vNICs • Virtio • SR-IOV **NOTE**: We recommend the Intel X520/X540 physical NICs for SR-IOV support on vSRX. For SR-IOV limitations, see the *Known Behavior* section of the *vSRX Release Notes*.	Junos OS Release 15.1X49-D20 and Junos OS Release 17.3R1

Table 3 on page 9 lists the software specifications on the vSRX.

Table 3: Software Specifications for vSRX 3.0 on Juniper Networks Contrail

Flavor Name	vCPU	Junos OS Release Introduced
Hypervisor support	Linux KVM	Junos OS Release 18.2R1 or later release
Memory	4 GB	Junos OS Release 18.2R1 or later release
	8 GB	Junos OS Release 18.2R1 or later release
Disk space	20 GB IDE drive	Junos OS Release 18.2R1 or later release
vCPUs	2 vCPUs	Junos OS Release 18.2R1 or later release
	5 vCPUs	Junos OS Release 18.2R1 or later release
vNICs	Up to 16 vNICs • Virtio • SR-IOV **NOTE**: We recommend the Intel X520 physical NICs for SR-IOV support on small flavor vSRX, Intel X710 for Medium flavor vSRX.	Junos OS Release 18.2R1 or later release

Contrail Recommendations for vSRX

Table 4 on page 10 lists the recommended software versions to run vSRX on Contrail.

Table 4: Contrail Recommendations for vSRX

Software	Version	Supported Release
Contrail	2.20	Junos OS Release 15.1X49-D20 and Junos OS Release 17.3R1 or later release
	3.1	Junos OS Release 15.1X49-D60 and Junos OS Release 17.3R1 or later release
	3.5	Junos OS Release 18.4R1
OpenStack	Juno or Icehouse	Junos OS Release 15.1X49-D20 and Junos OS Release 17.3R1 or later release
	Juno or Kilo	Junos OS Release 15.1X49-D60 and Junos OS Release 17.3R1 or later release
Host OS	Ubuntu 14.04.2	Junos OS Release 15.1X49-D20 and Junos OS Release 17.3R1 or later release
Linux Kernel	3.16	Junos OS Release 15.1X49-D20 and Junos OS Release 17.3R1 or later release

NOTE: We recommend that you enable hardware-based virtualization on the host machine. You can verify CPU compatibility here: http://www.linux-kvm.org/page/Processor_support. See Contrail - Server Requirements to review any additional requirements for Contrail.

Table 5 on page 11 lists the contrail recommendations for vSRX.

Table 5: Contrail Recommendations for vSRX 3.0

Software	Version	Supported Release
Contrail	3.1	Junos OS Release 18.2R1 or later release
	3.2	Junos OS Release 18.2R1 or later release
	5.X	Junos OS Release 19.3R1 or later release
OpenStack	Centos 7 or 8	Junos OS Release 18.2R1 or later release
Host OS	Ubuntu 14.04.2	Junos OS Release 18.2R1 or later release
Linux Kernel	Queens or later	Junos OS Release 18.2R1 or later release

Hardware Recommendations

Table 6 on page 11 lists the hardware specifications for the host machine that runs the vSRX VM.

Table 6: Hardware Specifications for the Host Machine

Component	Specification
Host memory size	4 GB (minimum) .
Host processor type	Intel x86_64 multicore CPU **NOTE**: DPDK requires Intel Virtualization VT-x/VT-d support in the CPU. See *About Intel Virtualization Technology*.

Table 6: Hardware Specifications for the Host Machine *(Continued)*

Component	Specification
Virtual network adapter	VMXNet3 device or VMWare Virtual NIC **NOTE**: Virtual Machine Communication Interface (VMCI) communication channel is internal to the ESXi hypervisor and the vSRX VM.

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 also 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 might become unresponsive if the NUMA nodes topology properties in OpenStack includes the line **hw:numa_nodes=2** to spread the instance's vCPUs across multiple host NUMA nodes. We recommend that you remove the **hw:numa_nodes=2** line from OpenStack to ensure that the PFE functions properly.

PCI NIC-to-VM Mapping

If the node on which vSRX is running is different from the node to which the Intel PCI NIC is connected, then packets will have to traverse an additional hop in the QPI link, and this will reduce overall throughput. On a Linux host OS, install the **hwloc** package and use the **lstopo** command to view information about relative physical NIC locations. On some servers where this information is not available, refer to the hardware documentation for the slot-to-NUMA node topology.

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
----------------------------------------------------
 25    instance-00000060              running
 31    instance-0000005b              running
 34    instance-000000bd              running
 35    instance-000000bc              running
```

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

```
hostOS# virsh domiflist 31
```

```
Interface  Type       Source     Model       MAC
-------------------------------------------------------
tapd3d9639c-d5 ethernet   -          virtio      02:d3:d9:63:9c:d5
tapc3c3751a-37 ethernet   -          virtio      02:c3:c3:75:1a:37
tap8af29333-1b ethernet   -          virtio      02:8a:f2:93:33:1b
tapf0387bee-9b ethernet   -          virtio      02:f0:38:7b:ee:9b
tap04e4b59a-91 ethernet   -          virtio      02:04:e4:b5:9a:91
```

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

Interface Mapping for vSRX on Contrail

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 7 on page 14 and Table 8 on page 15.

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 7 on page 14 shows the interface names and mappings for a standalone vSRX VM.

Table 7: 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 7: 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 8 on page 15 shows the interface names and mappings for a pair of vSRX VMs in a cluster (node 0 and node 1).

Table 8: 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 8: 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 Contrail

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 9 on page 16 lists the factory default settings for the vSRX security policies.

Table 9: Factory Default Settings for Security Policies

Source Zone	Destination Zone	Policy Action
trust	untrust	permit

Table 9: 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
- PCI Devices
- DPDK Release Notes

Junos OS Features Supported on vSRX

SUMMARY

This topic provides details of the Junos OS features supported and not supported on vSRX.

IN THIS SECTION

SRX Series Features Supported on vSRX

vSRX inherits most of the branch SRX Series features with the following considerations shown in Table 10 on page 18.

To determine the Junos OS features supported on vSRX, use the Juniper Networks Feature Explorer, a Web-based application that helps you to explore and compare Junos OS feature information to find the right software release and hardware platform for your network. Find Feature Explorer at: Feature Explorer: vSRX .

Table 10: vSRX Feature Considerations

Feature	Description	
IDP	The IDP feature is subscription based and must be purchased. After purchase, you can activate the IDP feature with the license key. For SRX Series IDP configuration details, see: Understanding Intrusion Detection and Prevention for SRX Series	
IPSec VPNs	Starting in Junos OS Release 19.3R1, vSRX supports the following authentication algorithms and encryption algorithms: • Authentication algorithm: hmac-sha1-96 and HMAC-SHA-256-128 authentication • Encryption algorithm: aes-128-cbc Starting in Junos OS Release 20.3R1, vSRX supports 10,000 IPsec VPN tunnels. To support the increased number of IPsec VPN tunnels, a minimum of 19 vCPUs are required. Out of the 19 vCPUs, 3 vCPUs must be dedicated to RE. You must run the **request system software add optional://junos-ike.tgz** command the first time you wish to enable increased IPsec tunnel capacity. For subsequent software upgrades of the instance, the junos-ike package is upgraded automatically from the new Junos OS releases installed in the instance. If chassis cluster is enabled then run this command on both the nodes. You can configure the number of vCPUs allocated to Junos Routing Engine using the **set security forwarding-options resource-manager cpu re <value>**. **NOTE**: 64 G memory is required to support 10000 tunnels in PMI mode. [See show security ipsec security-associations, show security ike tunnel-map, and show security ipsec tunnel-distribution.]	
IPsec VPN - Tunnel Scaling on vSRX	Types of Tunnels	Number of tunnels supported
	Site-Site VPN tunnels	2000

Table 10: vSRX Feature Considerations *(Continued)*

Feature	Description	
	AutoVPN tunnels	10,000
	IKE SA (Site-to-site)	2000
	IKE SA (AutoVPN)	10,000
	IKE SA (Site-to-site + AutoVPN)	10,000
	IPSec SA pairs (Site-to-site)	10,000 With 2000 IKE SAs, we can have 10,000 IPSec SA.
	IPSec SA pairs (AutoVPN)	10,000
	Site-to-site + AutoVPN IPSec SA pairs	2000 Site-to-site 8000 AutoVPN
	Site-to-site + AutoVPN tunnels	2000 Site-to-site 8000 AutoVPN
ISSU	ISSU is not supported.	
Logical Systems	Starting in Junos OS Release 20.1R1, you can configure logical systems and tenant systems on vSRX and vSRX 3.0 instances. With Junos OS, you can partition a single security device into multiple logical devices that can perform independent tasks. Each logical system has its own discrete administrative domain, logical interfaces, routing instances, security firewall and other security features. See Logical Systems Overview.	

Table 10: vSRX Feature Considerations *(Continued)*

Feature	Description
PowerMode IPsec	Starting in Junos OS Release 20.1R1, vSRX 3.0 instances support PowerMode IPsec that provides IPsec performance improvements using Vector Packet Processing (VPP) and Intel AES-NI instructions. PowerMode IPsec is a small software block inside the SRX PFE (SRX Packet Forwarding Engine) that is activated when PowerMode is enabled. Supported Features in PowerMode IPsec • IPsec functionality • Traffic selectors • Secure tunnel interface (st0) • All control plane IKE functionality • Auto VPN with traffic selector • Auto VPN with routing protocol • IPv6 • Stateful Layer 4 firewall • High-Availability • NAT-T Non-Supported Features in PowerMode IPsec • NAT • IPsec in IPsec • GTP/SCTP firewall • Application firewall/AppSecure • QoS • Nested tunnel • Screen

Table 10: vSRX Feature Considerations *(Continued)*

Feature	Description
	• Multicast • Host traffic
Tenant Systems	Starting in Junos OS Release 20.1R1, you can configure tenant systems on vSRX and vSRX 3.0 instances. A tenant system provides logical partitioning of the SRX device into multiple domains similar to logical systems and provides high scalability. See Tenant Systems Overview.
Transparent mode	The known behaviors for transparent mode support on vSRX are: • The default MAC learning table size is restricted to 16,383 entries. For information about configuring transparent mode for vSRX, see Layer 2 Bridging and Transparent Mode Overview.

Table 10: vSRX Feature Considerations *(Continued)*

Feature	Description
UTM	• The UTM feature is subscription based and must be purchased. After purchase, you can activate the UTM feature with the license key. • Starting in Junos OS Release 19.4R1, vSRX 3.0 instances support the Avira scan engine, which is an on-device antivirus scanning engine. See On-Device Antivirus Scan Engine. • For SRX Series UTM configuration details, see Unified Threat Management Overview. • For SRX Series UTM antispam configuration details, see Antispam Filtering Overview. • **Advanced resource management (vSRX 3.0)**—Starting in Junos OS Release 19.4R1, vSRX 3.0 manages the additional system resource requirements for UTM-and IDP-specific services by reallocating CPU cores and extra memory. These values for memory and CPU cores are not user configured. Previously, system resources such as memory and CPU cores were fixed. You can view the allocated CPU and memory for advance security services on vSRX 3.0 instance by using the **show security forward-options resource-manager settings** command. To view the flow session scaling, use the **show security monitoring** command. [See show security monitoring and show security forward-options resource-manager settings.]

Some Junos OS software features require a license to activate the feature. To understand more about vSRX Licenses, see, Licenses for vSRX. Please refer to the Licensing Guide for general information about License Management. Please refer to the product Data Sheets for further details, or contact your Juniper Account Team or Juniper Partner.

SRX Series Features Not Supported on vSRX

vSRX inherits many features from the SRX Series device product line. Table 11 on page 23 lists SRX Series features that are not applicable in a virtualized environment, that are not currently supported, or that have qualified support on vSRX.

Table 11: SRX Series Features Not Supported on vSRX

SRX Series Feature	vSRX Notes
Application Layer Gateways	
Avaya H.323	Not supported
Authentication with IC Series devices	
Layer 2 enforcement in UAC deployments	Not supported **NOTE**: UAC-IDP and UAC-UTM also are not supported.
Chassis cluster support **NOTE**: Support for chassis clustering to provide network node redundancy is only available on a vSRX deployment in Contrail, VMware, KVM, and Windows Hyper-V Server 2016.	
Chassis cluster for VirtIO driver	Only supported with KVM **NOTE**: The link status of VirtIO interfaces is always reported as UP, so a vSRX chassis cluster cannot receive link up and link down messages from VirtIO interfaces.
Dual control links	Not supported
In-band and low-impact cluster upgrades	Not supported
LAG and LACP (Layer 2 and Layer 3)	Not supported
Layer 2 Ethernet switching	Not supported
Low-latency firewall	Not supported
Class of service	

Table 11: SRX Series Features Not Supported on vSRX *(Continued)*

SRX Series Feature	vSRX Notes
High-priority queue on SPC	Not supported
Tunnels	Only GRE and IP-IP tunnels supported **NOTE**: A vSRX VM deployed on Microsoft Azure Cloud does not support GRE and multicast.
Data plane security log messages (stream mode)	
TLS protocol	Not supported
Diagnostic tools	
Flow monitoring cflowd version 9	Not supported
Ping Ethernet (CFM)	Not supported
Traceroute Ethernet (CFM)	Not supported
DNS proxy	
Dynamic DNS	Not supported
Ethernet link aggregation	
LACP in standalone or chassis cluster mode	Not supported
Layer 3 LAG on routed ports	Not supported
Static LAG in standalone or chassis cluster mode	Not supported

Table 11: SRX Series Features Not Supported on vSRX *(Continued)*

SRX Series Feature	vSRX Notes
Ethernet link fault management	
Physical interface (encapsulations) • **ethernet-ccc** • **ethernet-tcc** • **extended-vlan-ccc** • **extended-vlan-tcc**	Not supported
Interface family • **ccc, tcc** • **ethernet-switching**	Not supported
Flow-based and packet-based processing	
End-to-end packet debugging	Not supported
Network processor bundling	
Services offloading	
Interfaces	
Aggregated Ethernet interface	Not supported
IEEE 802.1X dynamic VLAN assignment	Not supported
IEEE 802.1X MAC bypass	Not supported

Table 11: SRX Series Features Not Supported on vSRX *(Continued)*

SRX Series Feature	vSRX Notes
IEEE 802.1X port-based authentication control with multisupplicant support	Not supported
Interleaving using MLFR	Not supported
PoE	Not supported
PPP interface	Not supported
PPPoE-based radio-to-router protocol	Not supported
PPPoE interface **NOTE**: Starting in Junos OS Release 15.1X49-D100 and Junos OS Release 17.4R1, the vSRX supports Point-to-Point Protocol over Ethernet (PPPoE) interface.	Not supported
Promiscuous mode on interfaces	Only supported if enabled on the hypervisor
IPSec and VPNs	
Acadia - Clientless VPN	Not supported
DVPN	Not supported
Hardware IPsec (bulk crypto) Cavium/RMI	Not supported
IPsec tunnel termination in routing instances	Supported on virtual router only
Multicast for AutoVPN	Not supported

Table 11: SRX Series Features Not Supported on vSRX *(Continued)*

SRX Series Feature	vSRX Notes
IPv6 support	
DS-Lite concentrator (also called Address Family Transition Router [AFTR])	Not supported
DS-Lite initiator (aka B4)	Not supported
J-Web	
Enhanced routing configuration	Not supported
New Setup wizard (for new configurations)	Not supported
PPPoE wizard	Not supported
Remote VPN wizard	Not supported
Rescue link on dashboard	Not supported
UTM configuration for Kaspersky antivirus and the default Web filtering profile	Not supported
Log file formats for system (control plane) logs	
Binary format (binary)	Not supported
WELF	Not supported
Miscellaneous	

Table 11: SRX Series Features Not Supported on vSRX *(Continued)*

SRX Series Feature	vSRX Notes
GPRS **NOTE**: Starting in Junos OS Release 15.1X49-D70 and Junos OS Release 17.3R1, vSRX supports GPRS.	Not supported
Hardware acceleration	Not supported
Logical systems	Not supported
Outbound SSH	Not supported
Remote instance access	Not supported
USB modem	Not supported
Wireless LAN	Not supported
MPLS	
Crcuit cross-connect (CCC) and translational cross-connect (TCC)	Not supported
Layer 2 VPNs for Ethernet connections	Only if promiscuous mode is enabled on the hypervisor
Network Address Translation	
Maximize persistent NAT bindings	Not supported
Packet capture	

Table 11: SRX Series Features Not Supported on vSRX *(Continued)*

SRX Series Feature	vSRX Notes
Packet capture	Only supported on physical interfaces and tunnel interfaces, such as *gr*, *ip*, and *st0*. Packet capture is not supported on redundant Ethernet interfaces (*reth*).
Routing	
BGP extensions for IPv6	Not supported
BGP Flowspec	Not supported
BGP route reflector	Not supported
CRTP	Not supported
Switching	
Layer 3 Q-in-Q VLAN tagging	Not supported
Transparent mode	
UTM	Not supported
Unified threat management	
Express AV	Not supported
Kaspersky AV	Not supported
Upgrading and rebooting	

Table 11: SRX Series Features Not Supported on vSRX *(Continued)*

SRX Series Feature	vSRX Notes
Autorecovery	Not supported
Boot instance configuration	Not supported
Boot instance recovery	Not supported
Dual-root partitioning	Not supported
OS rollback	Not supported
User interfaces	
NSM	Not supported
SRC application	Not supported
Junos Space Virtual Director	Only supported with VMware

2

CHAPTER

Installing vSRX in Contrail

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
   ```

