JUNIPER NETWORKS | Engineering Simplicity

Juniper Networks OpenStack Neutron Plug-in

Published
2021-07-01

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

Juniper Networks OpenStack Neutron Plug-in

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

YEAR 2000 NOTICE

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

END USER LICENSE AGREEMENT

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

Table of Contents

About This Guide

Use this guide to configure OpenStack Neutron plug-ins, which enable integration and orchestration of EX, MX, QFX, and SRX devices in the customer network.

CHAPTER 1

Overview

What's New in Juniper OpenStack Neutron Plug-in

The following features are introduced for the OpenStack Neutron plug-in version 3.0 release:

- VPN-as-a-Service (VPNaaS) support
- Virtual Extensible LAN L3 Routing with Ethernet VPN (VXLAN L3 Routing with EVPN)
- EVPN Multi-homing
- EVPN Bare Metal Server (BMS) support
- Source Network Address Translation (SNAT) and Destination Network Address Translation (DNAT) support

RELATED DOCUMENTATION

Configuring the VPN-as-a-Service (VPNaaS) Plug-in | 38

Configuring EVPN Multi-homing | 49

Configuring EVPN BMS | 52

Configuring Network Address Translation | 55

Overview of the OpenStack Neutron Plug-in

IN THIS SECTION

OpenStack is a cloud operating system, which builds public, private, and hybrid clouds by using commodity hardware. It offers high performance and throughput. Various network vendors who are specialized in networking gear have utilized the plug-in mechanism offered by Neutron and have moved out the L2, L3, Firewall, VPN, and Load balancing services on to their respective networking devices.

Juniper Networks provides OpenStack Neutron plug-ins, which enable integration and orchestration of EX, MX, QFX, and SRX devices in the customer network.

NOTE: The Openstack Neutron plug-in has not been tested in a nested virtualization environment.

Types of OpenStack Neutron Plug-in

Neutron plug-ins are categorized as follows:

- Core plug-ins - Implement the core API of the neutron, which consists of networking building blocks such as port, subnet, and network.

  Juniper provides the following core plug-ins:

  - ML2 VLAN plug-in
  - ML2 VXLAN plug-in with EVPN

- Service plug-ins - Implement the neutron API extensions such as L3 router and L4-L7 services such as FWaaS, LBaaS, and VPNaaS.

  Juniper provides the following service plug-ins:

  - Juniper L3 plug-in
  - FWaaS plug-in
  - VPNaaS plug-in

RELATED DOCUMENTATION

Configuring the ML2 Plug-in | 21
Configuring the L3 Plug-in | 28
Configuring the Firewall-as-a-Service (FWaaS) Plug-in | 31
Configuring the VPN-as-a-Service (VPNaaS) Plug-in | 38
Configuring OpenStack Extension for Physical Topology | 40
Configuring the Static Route Extension | 47
Configuring EVPN Multi-homing | 49
Configuring EVPN BMS | 52
Configuring Network Address Translation | 55

System Requirements

To use the Juniper Neutron plug-ins, the following are the system requirements:

- OpenStack Releases - Liberty and Mitaka.
- Operating Systems - Ubuntu 14 and Centos 7
- Juniper Network Devices - EX, QFX, SRX, and vSRX series
- Python version 2.7
- External Libraries - ncclient python library

2

CHAPTER

Installing and Configuring the OpenStack Neurton Plug-in

Installing the OpenStack Neutron Plug-in

IN THIS SECTION

Juniper plug-in binaries are available as RPM for CentOS and as Deb packages for Ubuntu. The plug-in must be installed only on the OpenStack Controller Node, where the OpenStack Neutron server is running.

Complete the following steps to install plug-ins on the appropriate operating system:

1. Download the package from https://www.juniper.net/support/downloads/?p=qpluginopen#sw.
2. Extract the binaries using the **tar -xvf juniper_plugins_*version*.tgz** command. The extracted folder contains the packages for Centos and Ubuntu.

The plug-ins are provided as a set of packages. All Neutron drivers and service plug-ins are provided as a single package and can be installed as follows.

- **CentOS**

```
rpm -ivh juniper_plugins_version/CentOS/ neutron-plugin-juniper-
version.noarch.rpm
```

- **Ubuntu**

```
sudo dpkg -i juniper_plugins_version/Ubuntu/python-neutron-plugin-
juniper_version_all.deb
```

Other software packages provide features such as Horizon GUI extensions, physical topology plug-in, and a Neutron client extension to support physical topology APIs. You can install these software packages in a similar manner.

The GUI packages and OpenStack Neutron client extension must be installed on the server that hosts Horizon.

The downloaded package also has an installation script, **install.sh**. This script is used to install the required plugins.

Installation Scripts for Juniper Plug-in

Installation script for Juniper OpenStack extensions is an interactive tool that explains the installation of Juniper OpenStack Neutron plug-ins.

Before running the installation script, ensure the following pre-requisites are met:

- Password-less SSH authentication is enabled between controller and all other nodes of OpenStack.
- keystonerc_admin is present in the home directory.
- Installation is done from the controller node on which Neutron server is running.

Table 1 on page 7 shows the available Juniper plug-in packages:

Table 1: Juniper plug-in packages

Plug-in Package	Function
Horizon physical topology plug-in	Provides Physical Networks dashboard for Administrator.
Horizon static route plug-in	Provides Static Route with preference dashboard.
Horizon BMS plug-in	Provides BMS dashboard.
Neutron plug-in	Provides Neutron ML2 extension and service plug-ins.
Neutron FWaaS plug-in	OpenStack Neutron plug-in for FWaaS. It supports both SRX and vSRX devices.
Neutron VPNaaS plug-in	OpenStack Neutron plug-in for VPNaaS. It supports both SRX and vSRX devices.
Neutron-client plug-in	Provides Neutron CLI for physical topology

The Juniper plug-in packages are classified into following categories based on the functionality of the package:

- Neutron server plug-in packages - Neutron plug-in, Neutron FWaaS plug-in, and Neutron VPNaaS plug-in

  OpenStack Neutron server plug-in packages are installed on OpenStack controller node, on which the Neutron server runs.

- User Interface packages - Horizon extensions and CLI extensions

  User Interface packages are installed on the node running Horizon, and CLI package is installed on any node on which the Neutron-client is installed.

When you are prompted during the installation, enter the required information and install the plug-in appropriately.

Setting Up L2 and L3 Topologies

IN THIS SECTION

Figure 1 on page 9 illustrates the L2 topology.

Figure 1: L2 Topology

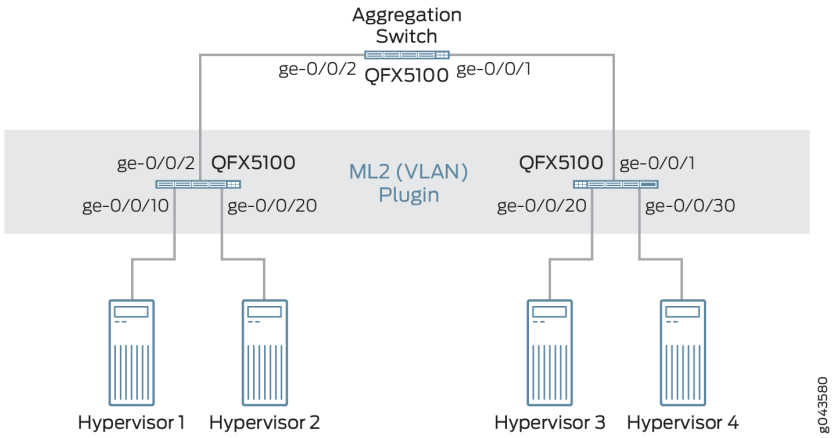


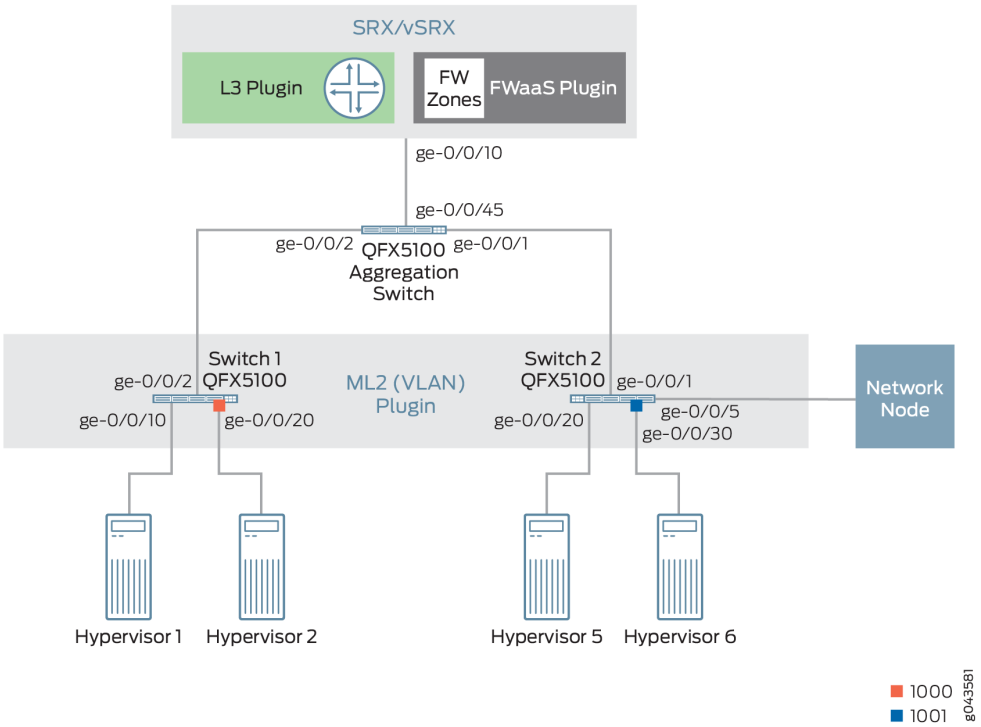
Figure 2 on page 10 shows switches such as Switch1, Switch2, and Aggregation Switch, which you can configure to set up the topology.

To set up a topology, complete the following steps:

1. On the Aggregation Switch, set ge-0/0/1 and ge-0/0/2 interfaces to Trunk All mode.
2. On Switch 1, set ge-0/0/2 interface to Trunk All mode and enable vlan-tagging on ge-0/0/10 and ge-0/0/20 interfaces.

3. On Switch 2, set ge-0/0/1 interface to Trunk All mode and enable vlan-tagging on ge-0/0/30 and ge-0/0/20 interfaces.

Figure 2: L3 Topology



NOTE: In Figure 2 on page 10, the SRX acts as both a router as well as a firewall.

Setting Up the Physical Topology

To set up the physical topology:

1. To configure the Juniper Networks plug-in with the reference physical topology, use the commands listed in Table 2 on page 11:

   Juniper Neutron plug-ins include CLI tools, which enable the administrator to define the network topology. The plug-ins depend on the topology definition to carry out network orchestration.

Table 2: CLI Tools

Name	Description
jnpr_device	Add device details
jnpr_nic_mapping	Add a mapping between physical network alias (ex: Physnet1) to the corresponding ethernet interface on the node.
jnpr_switchport_mapping	Add a mapping between the compute or network Node and its Ethernet Interface to the switch and the port that it is connected to.
jnpr_device_port	Define the downlink port of the router or firewall on which routed VLAN interface (RVI)RVI for each tenant VLAN is created.
jnpr_allocate_device	Define allocation of router and firewall to a tenant or group of tenants.
jnpr_vrrp_pool	Define the VRRP pool.