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
```

Create an Image Flavor with OpenStack

IN THIS SECTION

Before you begin, ensure that you have a working OpenStack installation. See the OpenStack Installation Guide for more details.

OpenStack launches instances of images, based on the image installed and VM templates called *flavors*. Flavors set the memory, vCPU, and storage requirements for the vSRX image. You can use the Horizon

GUI or the OpenStack **nova** commands to create flavors for the vSRX VMs. See "Requirements for vSRX on Contrail" on page 7 for the software requirement specifications for a vSRX VM.

> **CAUTION**: The packet forwarding engine (PFE) on the vSRX might become unresponsive if the NUMA nodes topology properties in OpenStack includes the line **hw:numa_nodes=2** to spread the instance's vCPUs across multiple host NUMA nodes. We recommend that you remove the **hw:numa_nodes=2** line from OpenStack to ensure that the PFE functions properly.

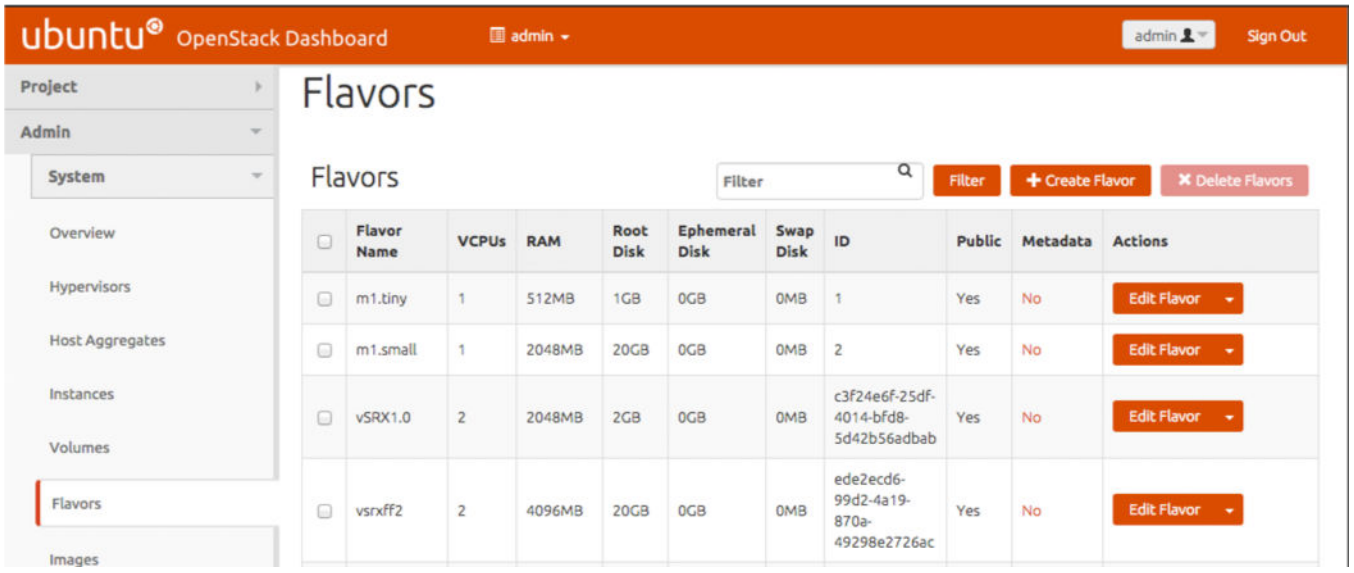
Create an Image Flavor for vSRX with Horizon

OpenStack uses VM templates, or flavors, to set the memory, vCPU, and storage requirements for an image. OpenStack includes a default set of flavors, but we recommend that you create a flavor to match the vSRX image requirements.

To create an image flavor for vSRX with Horizon:

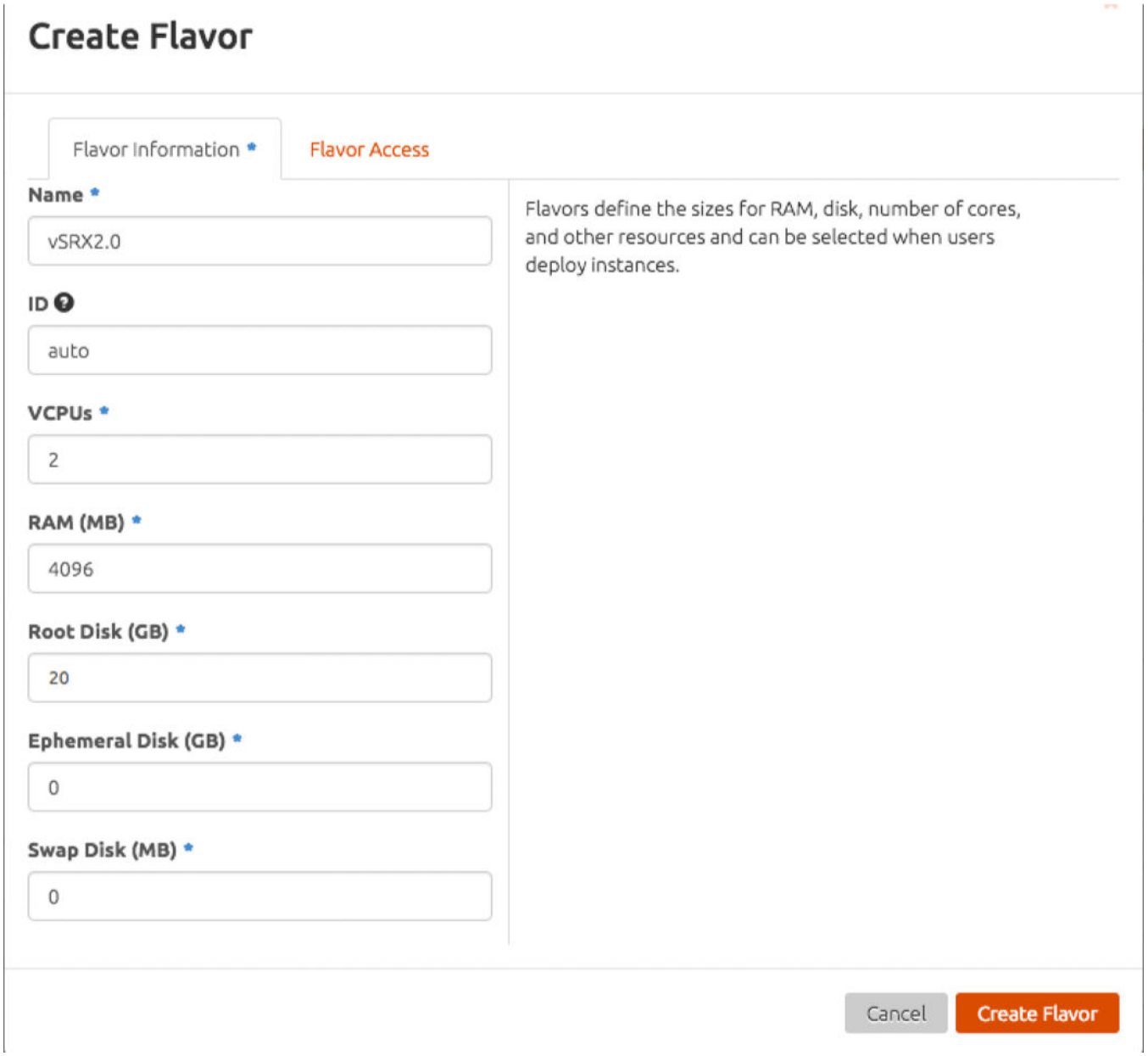
1. From the Horizon GUI, select your project, and select **Admin>System Panel>Flavors**. The list of existing image flavors appears, as shown in Figure 3 on page 34.

Figure 3: OpenStack Flavors



2. Click **Create Flavor**. The Create Flavor dialog box appears, as shown in Figure 4 on page 35.

Figure 4: Create a Flavor



3. Enter a name in the Name box for this vSRX flavor.
4. Enter the appropriate value in the vCPUs box for your configuration. The minimum required for vSRX is 2 vCPUs.
5. Enter the appropriate value in the RAM MB box. The minimum required for vSRX is 4096 MB.
6. Enter the appropriate value in the Root Disk GB box. The minimum required for vSRX is 20 GB.

7. Enter the appropriate values in the Emphemeral Disk GB and Swap Disk MB boxes. The minimum required for vSRX is 0 for each.
8. Click **Create Flavor**. The flavor appears on the Flavors tab.

Create an Image Flavor for vSRX with the Nova CLI

To create an image flavor for vSRX with the **nova** CLI command:

1. Use the **nova flavor-create** command on the OpenStack compute node that will host the vSRX VM. See Table 12 on page 36 for a list of mandatory parameters.

   NOTE: See the official **OpenStack** documentation for a complete description of available options for the **nova flavor-create** command.

   Table 12: nova flavor-create Command

Command Option	Description
--is-public true	Set the flavor as publicly available.
flavor_name	Name the vSRX flavor.
auto	Select auto to automatically assign the flavor ID.
ram_megabytes	Allocate RAM for the VM, in megabytes.
disk_gigabytes	Specify disk storage size for the VM.
vcpus	Allocate the number of vCPUs for the vSRX VM.

   NOTE: Use **nova help flavor-create** for more details on the command options.

2. Optionally, use the **nova flavor-list** to verify the flavors.

The following example creates a vSRX flavor with 4096 MB RAM, 2 vCPUs, and disk storage up to 20 GB:

```
$ nova flavor-create  --is-public true vsrx_flavor auto 4096 20 2
```

RELATED DOCUMENTATION

- OpenStack Installation Guide
- OpenStack End User Guide

Upload the vSRX Image

IN THIS SECTION

Contrail integrates with OpenStack for public, private, or hybrid cloud orchestration. You can install the vSRX image and use this installed image to provide security services in a service chain with Contrail.

Before installing vSRX, ensure that you have installed either Contrail and, optionally, OpenStack Glance.

- Contrail - Installation Overview
- OpenStack - Add the Image Service (glance)

You can upload the vSRX image with either Horizon, the OpenStack GUI dashboard, or Glance, the OpenStack CLI-based image services project.

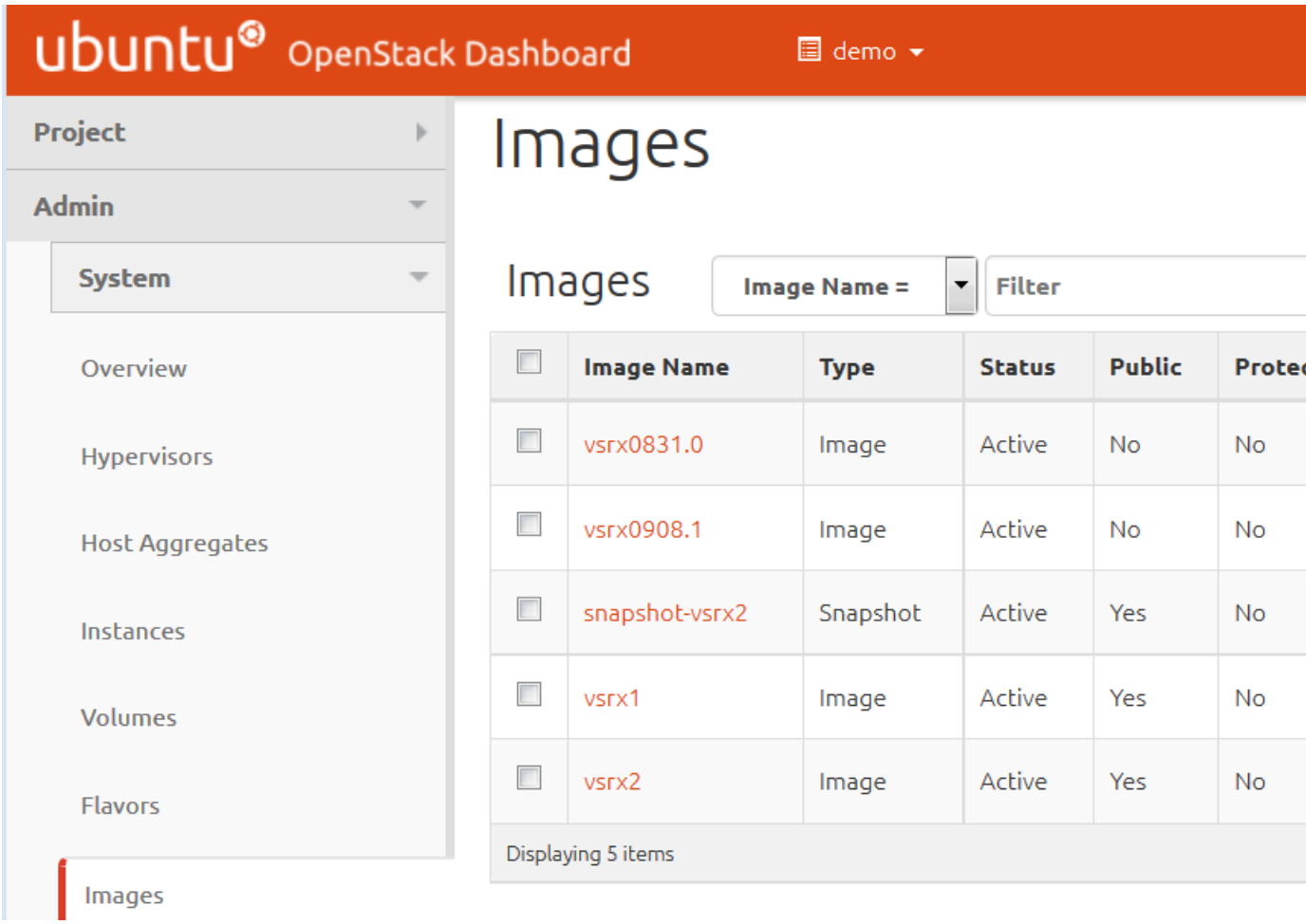
NOTE: To upgrade an existing vSRX instance, see *Migration, Upgrade, and Downgrade* in the *vSRX Release Notes*.

Upload the vSRX Image with OpenStack Horizon

To upload a vSRX image with Horizon:

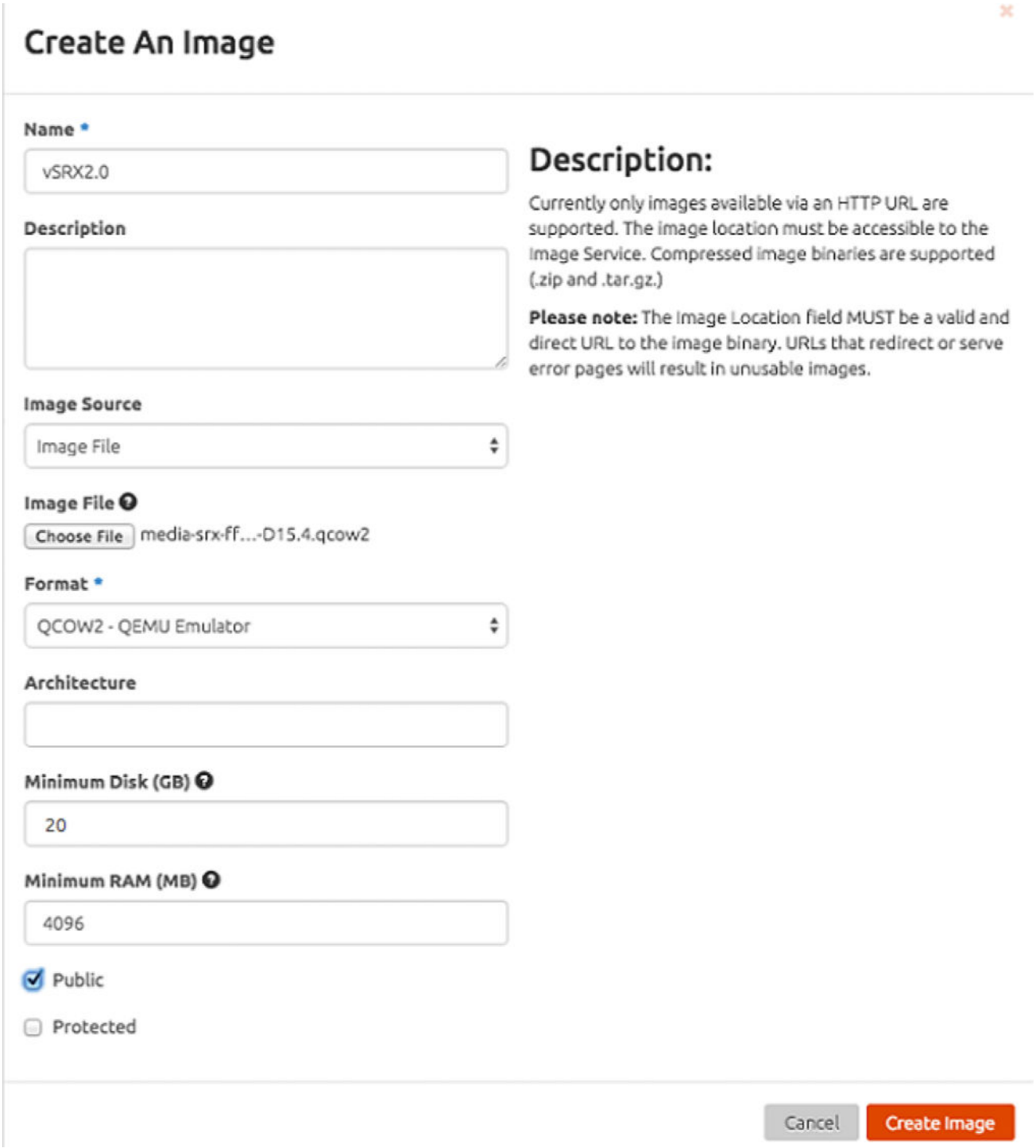
1. From the Horizon GUI, select your project, and select **Compute>Images**. The list of existing images appears, as shown in Figure 5 on page 38.

Figure 5: OpenStack Images



2. Click **Create Image**. The Create Image dialog box appears, as shown in Figure 6 on page 39.

Figure 6: Create an Image



3. Enter a name for the vSRX image, and enter the image location.

4. Select **QCOW2- QEMU Emulator** from the Format list.
5. Enter the appropriate value in the Minimum Disk (GB) box for your configuration. The minimum required for vSRX is 20 GB.
6. Enter the appropriate value in the Minimum RAM (MB) box. The minimium required for vSRX is 4096 MB.
7. Select **Public**.
8. Click **Create Image**. OpenStack uploads the image to the image service. The image appears on the Images tab.

NOTE: The default vSRX VM login ID is root with no password. By default, vSRX is assigned a DHCP-based IP address if a DHCP server is available on the network.

Upload the vSRX Image with the OpenStack Glance CLI

To upload a vSRX image with the Glance CLI:

1. Log in to the appropriate OpenStack compute node.
2. Use **wget** to download the vSRX image to the compute node.
3. Use **glance image-create** to add the image to the image service with a base configuration for disk, format, and memory requirements. Use **glance help image-create** for complete details on this command-line tool.

   For example, the following command adds the vSRX QCOW2 image to the image service with 20 GB disk space and 4096 MB of RAM:

   glance image-create --name='vSRXimage' --is-public=true --container-format=bare --disk-format=qcow2 --min-disk=20 --min-ram=4096 --file=media-srx-ffp-vsrx-vmdisk-15.1X49-D120.qcow2

   NOTE: vSRX requires at least 20 GB of disk space and 4096 MB of RAM.

NOTE: The default vSRX VM login ID is root with no password. By default, vSRX is assigned a DHCP-based IP address if a DHCP server is available on the network.

RELATED DOCUMENTATION

- Contrail - Installation Overview
- OpenStack Installation Guide for Ubuntu 14.04
- OpenStack - Add the Image Service (glance)
- OpenStack - Upload and Manage Images
- OpenStack - Manage images (glance)
- Migration, Upgrade, and Downgrade

Use Cloud-Init in an OpenStack Environment to Automate the Initialization of vSRX Instances

IN THIS SECTION

Starting in Junos OS Release 15.1X49-D100 and Junos OS Release 17.4R1, the cloud-init package (version 0.7x) comes pre-installed in the vSRX image to help simplify configuring new vSRX instances operating in an OpenStack environment according to a specified user-data file. Cloud-init is performed during the first-time boot of a vSRX instance.

Cloud-init is an OpenStack software package for automating the initialization of a cloud instance at boot-up. It is available in Ubuntu and most major Linux and FreeBSD operating systems. Cloud-init is designed to support multiple different cloud providers so that the same virtual machine (VM) image can be directly used in multiple hypervisors and cloud instances without any modification. Cloud-init support in a VM instance runs at boot time (first-time boot) and initializes the VM instance according to the specified user-data file.

A user-data file is a special key in the metadata service that contains a file that cloud-aware applications in the VM instance can access upon a first-time boot. In this case, it is the validated Junos OS configuration file that you intend to upload to a vSRX instance as the active configuration. 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.

When you create a vSRX instance, you can use cloud-init with a validated Junos OS configuration file (**juniper.conf**) to automate the initialization of new vSRX instances. The user-data file uses the standard Junos OS syntax to define all the configuration details for your vSRX instance. The default Junos OS configuration is replaced during the vSRX instance launch with a validated Junos OS configuration that you supply in the form of a user-data file.

NOTE: If using a release *earlier* than Junos OS Release 15.1X49-D130 and Junos OS Release 18.4R1, the user-data configuration file cannot exceed 16 KB. If your user-data file exceeds this limit, you must compress the file using gzip and use the compressed file. For example, the gzip junos.conf command results in the junos.conf.gz file.

Starting in Junos OS Release 15.1X49-D130 and Junos OS Release 18.4R1, if using a configuration drive data source in an OpenStack environment, the user-data configuration file size can be up to 64 MB.

The configuration must be validated and include details for the fxp0 interface, login, and authentication. It must also have a default route for traffic on fxp0. If any of this information is missing or incorrect, the instance is inaccessible and you must launch a new one.

WARNING: Ensure that the user-data configuration file is not configured to perform autoinstallation on interfaces using Dynamic Host Configuration Protocol (DHCP) to assign an IP address to the vSRX. Autoinstallation with DHCP will result in a "commit fail" for the user-data configuration file.

Starting in Junos OS Release 15.1X49-D130 and Junos OS Release 18.4R1, the cloud-init functionality in vSRX has been extended to support the use of a configuration drive data source in an OpenStack environment. The configuration drive uses the user-data attribute to pass a validated Junos OS configuration file to the vSRX instance. The user-data can be plain text or MIME file type text/plain. The configuration drive is typically used in conjunction with the Compute service, and is present to the instance as a disk partition labeled **config-2**. The configuration drive has a maximum size of 64 MB, and must be formatted with either the vfat or ISO 9660 filesystem.

The configuration drive data source also provides the flexibility to add more than one file that can be used for configuration. A typical use case would be to add a Day0 configuration file and a license file. In this case, there are two methods that can be employed to use a configuration drive data source with a vSRX instance:

- User-data (Junos OS Configuration File) alone—This approach uses the user-data attribute to pass the Junos OS configuration file to each vSRX instance. The user-data can be plain text or MIME file type text/plain.
- Junos OS configuration file and license file—This approach uses the configuration drive data source to send the Junos OS configuration and license file(s) to each vSRX instance.

> **NOTE**: If a license file is to be configured in vSRX, it is recommended to use the **–file** option rather than the **user-data** option to provide the flexibility to configure files larger than the 16 KB limit of user-data.

To use a configuration drive data source to send Junos OS configuration and license file(s) to a vSRX instance, the files needs to be sent in a specific folder structure. In this application, the folder structure of the configuration drive data source in vSRX is as follows:

```
-   OpenStack
        -   latest
              -   junos-config
                      -  configuration.txt
              -   junos-license
                      -  License_file_name.lic
                      -  License_file_name.lic
```

//OpenStack//latest/junos-config/configuration.txt

//OpenStack//latest/junos-license/license.lic

Before you begin:

- Create a configuration file with the Junos OS command syntax and save it. The configuration file can be plain text or MIME file type text/plain. The string **#junos-config** must be the first line of the user-data configuration file before the Junos OS configuration.

  > **NOTE**: The **#junos-config** string is mandatory in the user-data configuration file; if it is not included, the configuration will not be applied to the vSRX instance as the active configuration.

- Determine the name for the vSRX instance you want to initialize with a validated Junos OS configuration file.
- Determine the flavor for your vSRX instance, which defines the compute, memory, and storage capacity of the vSRX instance.
- Starting in Junos OS Release 15.1X49-D130 and Junos OS Release 18.4R1, if using a configuration drive, ensure the following criteria is met to enable cloud-init support for a configuration drive in OpenStack:
  - The configuration drive must be formatted with either the **vfat** or **iso9660** filesystem.

> **NOTE**: The default format of a configuration drive is an ISO 9660 file system. To explicitly specify the ISO 9660/vfat format, add the **config_drive_format=iso9660/vfat** line to the **nova.conf** file.

- The configuration drive must have a filesystem label of **config-2**.
- The folder size must be no greater than 64 MB.

Depending on your OpenStack environment, you can use either an OpenStack command-line interface (such as **nova boot** or **openstack server create**) or the OpenStack Dashboard ("Horizon") to launch and initialize a vSRX instance.

Perform Automatic Setup of a vSRX Instance Using an OpenStack Command-Line Interface

You can launch and manage a vSRX instance using either the **nova boot** or **openstack server create** commands, which includes the use of a validated Junos OS configuration user-data file from your local directory to initialize the active configuration of the target vSRX instance.

To initiate the automatic setup of a vSRX instance from an OpenStack command-line client:

1. If you have not done so already, create a configuration file with the Junos OS command syntax and save the file. The configuration file can be plain text or MIME file type text/plain.

   The user-data configuration file must contain the full vSRX configuration that is to be used as the active configuration on each vSRX instance, and the string **#junos-config** must be the first line of the user-data configuration file before the Junos OS configuration.

   > **NOTE**: The **#junos-config** string is mandatory in the user-data configuration file; if it is not included, the configuration will not be applied to the vSRX instance as the active configuration.

2. Copy the Junos OS configuration file to an accessible location from where it can be retrieved to launch the vSRX instance.
3. Depending on your OpenStack environment, use the **nova boot** or **openstack server create** command to launch the vSRX instance with a validated Junos OS configuration file as the specified user-data.

NOTE: You can also use the **nova boot** equivalent in an Orchestration service such as HEAT.

For example:

- **nova boot -user-data </path/to/vsrx_configuration.txt> --image vSRX_image --flavor vSRX_flavor_instance**
- **openstack server create -user-data </path/to/vsrx_configuration.txt> --image vSRX_image --flavor vSRX_flavor_instance**

  Where:

  -user-data </path/to/vsrx_configuration.txt> specifies the location of the Junos OS configuration file. The user-data configuration file size is limited to approximately 16,384 bytes.

  --image vSRX_image identifies the name of a unique vSRX image.

  --flavor vSRX_flavor_instance identifies the vSRX flavor (ID or name).

Starting in Junos OS Release 15.1X49-D130 and Junos OS Release 18.4R1, to enable the use of a configuration drive for a specific request in the OpenStack compute environment, include the **-config-drive true** parameter in the **nova boot** or **openstack server create** command.

NOTE: It is possible to enable the configuration drive automatically on all instances by configuring the OpenStack Compute service to always create a configuration drive. To do this, specify the **force_config_drive=True** option in the **nova.conf** file.

For example, to use the user-data attribute to pass the Junos OS configuration to each vSRX instance:

nova boot -config-drive true -flavor vSRX_flavor_instance -image vSRX_image -user-data </path/to/vsrx_configuration.txt>

Where:

-user-data </path/to/vsrx_configuration.txt> specifies the location of the Junos OS configuration file. The user-data configuration file size is limited to approximately 64 MB.

-image vSRX_image identifies the name of a unique vSRX image.

-flavor vSRX_flavor_instance identifies the vSRX flavor (ID or name).

For example, to specify the configuration drive with multiple files (Junos OS configuration file and license file):

nova boot -config-drive true -flavor vSRX_flavor_instance -image vSRX_image [-file /junos-config/configuration.txt=/path/to/file] [-file /junos-license/license.lic=path/to/license]

Where:

[-file /junos-config/configuration.txt=/path/to/file] specifies the location of the Junos OS configuration file.

[-file /junos-license/license.lic=path/to/license] specifies the location of the Junos OS configuration file.

-image vSRX_image identifies the name of a unique vSRX image.

-flavor vSRX_flavor_instance identifies the vSRX flavor (ID or name).

4. Boot or reboot the vSRX instance. During the initial boot-up sequence, the vSRX instance processes the cloud-init request.

   NOTE: The boot time for the vSRX instance might increase with the use of the cloud-init package. This additional time in the initial boot sequence is due to the operations performed by the cloud-init package. During this operation, the cloud-init package halts the boot sequence and performs a lookup for the configuration data in each data source identified in the cloud.cfg. The time required to look up and populate the cloud data is directly proportional to the number of data sources defined. In the absence of a data source, the lookup process continues until it reaches a predefined timeout of 30 seconds for each data source.

5. When the initial boot-up sequence resumes, the user-data file replaces the original factory-default Junos OS configuration loaded on the vSRX instance. If the commit succeeds, the factory-default configuration will be permanently replaced. If the configuration is not supported or cannot be applied to the vSRX instance, the vSRX will boot using the default Junos OS configuration.

SEE ALSO

- Cloud-Init Documentation
- OpenStack command-line clients
- Compute service (nova) command-line client
- Openstack Server Create
- Enabling the configuration drive (configdrive)
- Instances

Perform Automatic Setup of a vSRX Instance from the OpenStack Dashboard (Horizon)

Horizon is the canonical implementation of the OpenStack Dashboard. It provides a Web-based user interface to OpenStack services including Nova, Swift, Keystone, and so on. You can launch and manage a vSRX instance from the OpenStack Dashboard, which includes the use of a validated Junos OS configuration user-data file from your local directory to initialize the active configuration of the target vSRX instance.

To initiate the automatic setup of a vSRX instance from the OpenStack Dashboard:

1. If you have not done so already, create a configuration file with the Junos OS command syntax and save the file. The configuration file can be plain text or MIME file type text/plain.

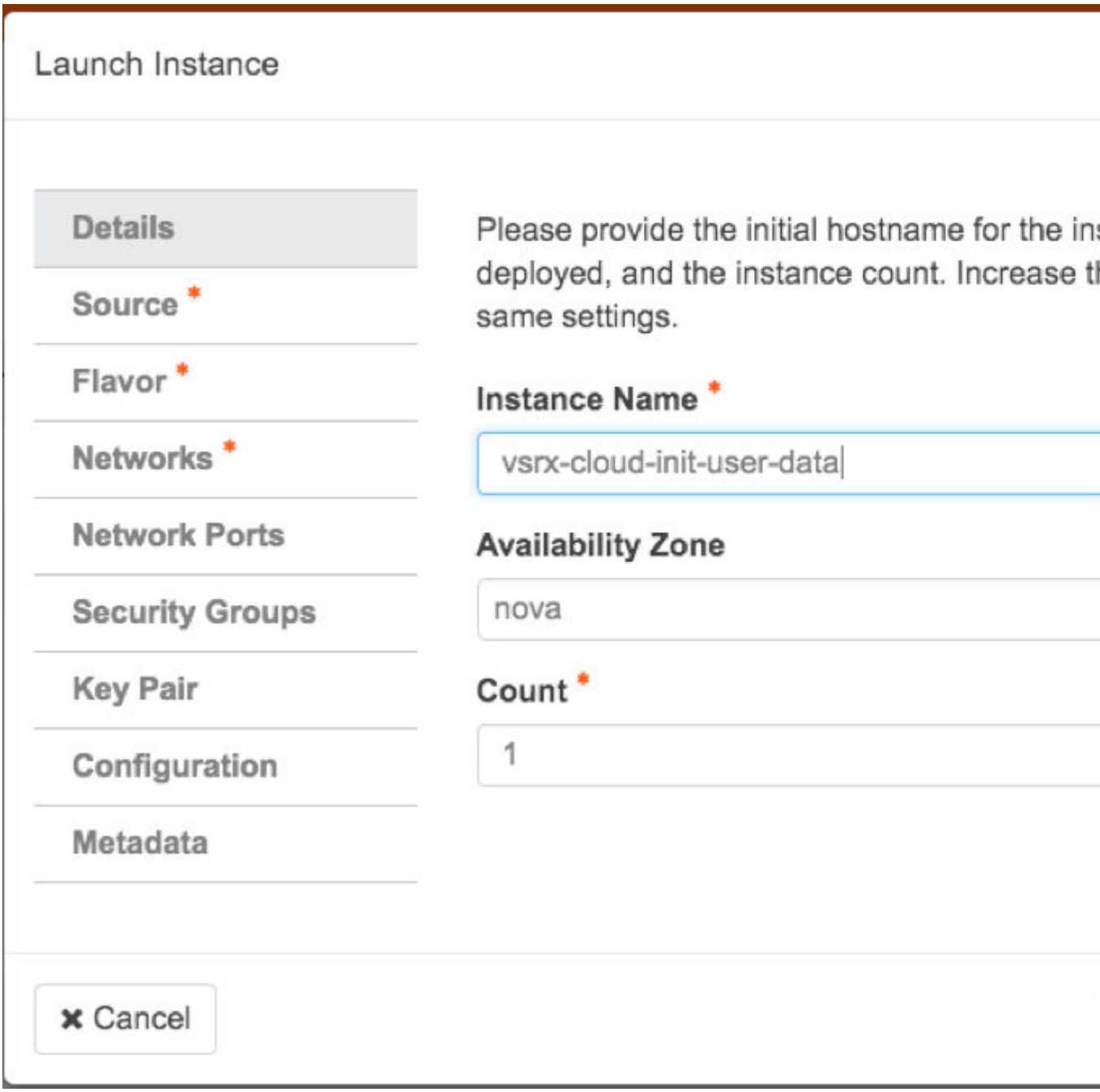
   The user-data configuration file must contain the full vSRX configuration that is to be used as the active configuration on each vSRX instance, and the string **#junos-config** must be the first line of the user-data configuration file before the Junos OS configuration.

   > **NOTE**: The **#junos-config** string is mandatory in the user-data configuration file; if it is not included, the configuration will not be applied to the vSRX instance as the active configuration.