2. To add devices to the topology, run the following command on the OpenStack Neutron controller:

> **NOTE**: Use a login credential with super-user class privilege on the devices that are added to the topology.

```
admin@controller:~$ jnpr_device add -d device-name or device-IP-address -c
{switch, router, firewall} -u username -p device-password
```

3. To add and view switches that are added to the topology, run the following command on the OpenStack Neutron controller:

> **NOTE**: In the physical topology, Switch1 and Switch2 are connected to the hypervisors.

a. To add Switch1 to the topology, run the following command on the OpenStack Neutron controller::

```
admin@controller:~$ jnpr_device add -d switch1.juniper.net -c switch -u
root -p root-password
+---------------------+--------------+------------+---------+-------
+------+---------------+
|        Device       |       Ip     | Device Type |  model  | login |
vtep | vrrp_priority |
+---------------------+--------------+------------+---------+-------
+------+---------------+
| switch1.juniper.net | 10.107.52.136 |    switch   | qfx3500 |  root |
0   |       0       |
+---------------------+--------------+------------+---------+-------
+------+---------------+
```

b. To add Switch2 to the topology, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_device add -d switch2.juniper.net -c switch -u
root -p password
+---------------------+--------------+------------+---------+-------
+------+---------------+
|        Device       |       Ip     | Device Type |  model  | login |
vtep | vrrp_priority |
+---------------------+--------------+------------+---------+-------
+------+---------------+
| switch2.juniper.net | 10.107.52.137 |    switch   | qfx3500 |  root |
0   |       0       |
+---------------------+--------------+------------+---------+-------
+------+---------------+
```

c. To view the switches that are added to the topology, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_device list
+---------------------+--------------+------------+---------+-------
+------+---------------+
|        Device       |       Ip     | Device Type |  model  | login |
vtep | vrrp_priority |
```

```
+--------------------+--------------+------------+---------+-------
+------+---------------+
| switch1.juniper.net | 10.107.52.136 |    switch    | qfx3500 |  root |
0    |       0       |
| switch2.juniper.net | 10.107.52.137 |    switch    | qfx3500 |  root |
0    |       0       |
+--------------------+--------------+------------+---------+-------
+------+---------------+
```

4. To add routers to the topology, run the following command on the OpenStack Neutron controller:

> **NOTE**: In the physical topology shown in Figure 2 on page 10, the SRX acts as both a router as well as a firewall.

```
admin@controller:~$ jnpr_device add -d srx.juniper.net -c router -u root -p
password
+-----------------+--------------+------------+-------+-------+------
+---------------+
|      Device     |       Ip      | Device Type | model | login | vtep |
vrrp_priority |
+-----------------+--------------+------------+-------+-------+------
+---------------+
| srx.juniper.net | 10.107.23.103 |    router   |   srx  |  root |  0
|       0      |
+-----------------+--------------+------------+-------+-------+------
+---------------+
```

5. To add firewall to the topology, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_device add -d srx.juniper.net -c firewall -u root -p
password
+-----------------+--------------+------------+-------+-------+------
+---------------+
|      Device     |       Ip      | Device Type | model | login | vtep |
vrrp_priority |
+-----------------+--------------+------------+-------+-------+------
+---------------+
| srx.juniper.net | 10.107.23.103 |   firewall  |   srx  |  root |  0
|       0      |
| srx.juniper.net | 10.107.23.103 |    router   |   srx  |  root |  0
```

```
|       0       |
+-----------------+--------------+-------------+-------+-------+------
+---------------+
```

6. Define the NIC to physical network mapping for each hypervisor.

   In OpenStack, you generally define an alias for the physical network and its associated bridge by using the following configuration in **/etc/neutron/plugins/ml2/ml2_conf.ini** file on the network node and all the compute nodes:

```
[ovs]
 tenant_network_type = vlan
 bridge_mappings = physnet1:br-eth1
```

   Because you can connect the bridge br-eth1 to any physical interface, you must add the link between the bridge br-eth1 and the physical interface to the topology by entering following command:

```
admin@controller:~$ jnpr_nic-mapping add -H compute-hostname -b physical-
network-alias-name -n NIC
```

   a. To add Hypervisor 1 to the topology, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_nic_mapping add -H hypervisor1.juniper.net -b
physnet1 -n eth1
```

      Adding mapping

```
+---------------+------------+------+
|      Host     | BridgeName | Nic  |
+---------------+------------+------+
| 10.107.65.101 |  physnet1  | eth1 |
+---------------+------------+------+
```

   b. To add Hypervisor 2 to the topology, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_nic_mapping add -H hypervisor2.juniper.net -b
physnet1 -n eth1
```

Adding mapping

```
+---------------+------------+------+
|      Host     | BridgeName | Nic  |
+---------------+------------+------+
| 10.107.65.102 |  physnet1  | eth1 |
+---------------+------------+------+
```

c. To add Hypervisor 5 to the topology, run the following command on the OpenStack Neutron controller:

NOTE: Hypervisor 5 is mapped to physnet1-- br-eth1 -- eth2.

```
admin@controller:~$ jnpr_nic_mapping add -H hypervisor5.juniper.net -b
physnet1 -n eth2
```

Adding mapping

```
+---------------+------------+------+
|      Host     | BridgeName | Nic  |
+---------------+------------+------+
| 10.107.65.105 |  physnet1  | eth2 |
+---------------+------------+------+
```

d. To add Hypervisor 6 to the topology, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_nic_mapping add -H hypervisor6.juniper.net -b
physnet1 -n eth1
```

Adding mapping

```
+---------------+------------+------+
|      Host     | BridgeName | Nic  |
+---------------+------------+------+
```

```
| 10.107.65.106 |  physnet1  | eth1 |
+---------------+------------+------+
```

e. To add network node to the topology, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_nic_mapping add -H networknode.juniper.net -b
physnet1 -n eth1
```

Adding mapping

```
+---------------+------------+------+
|      Host     | BridgeName | Nic  |
+---------------+------------+------+
| 10.108.10.100 |  physnet1  | eth1 |
+---------------+------------+------+
```

f. To view all the mappings, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_nic_mapping list
+---------------+------------+------+
|      Host     | BridgeName | Nic  |
+---------------+------------+------+
| 10.107.65.101 |  physnet1  | eth1 |
| 10.107.65.102 |  physnet1  | eth1 |
| 10.107.65.105 |  physnet1  | eth2 |
| 10.107.65.106 |  physnet1  | eth1 |
| 10.108.10.100 |  physnet1  | eth1 |
+---------------+------------+------+
```