2. Copy the Junos OS configuration file to an accessible location from where it can be retrieved to launch the vSRX instance.
3. Log in to the OpenStack Dashboard using your login credentials and then select the appropriate project from the drop-down menu at the top left.
4. On the Project tab, click the **Compute** tab and select **Instances**. The dashboard shows the various instances with its image name, its private and floating IP addresses, size, status, availability zone, task, power state, and so on.
5. Click **Launch Instance**. The Launch Instance dialog box appears.

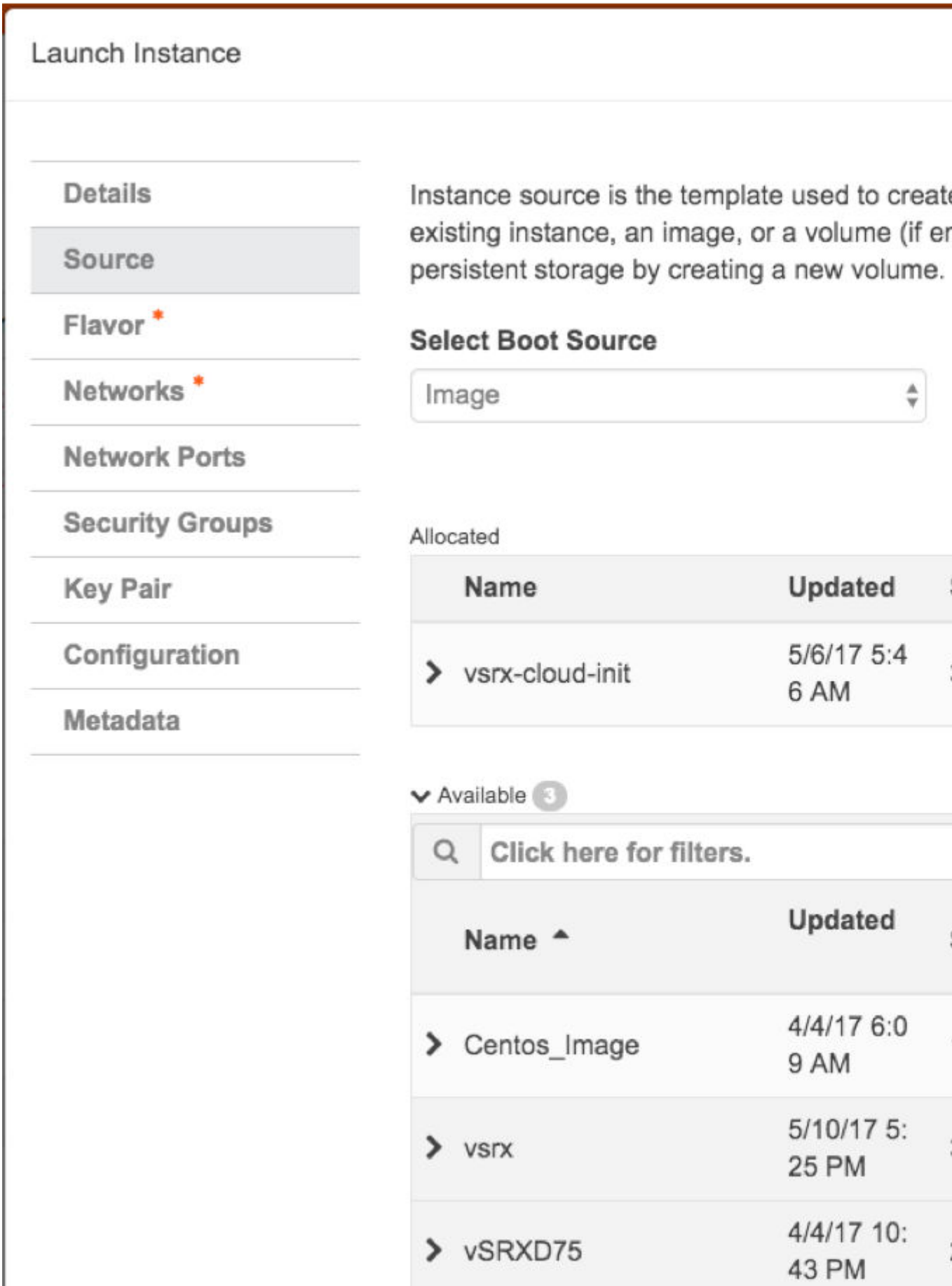
6. From the Details tab (see Figure 7 on page 48), enter an instance name for the vSRX VM along with the associated availability zone (for example, Nova) and then click **Next**. We recommend that you keep this name the same as the hostname assigned to the vSRX VM.

Figure 7: Launch Instance Details Tab



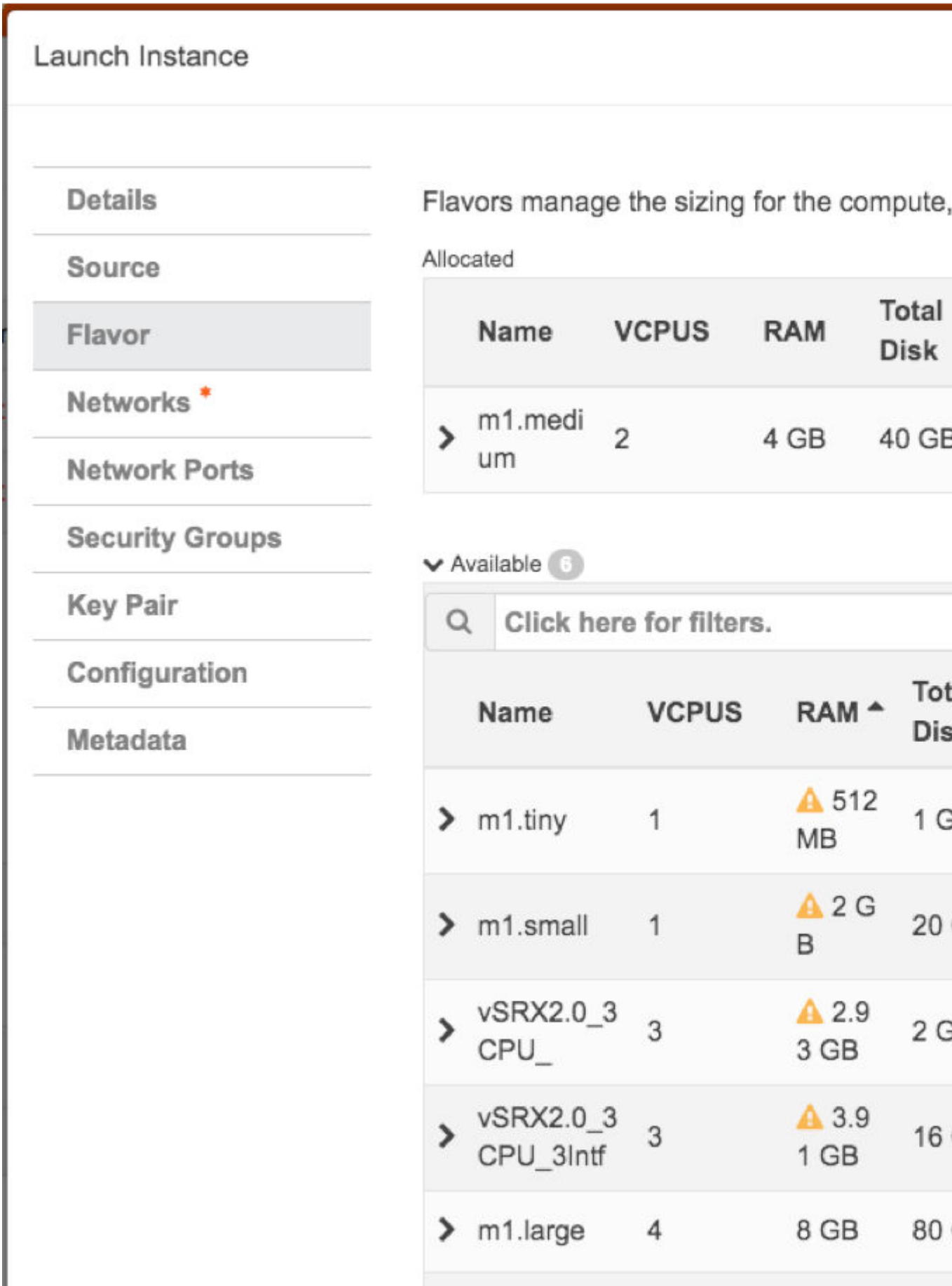
7. From the Source tab (see Figure 8 on page 50), select a vSRX VM image source file from the Available list and then click **+(Plus)**. The selected vSRX image appears under Allocated. Click **Next**.

Figure 8: Launch Instance Source Tab



8. From the Flavor tab (see Figure 9 on page 52), select a vSRX instance with a specific compute, memory, and storage capacity from the Available list and then click **+(plus sign)**. The selected vSRX flavor appears under Allocated. Click **Next**.

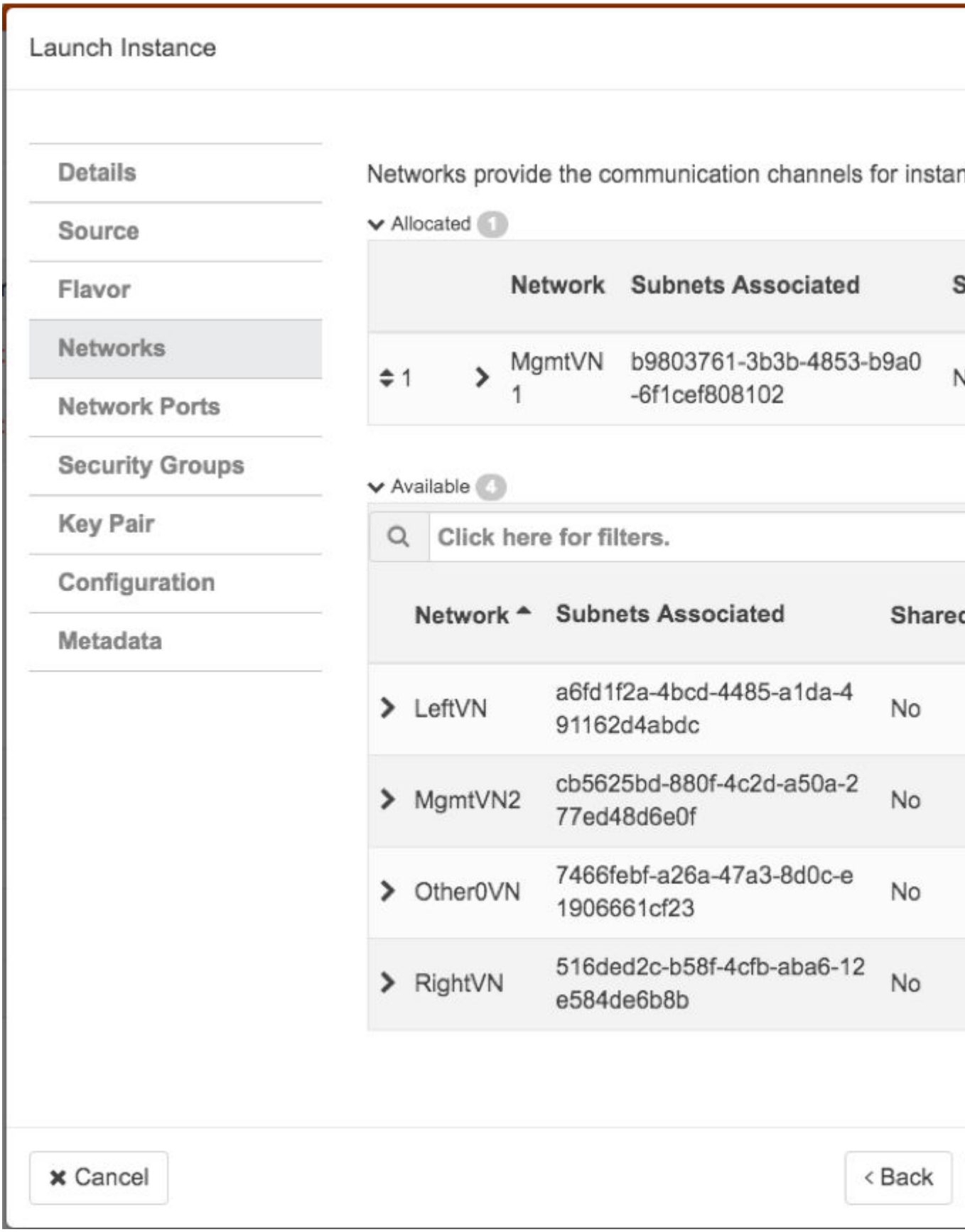
Figure 9: Launch Instance Flavor Tab



9. From the Networks tab (see Figure 10 on page 55), select the specific network of the vSRX instance from the Available list and then click **+(plus sign)**. The selected network appears under Allocated. Click **Next**.

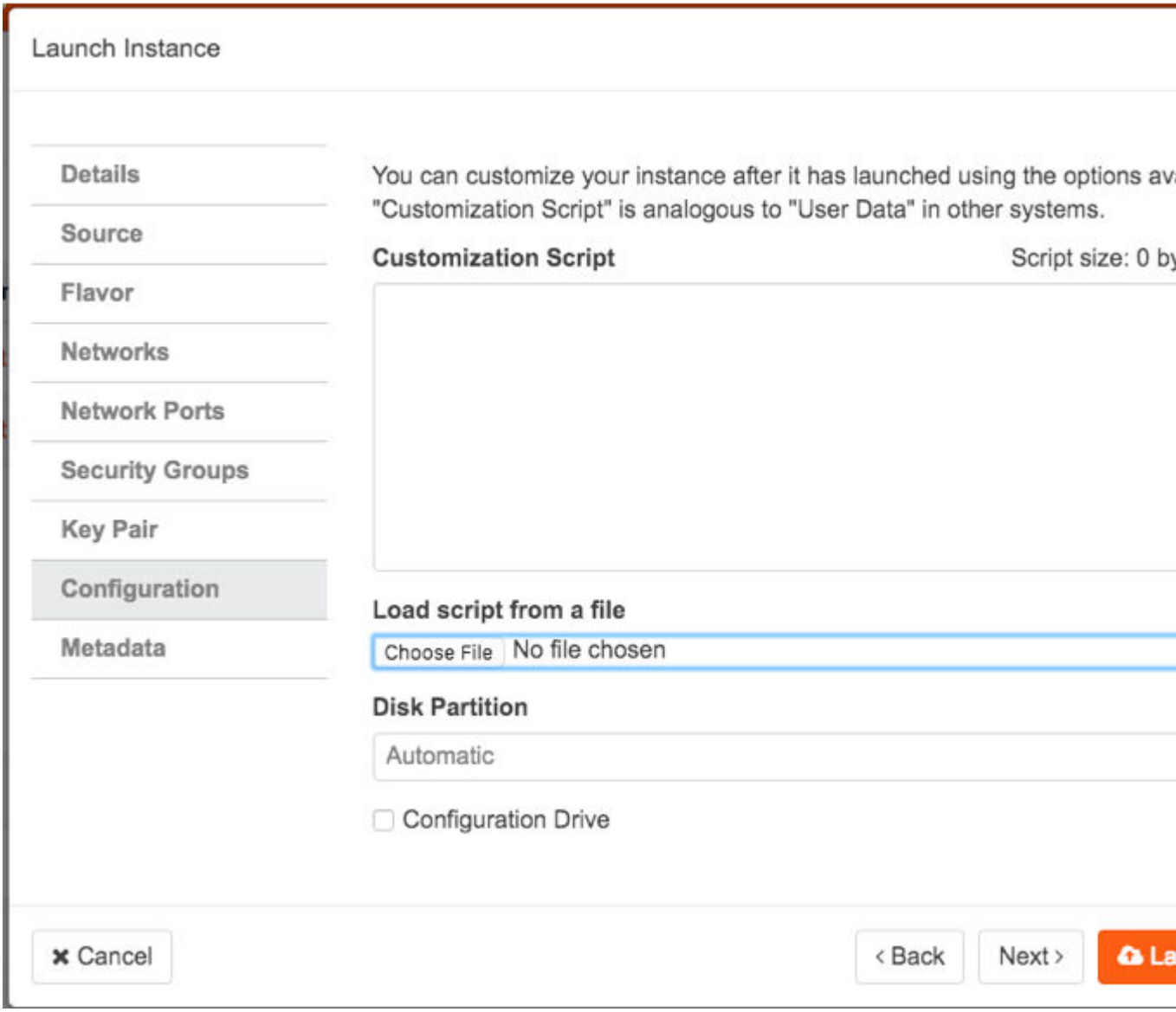
NOTE: Do not update any parameters in the Network Ports, Security Groups, or Key Pair tabs in the Launch Instance dialog box.

Figure 10: Launch Instance Networks Tab



10. From the Configuration tab (see Figure 11 on page 56), click **Browse** and navigate to the location of the validated Junos OS configuration file from your local directory that you want to use as the user-data file. Click **Next**.

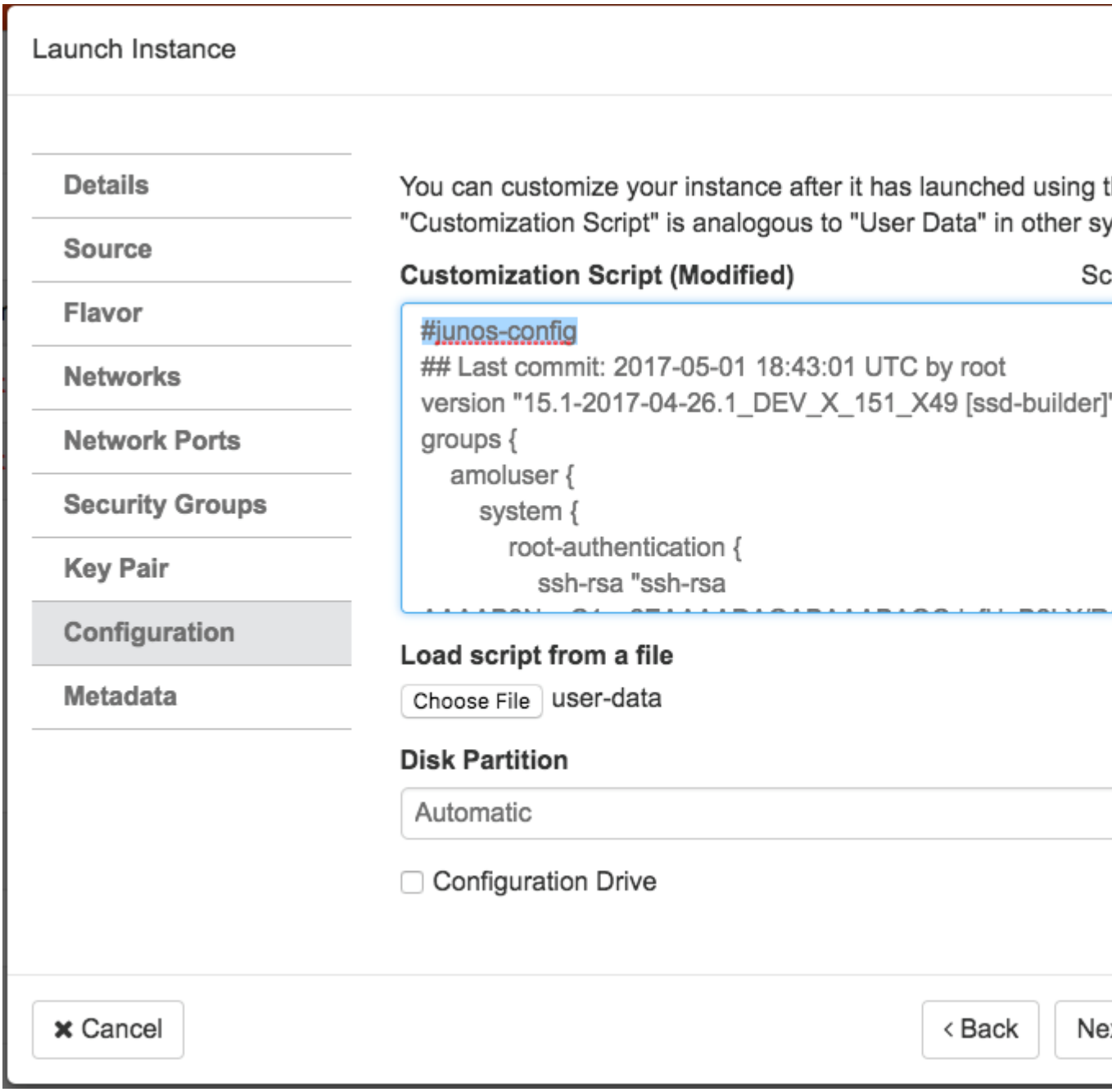
Figure 11: Launch Instance Configuration Tab



11. Confirm that the loaded Junos OS configuration contains the **#junos-config** string in the first line of the user-data configuration file (see Figure 12 on page 57) and then click **Next**.

NOTE: Do not update any parameters in the Metadata tab of the Launch Instance dialog box.

Figure 12: Launch Instance Configuration Tab with Loaded Junos OS Configuration



12. Click **Launch Instance**. During the initial boot-up sequence, the vSRX instance processes the cloud-init request.

> **NOTE**: The boot time for the vSRX instance might increase with the use of the cloud-init package. This additional time in the initial boot sequence is due to the operations performed by the cloud-init package. During this operation, the cloud-init package halts the boot sequence and performs a lookup for the configuration data in each data source identified in the cloud.cfg. The time required to look up and populate the cloud data is directly proportional to the number of data sources defined. In the absence of a data source, the lookup process continues until it reaches a predefined timeout of 30 seconds for each data source.

13. When the initial boot-up sequence resumes, the user-data file replaces the original factory-default Junos OS configuration loaded on the vSRX instance. If the commit succeeds, the factory-default configuration will be permanently replaced. If the configuration is not supported or cannot be applied to the vSRX instance, the vSRX will boot using the default Junos OS configuration.

SEE ALSO

- Cloud-Init Documentation
- OpenStack Dashboard
- Launch and Manage Instances
- Horizon: The OpenStack Dashboard Project

Release History Table

Release	Description
15.1X49-D130	Starting in Junos OS Release 15.1X49-D130 and Junos OS Release 18.4R1, the cloud-init functionality in vSRX has been extended to support the use of a configuration drive data source in an OpenStack environment. The configuration drive uses the user-data attribute to pass a validated Junos OS configuration file to the vSRX instance.
15.1X49-D100	Starting in Junos OS Release 15.1X49-D100 and Junos OS Release 17.4R1, the cloud-init package (version 0.7x) comes pre-installed in the vSRX image to help simplify configuring new vSRX instances operating in an OpenStack environment according to a specified user-data file. Cloud-init is performed during the first-time boot of a vSRX instance.

CHAPTER 3

vSRX Service Chains in Contrail

Overview of Service Chains with vSRX

IN THIS SECTION

You can use Contrail to chain various Layer 2 through Layer 7 services such as *firewall*, *NAT*, and *IDP* through one or more vSRX VMs. For example, you can insert a vSRX firewall VM between two other virtual machines (VMs). By using vSRX and service chains, you can tailor the security needs to a targeted virtual network and VM set. This provides agility and scalability in line with the fluidity of cloud network environments.

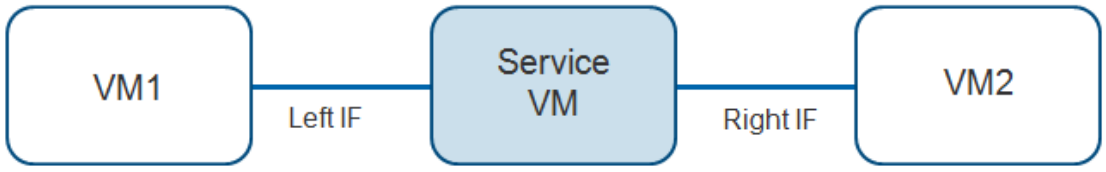
Understanding Service Chains

To create a service through vSRX, you instantiate one or more vSRX VMs to dynamically apply single or multiple services to network traffic.

Figure 13 on page 61 shows a basic service chain with a single vSRX VM. The vSRX service VM spawns a service, such as a firewall. The left interface (left IF) points to the internal end customer, who uses the service; and the right interface (right IF) points to the external network or Internet. You can also

instantiate multiple vSRX VMs to chain multiple services together. For example, you could add an IDP service after the firewall.

Figure 13: vSRX Service Chaining



When you create a service chain, Contrail creates tunnels across the underlay network that span all services in the chain.

Service Chain Modes

You can configure the following service modes:

- Transparent or bridge mode—Used for services that do not modify the packet. Also known as bump-in-the-wire or Layer 2 mode. Examples include Layer 2 firewall and IDP.
- In-network or routed mode—Provides a gateway service that routes packets between the service instance interfaces. Examples include NAT, Layer 3 firewall, and load balancing.
- In-network-nat mode—Similar to in-network mode; however, packets from the left (private) network are not routed to the right (public) source network. In-network-nat mode is particularly useful for NAT services.

> **NOTE**: Ensure that you define the service policy with the private network on the left and public on the right in order to get the public routes (usually the default) advertised into the left network.

Components of a Service Chain

Service chaining requires the following configuration components to build the chain:

- Service template
- Virtual networks
- Service instance
- Network policy

Service Templates

Service templates map out the basic configuration that Contrail uses to instantiate a service instance, or VM. Within Contrail, you configure service templates in the scope of a domain, and you can use the templates on all projects within a domain. You can use a template to launch multiple service instances of the same type in different projects within a domain. Within a service template, you select the service mode, a vSRX image name for the VM that will provide the service, and an ordered list of interfaces for the service. vSRX service VMs require the management interface to be the first interface in that ordered list. You can use OpenStack Horizon or Glance to add the vSRX image. You also select the OpenStack flavor to associate with all service instances that use the service template. An OpenStack flavor defines the number of vCPUs, storage, and memory you can assign to a VM. OpenStack includes default flavors, and you can create new flavors in the OpenStack dashboard.

Virtual Networks

Virtual networks provide the link between the service instance and the network traffic in the virtualized environment. You can create the virtual networks in Contrail or OpenStack and use those networks to direct traffic to or through the service instance.