7. Define the mapping from the compute to the switch.

To configure the VLANs on the switches, the ML2 plug-in must determine the port of the switch on which the Hypervisor is connected through its Ethernet interface. This provides the plug-in an overall view of the topology between physnet1 -- br-eth1 -- eth1 -- Switch-x: ge-0/0/x. You can determine this information by either enabling LLDP, or by configuring it by using the following command:

```
admin@controller:~$ jnpr_switchport_mapping add -H compute-hostname -n NIC -s
switch-IP-address or switch-name -p switch-port
```

a. To map Hypervisor 1 to Switch 1, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_switchport_mapping add -H hypervisor1.juniper.net
-n eth1 -s switch1.juniper.net -p ge/0/0/10
Database updated with switch port binding
+---------------+------+---------------+-----------+-----------+
|      Host     | Nic  |     Switch    |    Port   | Aggregate |
+---------------+------+---------------+-----------+-----------+
| 10.107.65.101 | eth1 | 10.107.52.136 | ge/0/0/10 |           |
+---------------+------+---------------+-----------+-----------+
```

b. To map Hypervisor 2 to Switch 1, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_switchport_mapping add -H hypervisor2.juniper.net
-n eth1 -s switch1.juniper.net -p ge/0/0/20
Database updated with switch port binding
+---------------+------+---------------+-----------+-----------+
|      Host     | Nic  |     Switch    |    Port   | Aggregate |
+---------------+------+---------------+-----------+-----------+
| 10.107.65.102 | eth1 | 10.107.52.136 | ge/0/0/20 |           |
+---------------+------+---------------+-----------+-----------+
```

c. To map Hypervisor 5 to Switch 2, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_switchport_mapping add -H hypervisor5.juniper.net
-n eth2 -s switch2.juniper.net -p ge/0/0/20
```

Database updated with switch port binding

```
+---------------+------+---------------+-----------+-----------+
|      Host     | Nic  |     Switch    |    Port   | Aggregate |
+---------------+------+---------------+-----------+-----------+
| 10.107.65.105 | eth2 | 10.107.52.137 | ge/0/0/20 |           |
+---------------+------+---------------+-----------+-----------+
```

d. To map Hypervisor 6 to Switch 2, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_switchport_mapping add -H hypervisor6.juniper.net
-n eth1 -s switch2.juniper.net -p ge/0/0/30
```

Database updated with switch port binding

```
+---------------+------+---------------+-----------+-----------+
|      Host     | Nic  |     Switch    |    Port   | Aggregate |
+---------------+------+---------------+-----------+-----------+
| 10.107.65.106 | eth1 | 10.107.52.137 | ge/0/0/30 |           |
+---------------+------+---------------+-----------+-----------+
```

e. To map Network Node to Switch 2, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_switchport_mapping add -H networknode.juniper.net
-n eth1 -s switch2.juniper.net -p ge/0/0/5
```

Database updated with switch port binding

```
+---------------+------+---------------+----------+-----------+
|      Host     | Nic  |     Switch    |   Port   | Aggregate |
+---------------+------+---------------+----------+-----------+
| 10.108.10.100 | eth1 | 10.107.52.137 | ge/0/0/5 |           |
+---------------+------+---------------+----------+-----------+
```

f. To list all mappings, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_switchport_mapping list
+---------------+------+---------------+-----------+-----------+
|      Host     | Nic  |     Switch    |    Port   | Aggregate |
+---------------+------+---------------+-----------+-----------+
| 10.107.65.101 | eth1 | 10.107.52.136 | ge/0/0/10 |           |
| 10.107.65.102 | eth1 | 10.107.52.136 | ge/0/0/20 |           |
| 10.107.65.105 | eth2 | 10.107.52.137 | ge/0/0/20 |           |
| 10.107.65.106 | eth1 | 10.107.52.137 | ge/0/0/30 |           |
```

```
| 10.108.10.100 | eth1 | 10.107.52.137 |  ge/0/0/5 |           |
+---------------+------+---------------+-----------+-----------+
```

8. Define the downlink port on the SRX device (Router) on which the RVI is created by the plug-in.

   You can update the plug-in database with the port on the SRX device to which the Aggregation Switch is connected by using the following command:

```
admin@controller:~$ jnpr_device_port -d SRX-device-name or switch-IP -p srx-
port-name -t port_type: Downlink
```

   a. To add the downlink port of the SRX device to the topology, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_device_port add -d srx.juniper.net -p ge-0/0/10 -
t Downlink
+---------------+-----------+---------------+
|     Device    |    port   |   port_type   |
+---------------+-----------+---------------+
| 10.107.23.103 | ge-0/0/10 | downlink_port |
+---------------+-----------+---------------+
```

9. Create a VRRP pool

   The L3 plug-in supports high availability via VRRP. In order to use this functionality, you must create a VRRP pool and assign only one of the devices in the pool to a tenant using the **jnpr_allocate_device** command.

   Complete the following steps to create a VRRP pool:

   a. To add routers, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_device add -d 10.20.30.40 -c router -u root -p
password
+-----------------+---------------+-------------+-------+-------+------
+---------------+
|      Device     |       Ip      | Device Type | model | login | vtep |
vrrp_priority |
+-----------------+---------------+-------------+-------+-------+------
+---------------+
| 10.20.30.40     | 10.20.30.40   |    router   |  srx  |  root |  0
|       0       |
+-----------------+---------------+-------------+-------+-------+------
```

```
+---------------+
admin@controller:~$ jnpr_device add -d 10.20.30.41 -c router -u root -p
password
+-----------------+---------------+-------------+-------+-------+------
+---------------+
|      Device     |       Ip      | Device Type | model | login | vtep |
vrrp_priority |
+-----------------+---------------+-------------+-------+-------+------
+---------------+
| 10.20.30.41     | 10.20.30.41   |    router   |  srx  |  root |  0
|       0       |
+-----------------+---------------+-------------+-------+-------+------
+---------------+
```

b. To create VRRP pools, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_vrrp_pool add -d 10.20.30.40 -p tenant1_pool1
+----------------------------------+-----------------+
|            Device ID             |  VRRP POOL NAME |
+----------------------------------+-----------------+
| 10.20.30.40                      | tenant1_pool1   |
+----------------------------------+-----------------+

admin@controller:~$ jnpr_vrrp_pool add -d 10.20.30.41 -p tenant1_pool1
+----------------------------------+-----------------+
|            Device ID             |  VRRP POOL NAME |
+----------------------------------+-----------------+
| 10.20.30.41                      | tenant1_pool1   |
+----------------------------------+-----------------+

admin@controller:~$ jnpr_vrrp_pool list
+---------------+----------------+
|   Device ID   | VRRP POOL NAME |
+---------------+----------------+
| 10.20.30.40   | tenant1_pool1  |
| 10.20.30.41   | tenant1_pool1  |
+---------------+----------------+
```

c. To define allocation of devices to a tenant or a group of tenants, run the following command on the OpenStack Neutron controller:

```
admin@controller:~$ jnpr_allocate_device add -t tenant-project_id -d
device-hostname-or-IP-address
admin@controller:~$ jnpr_allocate_device add -t
e0d6c7d2e25943c1b4460a4f471c033f  -d 10.20.30.40
+----------------------------------+---------------+
|            Tenant ID             |   Device IP   |
+----------------------------------+---------------+
| e0d6c7d2e25943c1b4460a4f471c033f | 10.20.30.40   |
+----------------------------------+---------------+
```

To use a device as the default device for multiple tenants, run the following command on the OpenStack Neutron controller and set the tenant as **default**. For example:

```
admin@controller:~$ jnpr_allocate_device add -t default -d 10.20.30.40
+----------------------------------+---------------+
|            Tenant ID             |   Device IP   |
+----------------------------------+---------------+
|             default              | 10.20.30.40   |
+----------------------------------+---------------+
```

Configuring the ML2 Plug-in

IN THIS SECTION

Juniper ML2 drivers support the following types of virtual networks:

- VLAN based network
- VXLAN based tunneled network with EVPN by using Hierarchical Port Binding

By using the features such as LAG and MC-LAG, ML2 drivers support the orchestration of aggregated links that connect the OpenStack nodes to the ToR switches.

Configuring the ML2 VLAN Plug-in

Juniper ML2 VLAN plug-in supports configuring VLAN of the network of each tenant on the corresponding switch port attached to the compute node. VM migration is supported from OpenStack Neutron version 2.7.1 onwards.

- **Supported Devices**

  EX series and QFX series devices.

- **Plug-in Configuration**

  To configure OpenStack Neutron to use VLAN type driver:

  1. On the OpenStack Controller, open the file **/etc/neutron/neutron.conf** and update as follows:

     core_plugin = neutron.plugins.ml2.plugin.Ml2Plugin or **core_plugin = neutron.plugins.ml2.plugin_pt_ext.Ml2PluginPtExt**

  2. On Openstack Controller, update the ML2 configuration file **/etc/neutron/plugins/ml2/ml2_conf.ini** to set Juniper ML2 plug-in as the mechanism driver:

     ```
     [ml2]
     type_drivers = vlan
     mechanism_drivers = openvswitch,juniper
     tenant_network_types = vlan
     ```

  3. Specify the VLAN range and the physical network alias to be used.

     ```
     [ml2_type_vlan]
     network_vlan_ranges = physnet1:1001:1200
     ```

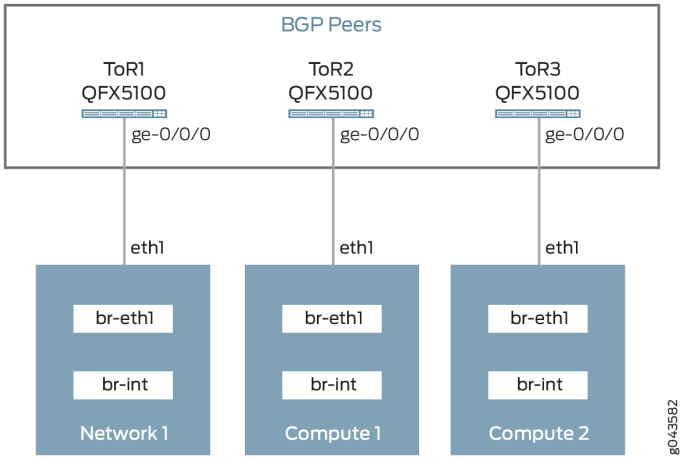
  4. Restart the OpenStack neutron server for the changes to take effect.

5. Login to OpenStack GUI and create VLAN based network and launch VMs. You can view the VLAN IDs of the OpenStack network are created on the switch and mapped to the interfaces that are configured through the **jnpr_switchport_mapping** command.

Configuring ML2 VXLAN Plug-in with EVPN

ML2 EVPN driver is based on Neutron hierarchical port binding design. It configures the ToR switches as VXLAN endpoints (VTEPs), which is used to extend VLAN based L2 domain across routed networks.

Figure 3: IP Fabric



To provide L2 connectivity between the network ports and compute nodes, the L2 packets are Tagged with VLANs and sent to the Top of Rack (TOR) switch. The VLANs used to tag the packets are only locally significant to the TOR switch (Switch Local VLAN).

At the ToR switch, the Switch Local VLAN is mapped into a global VXLAN ID. The L2 packets are encapsulated into VXLAN packets and sent to the virtual tunnel endpoint (VTEP) on the destination node, where they are de-encapsulated and sent to the destination VM.

To make the L2 connectivity between the endpoints work with VXLAN, each endpoint must be informed about the presence of destination VM and VTEP. EVPN uses a BGP-based control plane to learn this information. The plug-in assumes that the ToRs are setup with BGP-based peering.

Refer to Junos documentation for configuring BGP on the ToR switches available at BGP Configuration Overview.

Supported Devices

QFX 5100 only

Plug-in Configuration

To install the Juniper neutron plug-in on the neutron server node, complete the following steps:

1. Edit the ML2 configuration file **/etc/neutron/plug-ins/ml2/ml2_conf.ini** to add the following configuration for EVPN driver:

```
[ml2]
type_drivers = vlan,vxlan,evpn
tenant_network_types = vxlan
mechanism_drivers = jnpr_evpn,openvswitch

[ml2_type_vlan]
network_vlan_ranges=<ToR_MGMT_IP_SWITCH1>:<vlan-start>:<vlan-end>,
<ToR_MGMT_IP_SWITCH2>:<vlan-start>:<vlan-end>,<ToR_MGMT_IP_SWITCH3>:
<vlan-start>:<vlan-end>