Service Instances

A service instance is the instantiation of the selected service template to create one or more VMs that provide the service (for example, a firewall). When you create a service instance, you select a service template that defines the instance. You also associate the interfaces in the service template with the virtual networks needed to direct traffic into and out of the service instance. If you enable service scaling in the selected service template, you can instantiate more than one VM when you create the service instance.

Network Policies

By default, all traffic in a virtual network remains isolated. You configure a network policy to allow traffic between virtual networks and through the service instance. The network policy filters traffic to and from the service VM based on the rules you configure. You select the service instance VM and the virtual networks for the right and left interfaces of that VM that the network policy applies to. As a final step. you associate the network policy with each virtual network the policy applies to.

RELATED DOCUMENTATION

Contrail - Service Chaining

Spawn vSRX in a Contrail Service Chain

IN THIS SECTION

Ensure that you have installed Contrail and have loaded the vSRX images with OpenStack Horizon or Glance.

- Installation Overview (Contrail)
- Add the Image Service (glance)

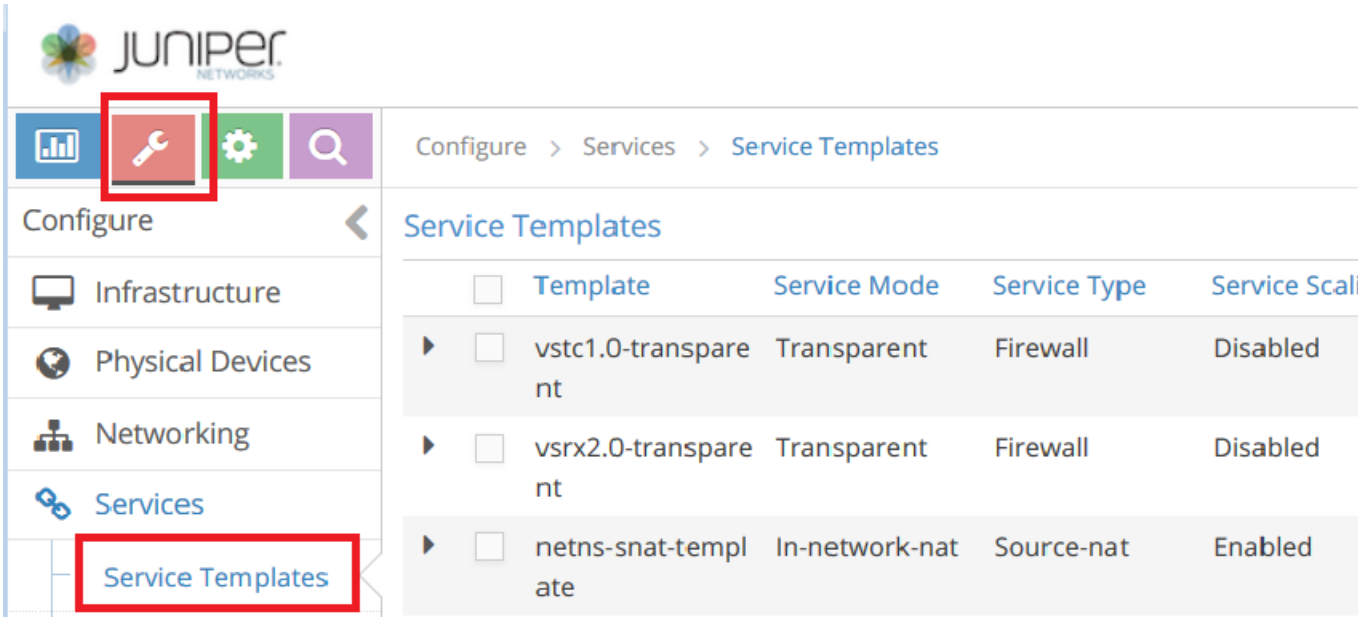
You can use Contrail to chain various Layer 2 through Layer 7 services such as firewall, NAT, and IDP through vSRX VMs.

Create a Service Template

To create a service template:

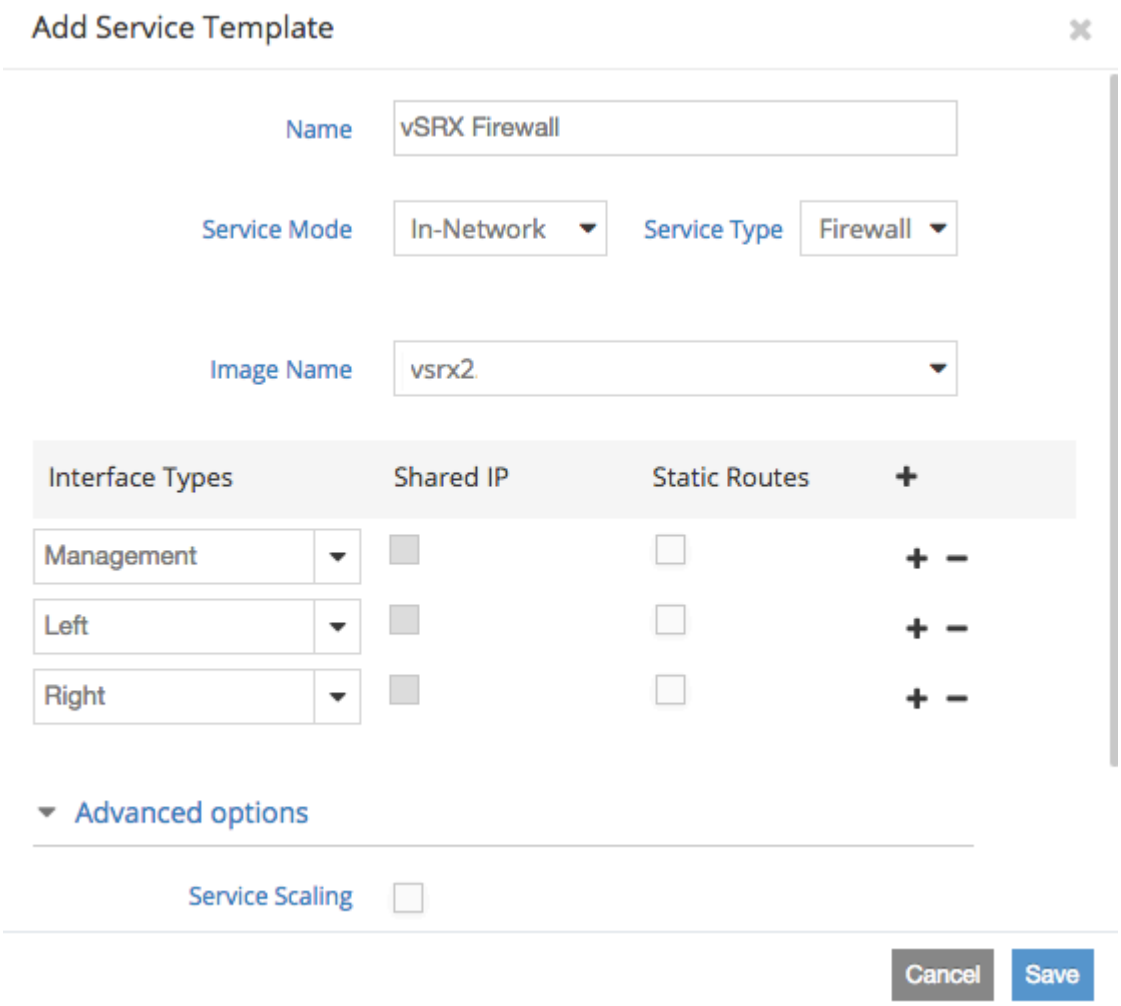
1. From Contrail, select **Configure>Services>Service Templates**. The list of existing service templates appears, as shown in Figure 14 on page 64.

Figure 14: Contrail Service Templates



2. Click **+** to create a new service template. The Add Service Template dialog box appears, as shown in Figure 15 on page 65.

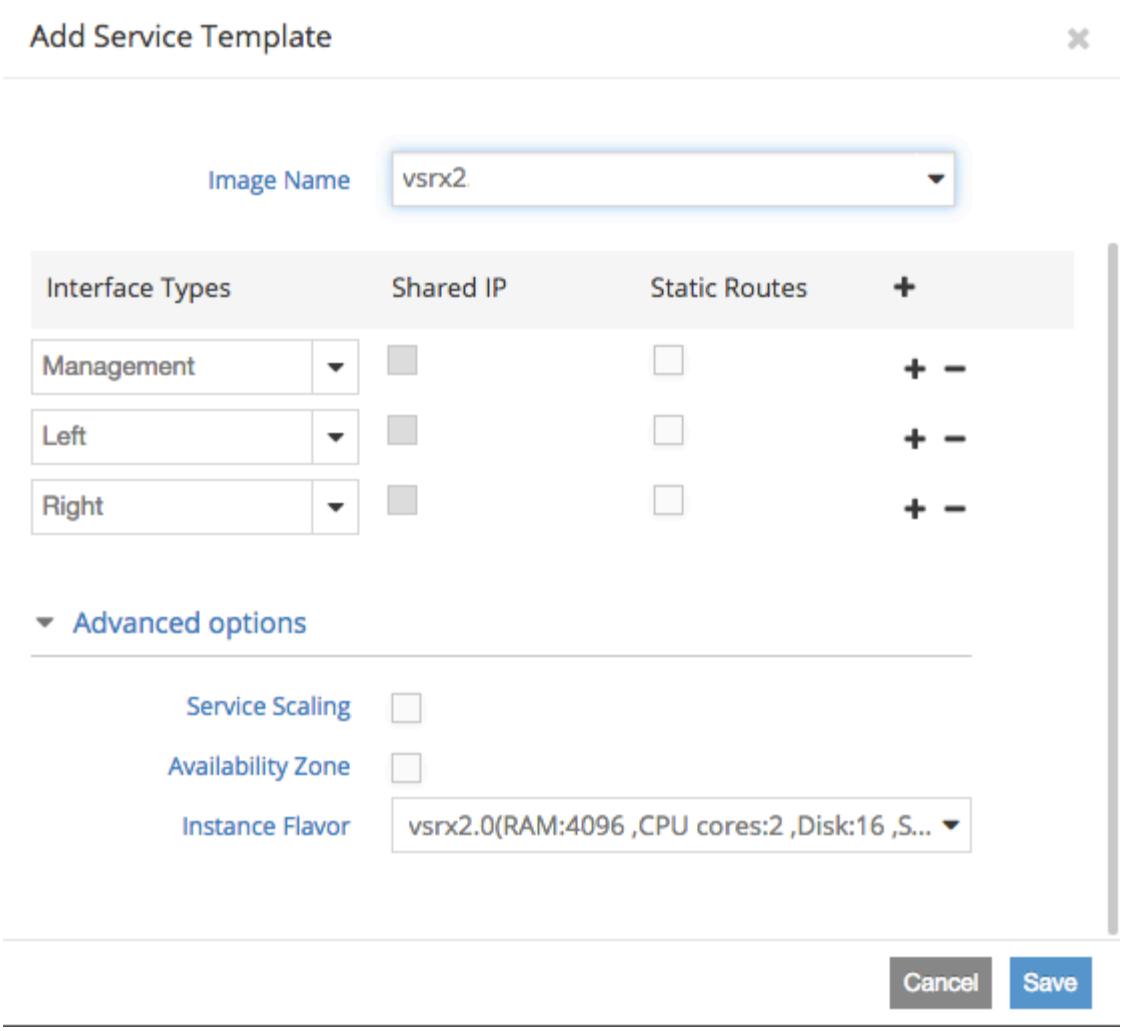
Figure 15: Contrail Add a Service Template



3. Add a name for the service template in the Name box.
4. Select the appropriate service mode and service type from the lists.
5. Select the vSRX image from the Image Name list. This is the image you installed previously in the OpenStack image service.
6. Click **+** to add three interfaces.
7. Select **Management** for the first interface type, **Left** for the second interface type, and **Right** for the third interface type. You associate the left and right interfaces with the left and right virtual networks when you create the service instance. Any additional interfaces must be of type Other.

8. Expand **Advanced Options** and select an instance flavor from the Instance Flavor list, as shown in Figure 16 on page 66. You can use an appropriate default flavor from OpenStack or a custom flavor you created previously for vSRX.

Figure 16: Advanced Options - Add Service Template



9. Optionally, check **Scaling** to create multiple identical vSRX instances from this service template for load balancing.
10. Click **Save** to create this new service template.

See Contrail - Creating an In-Network or In-Network-NAT Service Chain for more details.

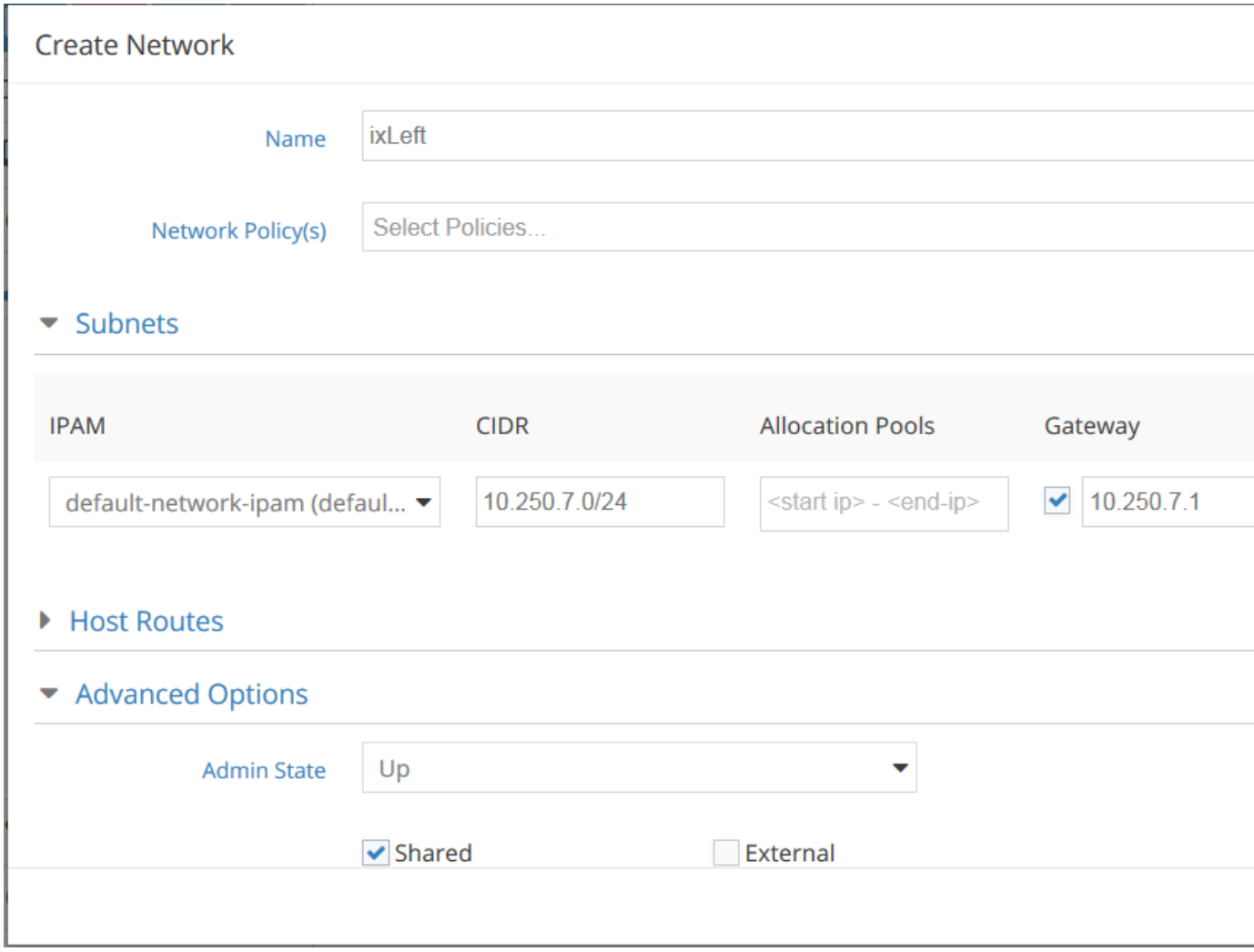
Create Left and Right Virtual Networks

Ensure that you have IP Address Management (IPAM) set up for your project.

To create a virtual network:

1. From Contrail, select **Configure>Networking>Networks**. The list of existing networks appears.
2. Verify that your project is displayed as active in the upper right Project list, and click **+** to create a new virtual network. The Create Network dialog box appears, as shown in Figure 17 on page 67

Figure 17: Creating a Virtual Network in Contrail



3. Enter a name for the left virtual network.

   Do not select a network policy yet. You create the network policy after you create the service instance and then you update this virtual network to add the policy.

4. Expand **Subnet** and click **+** to add IPAM to this virtual network.
5. Select the appropriate IPAM from the list.
6. Set the CIDR and Gateway fields.
7. Expand **Advanced Options** and select appropriate options for your network.
8. Click **Save**. The new virtual network appears in the list of configured networks.
9. Repeat this procedure for the right virtual network.

See Contrail - Creating a Virtual Network for more details

Create a vSRX Service Instance

To create a vSRX service instance:

1. Select **Configure>Services>Service Instances**. The list of existing service instances appears.
2. Click **+** to create a new service instance. The Create Service Instance dialog box appears.
3. Enter a name for the service instance.

   NOTE: Do not use white space in the service instance name.

4. Select the service template you created for vSRX from the Services Template list. This service template includes the vSRX image used to provide the service.
5. Select **Management** from the Interface 1 list. Management must be the first interface for vSRX service instances.
6. Select **Left** from the Interface 2 list, and **Right** from the Interface 3 list.
7. Select **Auto Configured** for the Management interface.
8. Select the left virtual network for the left interface, and the right virtual network for the right interface.
9. Click **Save** to save this service instance. Contrail launches the vSRX VM for this service instance.
10. Optionally, select **Configure>Services>Service Instances** to view this new vSRX instance status. You can expand the row for this instance in the table and click **View Console** to access the vSRX console port.

NOTE: You can also view this service instance from the OpenStack Instances table, but you should only use Contrail to delete service instances.

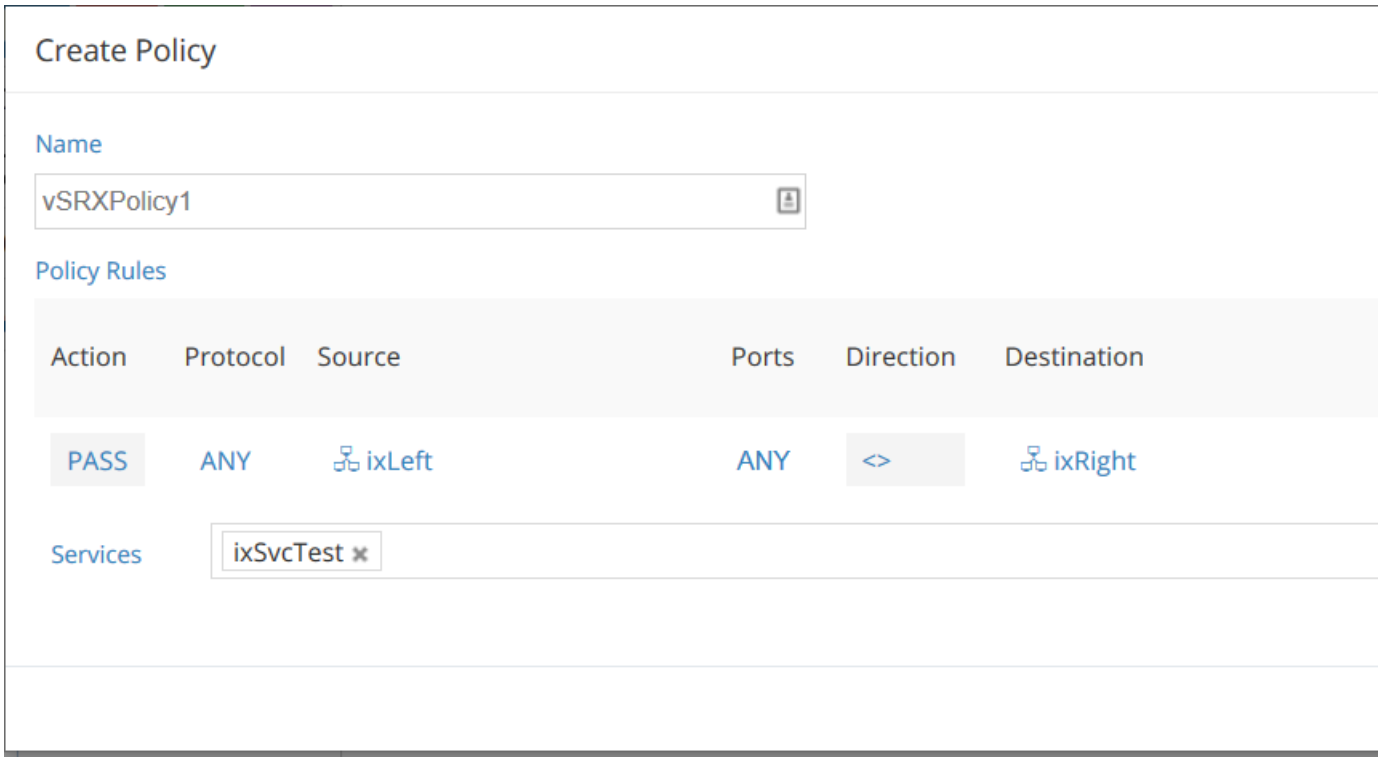
See Contrail - Creating an In-Network or In-Network-NAT Service Chain for more details.

Create a Network Policy

To create a network policy:

1. Select **Configure>Networking>Policies**. The table of policies appears.
2. Click **+** to create a new policy. The Create Policy dialog box appears, as shown in Figure 18 on page 69.

Figure 18: Creating a Network Policy in Contrail



3. Name the policy.
4. Click **+** to create a new rule for this policy.
5. Select the left virtual network you created from the Source list and select the right virtual network from the Destination list.
6. Select the appropriate protocol from the Protocol list and select the source and destination ports for this policy.
7. Select **Services** and select the vSRX instance you want to apply this policy to.
8. Optionally, add more policy rules to this policy.
9. Click **Save** to create this policy.

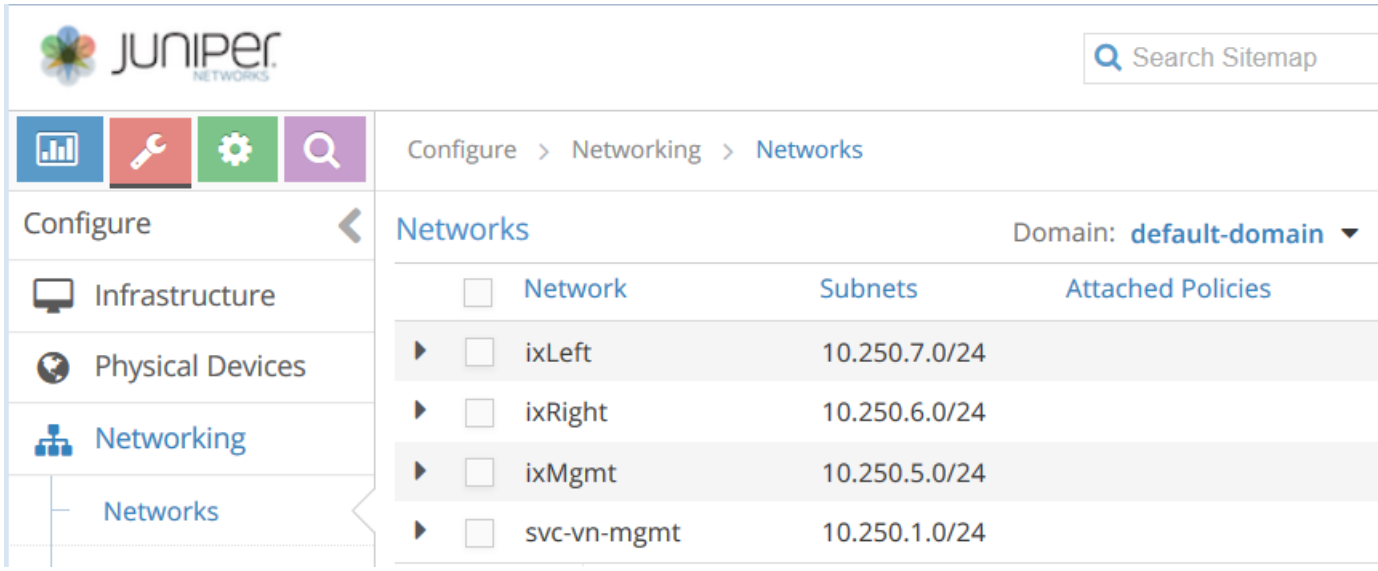
See Contrail - Creating a Network Policy for more details.

Add a Network Policy to a Virtual Network

To add a network policy to a virtual network:

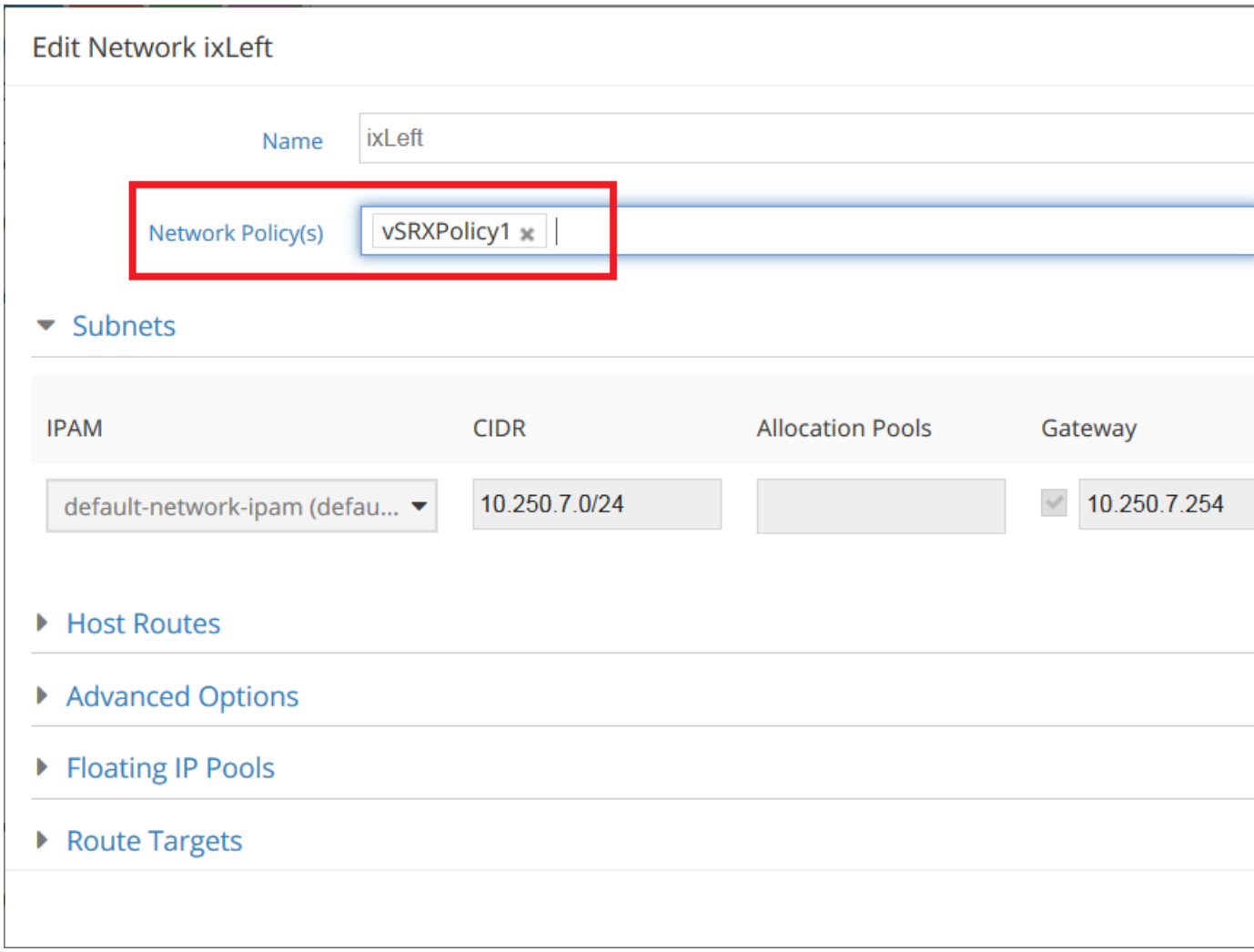
1. Select **Configure>Networking**, and select the settings icon to the right of the virtual network you want to add a network policy to, as shown in Figure 19 on page 70.

Figure 19: Contrail Virtual Networks



2. Click **Edit**. The Edit Networks dialog box appears, as shown in Figure 20 on page 71.

Figure 20: Adding a Network Policy to a Virtual Network in Contrail



3. Select the appropriate policy from the Networks Policy(s) list.
4. Click **Save** to save this change.
5. Repeat this procedure for the other virtual network in this service chain.

See Contrail - Associating a Network to a Policy for more details.

RELATED DOCUMENTATION

- Contrail - Creating an In-Network or In-Network-NAT Service Chain
- Contrail - Installation Overview

CHAPTER 4

vSRX VM Management

Connect to the vSRX Management Console

IN THIS SECTION

Ensure that you have launched the vSRX VM with Contrail.

You can connect to the vSRX console through OpenStack or Contrail.

Connect to the vSRX Management Console with Horizon

To connect to the vSRX console with OpenStack Horizon:

1. From the Horizon GUI, select your project, and select **Compute>Instances**. The list of existing instances appears.
2. From the Actions column, select **Console** from the More list. The vSRX console appears, and you can log in to the management port for the vSRX instance.

Connect to the vSRX Management Console with Contrail

To connect to the vSRX console of a vSRX VM with Contrail:

1. From the Contrail GUI, select your project, and select **Configure>Services>Service Instances**. The list of existing service instances appears.
2. Click on the left arrow next to the vSRX VM to expand to the Service Instance Details view.
3. Click the **View Console** link on the right. The vSRX console appears, and you can log in to the management port for the vSRX.

RELATED DOCUMENTATION

OpenStack End User Guide

Manage the vSRX VM

IN THIS SECTION

Each vSRX instance is an independent *virtual machine* (VM) that you can power on, pause, or shut down.

Power On the VM from OpenStack

To power on the VM:

1. From the OpenStack dashboard for your project, select **Compute>Instances**. The list of existing instances appears.
2. Check the VM you want to power on.
3. From the Actions column, select **Start Instance** from the list.

Pause the VM

To pause the VM:

1. From the Horizon GUI for your project, select **Compute>Instances**. The list of existing instances appears.
2. Check the VM that you want to pause.
3. From the Actions column, select **Pause Instance** from the list.

Restart the VM

To restart the VM:

1. From the Horizon GUI for your project, select **Compute>Instances**. The list of existing instances appears.
2. Check the VM that you want to reboot.
3. Select **Soft Reboot Instance** from the More list to restart the VM.

Power Off the VM from OpenStack

To power off the VM:

1. From the OpenStack dashboard for your project, select **Compute>Instances**. The list of existing instances appears.
2. Check the VM you want to power off.
3. From the Actions column, select **Console** from the list. The console opens.
4. From the console, power off the VM.

   user@host>request system power-off

Delete the vSRX VM from Contrail

BEST PRACTICE: We recommend that you use Contrail to delete any VMs used in service chains created by Contrail.

To delete the VM from Contrail:

1. From the Contrail GUI for your project, select **Configure>Services>Service Instances**. The list of existing service instances appears.
2. Select the VM that you want to delete.
3. Click trash icon on the upper right menu to delete the selected VMs.

RELATED DOCUMENTATION

Contrail - Creating an In-Network or In-Network-NAT Service Chain

Upgrade Multicore vSRX with Contrail

IN THIS SECTION

Starting in Junos OS Release 15.1X49-D70 and Junos OS Release 17.3R1, you can scale up the number of vCPUs or vRAM for a vSRX VM. You must gracefully power off the vSRX VM before you can scale up vSRX. See "Manage the vSRX VM" on page 74 for details.

You can modify an existing flavor with the OpenStack Dashboard (Horizon). You cannot use the OpenStack CLI (**nova flavor)** commands to modify the CPU or RAM settings on an existing flavor. Instead, create a new flavor and modify the vSRX service template in Contrail to use this new flavor. See the "Create an Image Flavor with OpenStack" on page 33 for details.

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

Configure Multi-queue Virtio Interface for vSRX VM with OpenStack

Before you plan to scale up vSRX performance, enable network multi-queuing as a means to support an increased number of dataplane vCPUs for the vSRX VM. The default for vSRX in Contrail is 2 dataplane vCPUs, but you can scale that number to 4 vCPUs.

To use multiqueue virtio interfaces, ensure your system meets the following requirements:

OpenStack Liberty supports the ability to create VMs with multiple queues on their virtio interfaces. Virtio is a Linux platform for I/O virtualization, providing a common set of I/O virtualization drivers. Multiqueue 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 OpenStack version must be Liberty or greater.
- The maximum number of queues in the vSRX VM interface is set to the same value as the number of vCPUs in the guest.
- The vSRX VM image metadata property is set to enable multiple queues inside the VM.

Use the following command on the OpenStack node to enable multiple queues on a vSRX VM in Contrail:

source /etc/contrail/openstackrc

nova image-meta <image_name> set hw_vif_multiqueue_enabled="true"

After the vSRX VM is spawned, use the following command on the virtio interface in the guest to enable multiple queues inside the vSRX VM:

ethtool –L <interface_name> combined <#queues>

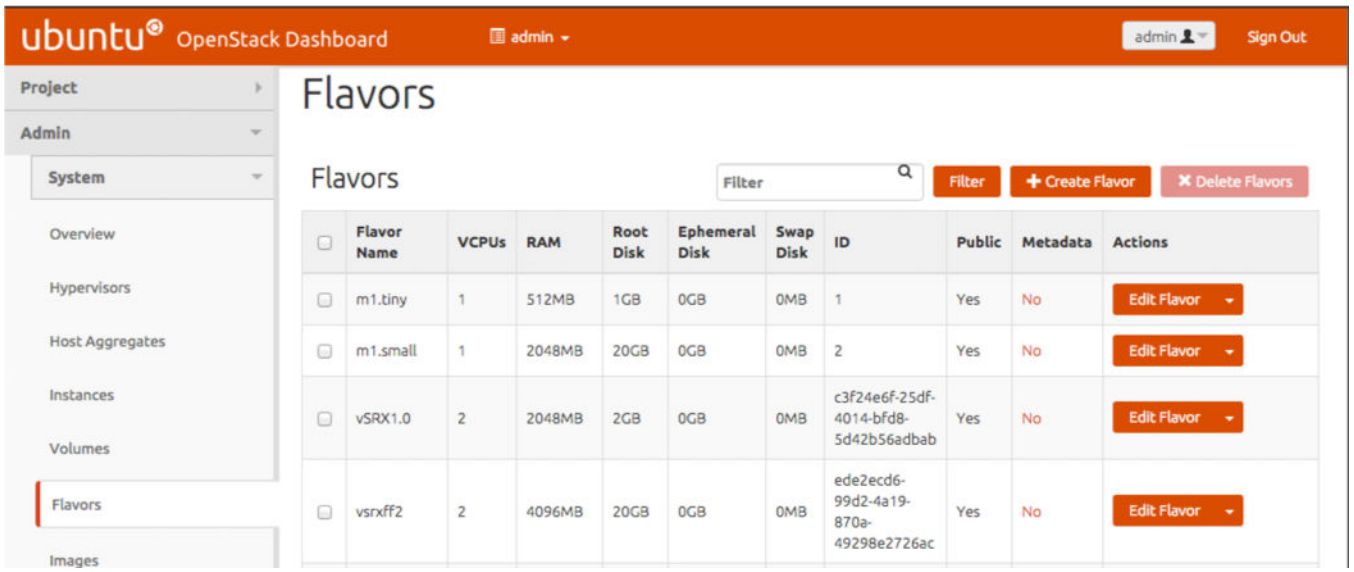
Modify an Image Flavor for vSRX with the Dashboard

OpenStack uses VM templates, or flavors, to set the memory, vCPU, and storage requirements for an image.

To Modify an image flavor for vSRX with the OpenStack dashboard:

1. From the dashboard select your project, and select **Admin>System Panel>Flavors**. The list of existing image flavors appears, as shown in Figure 21 on page 78.

Figure 21: OpenStack Flavors



2. Select the vSRX flavor and click **Edit Flavor**. The Edit Flavor dialog box appears.
3. Increase the number of vCPUs for your configuration. The minimum required for vSRX is 2 vCPUs.
4. Increase the RAM MB value. The minimum required for vSRX is 4096 MB.
5. Click **Create Flavor**. The flavor appears on the Flavors tab.

Update a Service Template

If you created a new image flavor for an existing vSRx instance, you need to update the service template to use this new image flavor before you relaunch the vSRX instance.

To update a service template:

1. From Contrail, select **Configure>Services>Service Templates**. The list of existing service templates appears.
2. Click on the vSRX service template and select edit.
3. Expand **Advanced Options** and select the new instance flavor from the Instance Flavor list.
4. Click **Save** to update this service template.
5. Power on the vSRX VM. See "Manage the vSRX VM" on page 74 for details.

Release History Table

Release	Description
15.1X49-D70	Starting in Junos OS Release 15.1X49-D70 and Junos OS Release 17.3R1, you can scale up the number of vCPUs or vRAM for a vSRX VM.

RELATED DOCUMENTATION

- OpenStack Installation Guide
- OpenStack End User Guide

Monitor vSRX with Contrail

To monitor basic statistics on the vSRX VM with Contrail:

1. On Contrail, select **Monitor>Networking>Instances**. The list of existing VMs appears.
2. Expand the row for the VM that you want to monitor. The CPU and memory statistics appear.
3. On Contrail, select **Monitor>Networking>Networks**. The list of existing virtual networks appears.
4. Expand the row for the virtual network that you want to monitor and select **Traffic Statistics**. The traffic and throughput statistics appear.

RELATED DOCUMENTATION

- Contrail - Monitor Networking

5

CHAPTER

Configuring and Managing vSRX

vSRX Configuration and Management Tools

SUMMARY

This topic provides an overview of the various tools available to configure and manage a vSRX VM once it has been successfully deployed.

IN THIS SECTION

Understanding the Junos OS CLI and Junos Scripts

Junos OS CLI is a Juniper Networks specific command shell that runs on top of a UNIX-based operating system kernel.

Built into Junos OS, Junos script automation is an onboard toolset available on all Junos OS platforms, including routers, switches, and security devices running Junos OS (such as a vSRX instance).

You can use the Junos OS CLI and the Junos OS scripts to configure, manage, administer, and troubleshoot vSRX.

Understanding the J-Web Interface

The *J-Web* interface allows you to monitor, configure, troubleshoot, and manage vSRX instances by means of a Web browser. J-Web provides access to all the configuration statements supported by the vSRX instance.

Understanding Junos Space Security Director

As one of the Junos Space Network Management Platform applications, Junos Space Security Director helps organizations improve the reach, ease, and accuracy of security policy administration with a scalable, GUI-based management tool. Security Director automates security provisioning of a vSRX

instance through one centralized Web-based interface to help administrators manage all phases of the security policy life cycle more quickly and intuitively, from policy creation to remediation.

RELATED DOCUMENTATION

- CLI User Interface Overview
- J-Web Overview
- Security Director
- Mastering Junos Automation Programming
- Spotlight Secure Threat Intelligence

Configure vSRX Using the CLI

To configure the vSRX instance using the CLI:

1. Verify that the vSRX is powered on.
2. Log in as the root user. There is no password.
3. Start the CLI.

```
root#cli
root@>
```

4. Enter configuration mode.

```
configure
[edit]
root@#
```

5. Set the root authentication password by entering a *cleartext* password, an encrypted password, or an SSH public key string (*DSA* or *RSA*).

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

6. Configure the hostname.

```
[edit]
root@# set system host-name host-name
```

7. Configure the management interface.

```
[edit]
root@# set interfaces fxp0 unit 0 family inet dhcp-client
```

8. Configure the traffic interfaces.

```
[edit]
root@# set interfaces ge-0/0/0 unit 0 family inet dhcp-client
```

9. Configure basic security zones and bind them to traffic interfaces.

```
[edit]
root@# set security zones security-zone trust interfaces ge-0/0/0.0
```

10. Verify the configuration.

```
[edit]
root@# commit check
configuration check succeeds
```

11. Commit the configuration to activate it on the vSRX instance.

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

12. Optionally, use the **show** command to display the configuration to verify that it is correct.

NOTE: Certain Junos OS software features require a license to activate the feature. To enable a licensed feature, you need to purchase, install, manage, and verify a license key that corresponds to each licensed feature. To conform to software feature licensing requirements, you must

purchase one license per feature per instance. The presence of the appropriate software unlocking key on your virtual instance allows you to configure and use the licensed feature. See Managing Licenses for vSRX for details.

RELATED DOCUMENTATION

CLI User Guide

Configuring vSRX Using the J-Web Interface

IN THIS SECTION

Accessing the J-Web Interface and Configuring vSRX

Use the Junos OS CLI to configure, at a minimum, the following parameters before you can access a vSRX VM using J-Web:

- Configure an IP address on fxp0.
- Configure a default route if the fxp0 IP address is on a different subnet than the host server.
- Enable Web management through the fxp0 interface.

```
system {
    services {
        web-management {
            http {
                interface fxp0.0;
```

```
                    }
                }
            }
        }
```

To configure vSRX using the *J-Web* Interface:

1. Launch a Web browser from the management instance.
2. Enter the vSRX fxp0 interface IP address in the Address box.
3. Specify the username and password.
4. Click **Log In**, and select the **Configuration Wizards** tab from the left navigation panel. The J-Web Setup wizard page opens.
5. Click **Setup**.

   You can use the Setup wizard to configure the vSRX VM or edit an existing configuration.

   - Select **Edit Existing Configuration** if you have already configured the wizard using the factory mode.
   - Select **Create New Configuration** to configure the vSRX VM using the wizard.

     The following configuration options are available in the guided setup:

     - Basic

       Select **basic** to configure the vSRX VM name and user account information as shown in Table 13 on page 85.

       - Instance name and user account information

         Table 13: Instance Name and User Account Information

Field	Description
Instance name	Type the name of the instance. For example: **vSRX**.
Root password	Create a default root user password.
Verify password	Verify the default root user password.

Table 13: Instance Name and User Account Information *(Continued)*

Field	Description
Operator	Add an optional administrative account in addition to the root account. User role options include: • **Super User**: This user has full system administration rights and can add, modify, and delete settings and users. • **Operator**: This user can perform system operations such as a system reset but cannot change the configuration or add or modify users. • **Read only**: This user can only access the system and view the configuration. • **Disabled**: This user cannot access the system.

- Select either **Time Server** or **Manual**. Table 14 on page 86 lists the system time options.

Table 14: System Time Options

Field	Description
Time Server	
Host Name	Type the hostname of the time server. For example: **ntp.example.com**.
IP	Type the IP address of the time server in the IP address entry field. For example: **192.0.2.254**.
NOTE: You can enter either the hostname or the IP address.	
Manual	
Date	Click the current date in the calendar.

Table 14: System Time Options *(Continued)*

Field	Description
Time	Set the hour, minute, and seconds. Choose **AM** or **PM**.
Time Zone (mandatory)	
Time Zone	Select the time zone from the list. For example: GMT Greenwich Mean Time GMT.

- Expert

  a. Select **Expert** to configure the basic options as well as the following advanced options:

  - Four or more internal zones
  - Internal zone services
  - Application of security policies between internal zones

  b. Click the **Need Help** icon for detailed configuration information.

  You see a success message after the basic configuration is complete.

Applying the Configuration

To apply the configuration settings for vSRX:

1. Review and ensure that the configuration settings are correct, and click **Next**. The Commit Configuration page appears.
2. Click **Apply Settings** to apply the configuration changes to vSRX.
3. Check the connectivity to vSRX, as you might lose connectivity if you have changed the management zone IP. Click the URL for reconnection instructions on how to reconnect to the instance.
4. Click **Done** to complete the setup.

   After successful completion of the setup, you are redirected to the J-Web interface.

CAUTION: After you complete the initial setup, you can relaunch the J-Web Setup wizard by clicking **Configuration>Setup**. You can either edit an existing configuration or create a new configuration. If you create a new configuration, the current configuration in vSRX will be deleted.

Adding vSRX Feature Licenses

Certain Junos OS software features require a license to activate the feature. To enable a licensed feature, you need to purchase, install, manage, and verify a license key that corresponds to each licensed feature. To conform to software feature licensing requirements, you must purchase one license per feature per instance. The presence of the appropriate software unlocking key on your virtual instance allows you to configure and use the licensed feature.

See Managing Licenses for vSRX for details.

Software Receive Side Scaling

IN THIS SECTION

Overview

Contemporary NICs support multiple receive and transmit descriptor queues (multi-queue). On reception, a NIC can send different packets to different queues to distribute processing among CPUs. The NIC distributes packets by applying a filter to each packet that assigns it to one of a small number of logical flows. Packets for each flow are steered to a separate receive queue, which in turn can be processed by separate CPUs. This mechanism is generally known as Receive-side Scaling (RSS). The goal of RSS technique is to increase performance uniformly. RSS is enabled when latency is a concern or whenever receive interrupt processing forms a bottleneck. Spreading load between CPUs decreases

queue length. For low latency networking, the optimal setting is to allocate as many queues as there are CPUs in the system (or the NIC maximum, if lower). The most efficient high-rate configuration is likely the one with the smallest number of receive queues where no receive queue overflows due to a saturated CPU. You can improve bridging throughput with Receive Side Scaling.

As per flow thread affinity architecture each flow thread (FLT) polls for packet from dedicated receiving queue of NIC and process the packets until run to completion. Therefore, flow threads are bound to NIC receiving (RX) and transmitting (TX) queues for packet processing to avoid any disagreement. Hence, NIC must have same number of RX and TX queues as number of vSRX data plane CPU to support multi core vSRX flavors. Software RSS (SWRSS) removes this limitation of NIC HW queues to run vSRX multi-core flavors by implementing software-based packet spraying across various FLT thread.

Software RSS offloads the handling of individual flows to one of the multiple kernel, so the flow thread that takes the packets from the NIC can process more packets. Similar to RSS, network throughput improvement when using SWRSS has a linear correlation with CPU utilization.

In SWRSS, each NIC port is initialized with equal or lesser number of hardware RX/TX queues as that of I/O threads. I/O threads are determined based on total data-path CPU and minimum of NIC queues among all the NIC interface in vSRX. For example, if I/O thread is computed as 4, then number of HW queue per NIC port can be less or equal to 4 queues.

If NICs do not have sufficient number of queues as FLT threads in vSRX instances supported, then Software RSS (SWRSS) is enabled by flowd data-path. SWRSS implements software model of packet distribution across FLTs after obtaining the packets from NIC receiving queues. By removing NIC HW queue limitation, SWRSS helps to scale vCPUs by supporting various vSRX instance types.

During the I/O operation the packets are fetched from receiving queues of NIC ports and packet classification is performed. Followed by distribution of packets to FLT threads virtual queues. These virtual queues are implemented over DPDK ring queue. In the transmission path, SWRSS fetches the packets from virtual transmitting queues of FLT threads and pushes these packets to NIC transmitting queues for transmit.

Number of SWRSS I/O threads are selected based on total CPU and number of NIC queues found in vSRX instances. Mix mode of operation with HWRSS and and SWRSS is not supported.

Understanding Software Receive Side Scaling Configuration

This topic provide you details on types of Software Receive Side Scaling (SWRSS) and its configuration.

SWRSS supports two modes of operation and it gets enabled based on number of data-path CPU needed. These modes are Shared IO mode and dedicated IO mode. These modes are enabled based on number of data-path CPUs needed. vSRX and vSRX3.0 supports dedicated I/O mode only.

In dedicated I/O mode flowd process creates dedicated I/O threads for I/O operation. Based on number of required I/O threads for vSRX, I/O thread is associated to a dedicated NIC port. NIC ports receiving and transmitting queue is then bonded to each I/O thread in round robin method for uniform distribution and to avoid I/O thread locks. Each dedicated I/O thread pulls the packets in burst mode from NIC receiving queue and distributes to FLT threads and vice versa for TX path for packet transmit.

SWRSS is enabled based on the number of vCPUs. If NIC does not have sufficient number of queues as flow thread (FLT) in vSRX with different vCPUs, then Software RSS (SWRSS) is enabled by flowd process.

SWRSS is not enabled in the following scenarios:

- When the NIC has sufficient number of hardware RX or TX queues for required PFE data-path CPU.
- When the vSRX (based on number of vCPUs) and NIC result the smaller number of FLT CPUs as that obtained in nearest hardware RSS (HWRSS) mode. In such scenario, vSRX will be enabled with HWRSS mode which results more FLT CPU than SWRSS mode, providing better packet processing throughput.
- SWRSS is not recommended for vSRX with certain type of NIC that supports lesser number of NIC queues than needed to run dedicated IO thread. In such cases, SWRSS is enabled but extra CPUs are attached to FLT CPU, until I/O CPUs are completely utilized.

If SWRSS is not enabled use the **set security forwarding-options receive-side-scaling software-rss mode enable** command to enable SWRSS. When you run this command SWRSS will be enabled by force regardless of the NIC RSS or the number of vCPUs. If you do not enable SWRSS using the CLI then enabling of SWRSS automatically is decided based on the default ratio of FLT: IO (4:1).

To configure the number of required IO threads, use the **set security forwarding-options receive-side-scaling software-rss io-thread-number** ***<1-8>*** command. To view the actual number of vCPUs assigned to IO flow threads use the **show security forwarding-options resource-manager** command.

You can decide enabling of SWRSS automatically or by force based on the architecture and conception of IO thread and worker thread. Enabling SWRSS impacts the performance, so we recommend that the number of IO thread should be changed only if required and until the performance impact bottleneck point is reached.

GTP Traffic with TEID Distribution and SWRSS

IN THIS SECTION

Overview GTP Traffic Distribution with TEID Distribution and SWRSS

IN THIS SECTION

The topic provides an overview of asymmetric fat tunnel solution for GTP traffic with TEID distribution and SWRSS.

With TEID-based hash distributions feature, the GTP packets would be distributed to the flow thread according to the hash value calculated by TEID. The algorithm of hash calculation is same as GTP distribution in flow module, which ensures the GTP packets would not be reinjected again in the flow process.

There is a 4-byte field inside GTP payload called tunnel endpoint identifier (TEID), which is used to identify different connections in the same GTP tunnel.

A fat GTP tunnel carries data from different users. IPsec tunnels on the security gateway could be a fat tunnel due to the fat GTP tunnel. vSRX can create one GTP session with a high-bandwidth of GTP traffic. However, the throughput is limited to one core processor's performance.

If you use TEID-based hash distribution for creating GTP-U sessions, then you can:

- Enable vSRX and vSRX 3.0 instances to process asymmetric fat tunnels for parallel encryption on multiple cores for one tunnel.
- You can split a fat GTP session to multiple sessions and distribute them to different cores. This helps to increase the bandwidth for fat GTP tunnel.

The TEID based hash distribution creates GTP-U sessions to multiple cores. The clear text traffic acts as a fat GTP tunnel. This helps a fat GTP session to split into multiple slim GTP sessions and handle them on multiple cores simultaneously.

GTP Traffic Performance with TEID Distribution and SWRSS

vSRX instances support Software Receive Side Scaling (SWRSS) feature. SWRSS is a technique in the networking stack to increase parallelism and improve performance for multi-processor systems. If NICs do not have sufficient number of queues as flow thread (FLT), based on vSRX type, then Software RSS (SWRSS) is enabled by flowd process.

With Software Receive Side Scaling (SWRSS) support on vSRX and vSRX 3.0, you can assign more vCPUs to the vSRX regardless of the limitation of RSS queue of underlying interfaces.

Based on SWRSS you can improve the GTP traffic performance using Tunnel endpoint identifier (TEID) distribution and asymmetric fat tunnel solution by:

- Assigning specific number of vCPUs for input output flow usage—With SWRSS enabled, you can assign more vCPUs for input/output (IO) threads when the IO threads are less. Or you can assign less vCPUs for IO threads if the flow process is consuming more vCPU. Use the **set security forwarding-options receive-side-scaling software-rss io-thread-number <io-thread-number>**.

- Distributing the packets to flow threads according to the TEID inside the packet, which would avoid reinjecting the packets in flow process—This feature is enabled when both SWRSS is enabled and when you configure the **set security forwarding-process application-services enable-gtpu-distribution** command.

  With this feature, the GTP packets would be distributed to the flow thread according to the hash value calculated by TEID. The algorithm of hash calculation is same as GTP distribution in flow module, which ensures the GTP packets would not be reinjected again in flow process.

- Utilizing fragment matching and forwarding mechanism in input/output thread when GTPU distribution is enabled—This mechanism ensures that all the fragments of the same packet would be distributed to one flow thread according to the TEID.

  SWRSS uses IP pair hash to distribute packets to flow threads. For GTP traffic with GTPU distribution enabled, TEID distribution is used to distribute packets to the flow threads. For fragmented packets, TEID cannot be retrieved from non-first fragments. This will require fragment matching and forwarding logic to ensure all fragments are forwarded to the flow thread based on TEID.

Enabling GTP-U TEID Distribution with SWRSS for Asymmetric Fat Tunnels

The following configuration helps you enable PMI and GTP-U traffic distribution with SWRSS enabled.

Before you begin, understand:

- SWRSS concepts and configurations.
- How to establish PMI and GTP-U

With Software Recieve Side Scaling (SWRSS) enabled, you can assign more vCPUs for input/output (IO) threads when the IO threads are less. Or you can assign less vCPUs for IO threads if the flow process is consuming more vCPU. You can configure the number of IO threads required. With SWRSS is enabled and IO threads configured, reboot the vSRX for configuration to take effect. After IO threads are configured, distribute the GTP traffic to the configured IO threads according to TEID-based hash distribution for splitting a fat GTP session to multiple slim GTP sessions and process them on multiple cores in parallel.

NOTE: When PMI mode is enabled with TEID distribution and SWRSS support, performance of PMI is improved. If you want to enable PMI mode then run the **set securtiy flow power-mode-ipsec** command.

The following steps provide you details on how to enable SWRSS, configure IO threads, enable PMI mode for GTP sessions with TEID distribution for obtaining asymmetric fat tunnels:

1. SWRSS is enabled by default when NICs do not have sufficient number of queues as flow thread (FLT) based on vSRX type, then Software RSS (SWRSS) is enabled by flowd process. But, when SWRSS is not enabled use the following CLIs to enable. When you run this command SWRSS will be be enabled by force regardless of the NIC RSS or number of vCPUs.

   Enable SWRSS.

```
[edit]
user@host# set security forwarding-options receive-side-scaling software-rss mode enable
```

2. Configure the number of IO threads required. In this configuration we are configuring eight IO threads. The assigned number of vCPUs would be assigned for IO threads, and the rest vCPUs would be assigned for flow thread.

```
[edit]
user@host# set security forwarding-options receive-side-scaling software-rss io-thread-number 8
```

3.

```
[edit security]
user@host# set flow power-mode-ipsec
```

4. Configure GTP-U session distribution.

```
[edit security]
user@host# set forwarding-process application-services enable-gtpu-distribution
```

5. From the configuration mode, confirm your configuration by entering the **show** command.

```
[edit security]
user@host# show
forwarding-options {
    receive-side-scaling {
        software-rss {
             mode enable;
        io-thread-number 8;
flow {
    power-mode-ipsec;
}
forwarding-process {
    application-services {
        enable-gtpu-distribution;
    }
}
```

From the operational mode run the following command to view the actual number of vCPUs assigned to IO/flow threads.

```
show security forward-options resource-manager settings
```

```
------------------------------------------------------------------------
Owner          Type                      Current settings      Next settings
SWRSS-IO       CPU core number           2                     2
SWRSS          SWRSS mode                Enable                Enable
```

6. Commit the configuration.

```
[edit security]
user@host# commit
```

7. Reboot the vSRX for the configuration to take effect. After rebooting the whole device, PFE would check the IO-thread value according to the NIC RSS queue and its memory.