[ml2_type_vxlan]
vni_ranges = <vni-start>:<vni-end>
```

2. Restart the OpenStack neutron server to load the EVPN ML2 driver.
3. Update the plug-in topology database. To add a ToR switch and two compute nodes connected to it, run the following commands on the neutron server node:

```
admin@controller:~$ jnpr_device add -d ToR1 -u root -p root-password -c switch
admin@controller:~$ jnpr_switchport_mapping add -H Compute1 -n eth1 -s
tor1_mgmt_ip-address -p ge-0/0/1
admin@controller:~$ jnpr_switchport_mapping add -H Compute2 -n eth1 -s
tor2_mgmt_ip-address -p ge-0/0/1
admin@controller:~$ jnpr_switchport_mapping add -H Network1 -n eth1 -s
tor3_mgmt_ip-address -p ge-0/0/1
```

4. Update OVS L2 agent on all compute and network nodes.

   All the compute and network nodes needs to be updated with the bridge mapping. The bridge name should be the IP address of the ToR switch.

5. Edit the file **/etc/neutron/plug-ins/ml2/ml2_conf.ini** (ubuntu) or **/etc/neutron/plug-ins/ml2/openvswitch_agent.ini** (centos) to add the following:

```
[ovs]
bridge_mappings = <ToR_MGMT_IP>:br-eth1
```

Here br-eth1 is an OVS bridge which enslaves the eth1 physical port on the OpenStack node connected to ToR1. It provides the physical network connectivity for tenant networks.

6. Restart the OVS agent on all the compute and network nodes.
7. Login to OpenStack GUI to create a virtual network and launch VMs. You can view the VXLAN IDs of the OpenStack network, where the switch local VLANs are created on the switch and mapped to the VXLAN ID on each ToR.

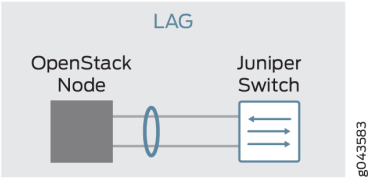
Configuring ML2 Driver with Link Aggregation

You can use Link Aggregation (LAG) between an OpenStack compute node and a Juniper switch to improve network resiliency.

Plug-in Configuration

Figure 4 on page 25 describes the connectivity of OpenStack compute node to the Top of Rack (TOR) switch.

Figure 4: LAG



To configure LAG on Juniper ToR switches, refer to the following link:

Configuring Aggregated Ethernet Links (CLI Procedure)

To configure LAG on the OpenStack compute node:

1. Connect the data ports of the OpenStack networking to the LAG interface on the Juniper switches. For example, connect eth1 and eth2 for LAG as follows:

```
admin@controller:~$ ovs-vsctl add-bond br-eth1 bond0 eth1 eth2 lacp=active
```

2. Create NIC mapping:

```
admin@controller:~$ jnpr_nic_mapping add -H openstack-node-name-or-ip-address
-b physnet1 -n nic
```

```
admin@controller:~$ jnpr_nic_mapping add -H openstack-node-name-or-ip-address
-b physnet1 -n nic
admin@controller:~$ jnpr_nic_mapping add -H 10.207.67.144 -b physnet1 -n eth1
admin@controller:~$ jnpr_nic_mapping add -H 10.207.67.144 -b physnet1 -n eth2
```

3. Add switch port mapping with aggregate interface:

```
admin@controller:~$ jnpr_switchport_mapping add -H openstack-node-name-or-ip-
address -n eth1 -s switch-name-or-ip-address -p port -a lag-name
admin@controller:~$ jnpr_switchport_mapping add -H openstack-node-name-or-ip-
address -n eth2 -s switch-name-or-ip-address -p port -a lag-name
admin@controller:~$ jnpr_switchport_mapping add -H 10.207.67.144 -n eth1 -s
dc-nm-qfx3500-b -p ge-0/0/2 -a ae0
admin@controller:~$ jnpr_switchport_mapping add -H 10.207.67.144 -n eth2 -s
dc-nm-qfx3500-b -p ge-0/0/3 -a ae0
```

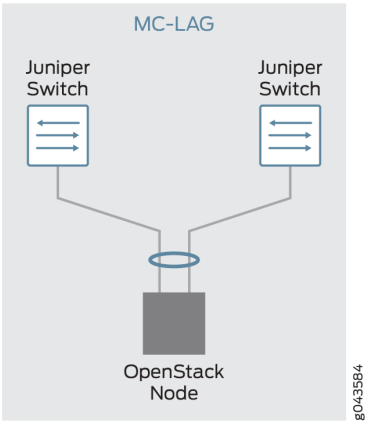
4. Verify switch port mapping with aggregate details:

```
admin@controller:~$ jnpr_switchport_mapping list
+---------------+--------+-----------------+-----------+-----------+
| Host          | Nic    | Switch          | Port      | Aggregate |
+---------------+--------+-----------------+-----------+-----------+
| 10.207.67.144 | eth1   | dc-nm-qfx3500-b | ge-0/0/2  | ae0       |
| 10.207.67.144 | eth2   | dc-nm-qfx3500-b | ge-0/0/3  | ae0       |
+---------------+--------+-----------------+-----------+-----------+
```

Configuring ML2 Driver with Multi-Chassis Link Aggregation

You can configure Multi-Chassis Link Aggregation (MC-LAG) and use it with Juniper Neutron plug-ins.

Figure 5: MC-LAG



Plugin Configuration

To configure MC-LAG on Juniper switches, refer to the following link:

Configuring Multichassis Link Aggregation on EX Series Switches

To configure LAG on Openstack compute:

1. Connect the data ports of the OpenStack networking to the LAG interface on two different Juniper switches. For example, connect eth1 and eth2 for LAG as follows:

```
admin@controller:~$ ovs-vsctl add-bond br-eth1 bond0 eth1 eth2 lacp=active
```

2. Create NIC mapping:

```
admin@controller:~$ jnpr_nic_mapping add -H openstack-node-name-or-ip-address
-b physnet1 -n nic
admin@controller:~$ jnpr_nic_mapping add -H openstack-node-name-or-ip-address
-b physnet1 -n nic
admin@controller:~$ jnpr_nic_mapping add -H 10.207.67.144 -b physnet1 -n eth1
admin@controller:~$ jnpr_nic_mapping add -H 10.207.67.144 -b physnet1 -n eth2
```

3. Add switch port mapping with aggregate interface on the OpenStack controller:

```
admin@controller:~$ jnpr_switchport_mapping add -H openstack-node-name-or-ip-address -n eth1 -s switch-name-or-ip-address -p port -a lag-name
admin@controller:~$ jnpr_switchport_mapping add -H openstack-node-name-or-ip-address -n eth2 -s switch-name-or-ip-address -p port -a lag-name
admin@controller:~$ jnpr_switchport_mapping add -H 10.207.67.144 -n eth1 -s dc-nm-qfx3500-b -p ge-0/0/2 -a ae0
admin@controller:~$ jnpr_switchport_mapping add -H 10.207.67.144 -n eth2 -s dc-nm-qfx3500-b -p ge-0/0/3 -a ae0
```

4. Verify switch port mapping with aggregate details:

```
admin@controller:~$ jnpr_switchport_mapping list
+---------------+--------+-----------------+-----------+-----------+
| Host          | Nic    | Switch          | Port      | Aggregate |
+---------------+--------+-----------------+-----------+-----------+
| 10.207.67.144 | eth1   | dc-nm-qfx3500-a | ge-0/0/2  | ae0       |
| 10.207.67.144 | eth2   | dc-nm-qfx3500-b | ge-0/0/3  | ae0       |
+---------------+--------+-----------------+-----------+-----------+
```

Configuring the L3 Plug-in

IN THIS SECTION

Juniper L3 plug-in supports the following features:

- L3 routing
- Adding static routes
- Provides router High Availability via VRRP
- Routed external networks

Supported Devices

EX, QFX, SRX, and vSRX series devices.

Plug-in Configuration

To configure the L3 plug-in:

1. Update the configuration file **/etc/neutron/neutron.conf** as follows:

```
[DEFAULT]
service_plugins = neutron.services.juniper_l3_router.dmi.l3.JuniperL3Plugin
```

2. Restart neutron-server for the changes the take effect.

3. Login to OpenStack GUI, create networks, and assign to a router. On the device configured as router, you can view the routing instance and the routing virtual interface (RVI) corresponding to the networks assigned to the routing instance.

Configuring the L3 VXLAN Plug-in with EVPN

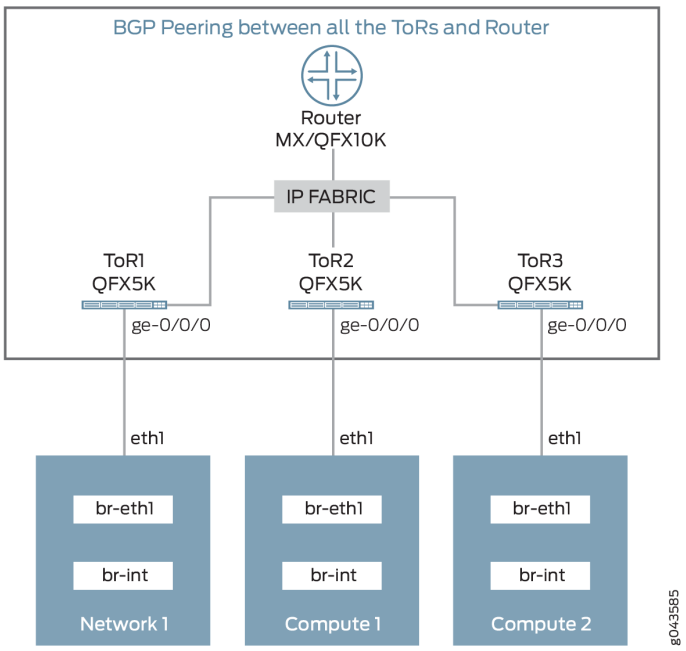
The Juniper VXLAN EVPN ML2 plug-in uses VXLAN tunnels along with the Neutron hierarchal port binding design to provide L2 networks in OpenStack.

The default L3 service plug-in present in OpenStack implements virtual router using Linux network name spaces.

This release of Neutron Plug-ins from Juniper Networks adds support for L3 routing for VXLAN networks. This is done by creating a VTEP on MX and QFX10000 series devices to convert the VXLAN to VLAN based network and configuring routing Instances to route packets between these VLANs.

This feature works in conjunction with Juniper Networks VXLAN EVPN ML2 plug-in, whereas the VXLAN EVPN ML2 plug-in is used to provide L2 connectivity, the VXLAN EVPN L3 service plug-in provides L3 routing between the VXLAN based virtual networks.

Figure 6: BGP Peering between all the TORs and Router



Supported Devices

The VXLAN EVPN L3 service plug-in can orchestrate MX and QFX10K devices to provide L3 based routing between VLAN based networks.

EVPN is supported on version 14.2R6 on MX and QFX10K.

Plug-in Configuration

The EVPN L3 plug-in depends on the VXLAN EVPN ML2 plug-in for orchestration of layer 2 ports. Configuration of ML2 VXLAN EVPN plug-in is a pre-requisite for this plug-in. To configure the ML2 VXLAN EVPN plug-in , refer to Configuring ML2 VXLAN Plug-in with EVPN.

To configure the VXLAN EVPN L3:

1. Edit the **/etc/neutron/neutron.conf file** and update the service plug-in definition as follows:

```
[DEFAULT]
...
```

```
service_plugins = neutron.services.juniper_l3_router.evpn.l3.JuniperL3Plugin
…
```

2. Add the QFX10K or MX device as a physical router to the plug-ins topology database:

```
admin@controller:~$ jnpr_device add -d QFX10K/MX IP -c router -u root -p
root_password -t vtep_ip
```

3. Update the VLAN allocation range in **/etc/neutron/plugins/ml2/ml2_conf.ini** file and add the per device VLAN range for the physical router. You can do this by adding the IP address and the VLAN range of the routing device followed by the VLAN range as follows:

```
[ml2_type_vlan]


network_vlan_ranges = <ToR_MGMT_IP_SWITCH1>:vlan-start:vlan-end  …,
…,<MGMT_IP_ROUTER>:vlan-start:vlan-end
```

Make sure that the VXLAN range is configured with the correct VNI range.

```
[ml2_type_vxlan]


tunnel_id_ranges = vxlan-start:vxlan-end
```

4. Verify that the EVPN L3 plug-in is functioning properly. Restart the neutron server and create a virtual router by using the Horizon dashboard or CLI. It creates a routing Instance on the configured routing device, QFX10K/MX.

Configuring the Firewall-as-a-Service (FWaaS) Plug-in

IN THIS SECTION

Juniper Networks Firewall-as-a-Service (FWaaS) plug-in builds on top of Juniper ML2 and L3 plug-ins. It enables Neutron to configure firewall rules and policies on SRX and vSRX devices. In OpenStack, a tenant can create a firewall and assign a security policy to it. A security policy is a collection of firewall rules. Figure 7 on page 32 illustrates the relationship of firewall rules, firewall policy, and firewall.

Figure 7: Firewall Policy

Firewall Rule - Defines the source address and port(s), destination address and port(s), protocol and the action to be taken on the matching traffic.

Firewall Policy - Collection of firewall rules

Firewall - Represents a firewall device.

When you enable a FwaaS plug-in, the SRX or vSRX should act as a router as well as a firewall. The administrator must ensure this while setting up the topology.

Supported Devices

SRX and vSRX series devices

Plug-in Configuration

NOTE: Before proceeding further, ensure that the following pre-requisites are met:

- Topology is set up
  - devices is added to jnpr_devices table
  - compute nic → physical network alias mapping is added to jnpr_nic_mapping table.
  - Compute → Switch connectivity is captured in jnpr_switchport_mapping table (needed for L2 VLAN orchestration)

- L2 plug-in is set up. This is optional if you are using a 3rd party ML2 plug-in.
- L3 plug-in is set up to use the SRX/vSRX as the router.

To configure Neutron to use Juniper FwaaS service plug-in:

1. Update the Neutron configuration file **/etc/neutron/neutron.conf** file and append service_plug-ins with the following:

```
service_plug-ins =
 neutron.services.juniper_l3_router.dmi.l3.JuniperL3Plugin,
neutron_fwaas.services.juniper_fwaas.dmi.fwaas.JuniperFwaaS
```

2. Add firewall to the topology:

```
admin@controller:~$ jnpr_device add -d dns-name-or-device-ip-address -c
firewall -u root-user -p root_password
```

3. Define the downlink trunk port on the SRX device on which the RVIs are created by the plug-in.

   Update the plug-in database with the port on the SRX device to which the Aggregation Switch is connected:

```
admin@controller:~$ jnpr_device_port -d srx-device-name-or-switch-ip-address -
p port-on-the-srx -t port-type
```

   For example:

```
admin@controller:~$ jnpr_device_port add -d srx1 -p ge-0/0/1 -t Downlink
```

4. Allocate the firewall to a tenant or as a default for all tenants:

```
admin@controller:~$ jnpr_allocate_device add -t project_id -d SRX/vSRX ip
```

To allocate the firewall as a default to all the tenants that do not have a firewall allocated to them, use the below command:

```
admin@controller:~$ jnpr_allocate_device add -t default -d SRX/vSRX ip
```

5. Enable Horizon to show Firewall panel.

   To display the Firewall panel under the Networks group in the Horizon user interface, open **/usr/share/openstack-dashboard/openstack_dashboard/local/local_settings.py**, and update the following configuration:

   enable_firewall: True

6. After completing the FWaaS plug-in configuration, restart the following:

   - Neutron-Server
     - Ubuntu – service neutron-server restart
     - CentOS – systemctl restart neutron-server
   - Apache (restarts Horizon)
     - Ubuntu – service apache2 restart
     - CentOS – systemctl restart httpd

7. From the Horizon GUI, create firewall rules and associate them to a policy. Create a firewall and assign the routers and the firewall policy to it.

8. On the SRX, you can verify a firewall zone for each routing instance and the corresponding policies pushed within the zones.

Configuring a Dedicated Perimeter Firewall

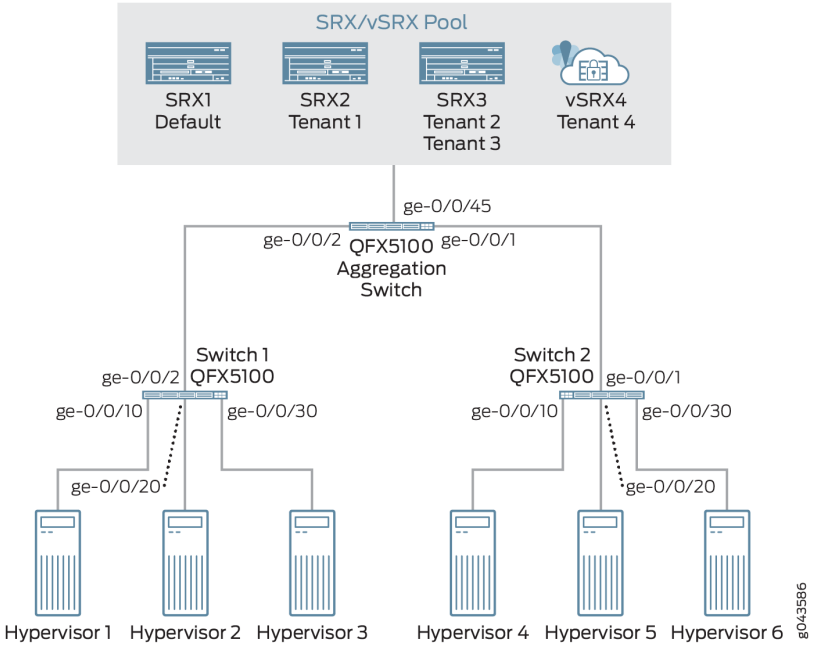
Requirement of each tenant varies according to the performance and cost of the firewall. For example, a dedicated firewall for better performance and compliance, a lower cost firewall enabled by sharing network resources, or a firewall with a complete administrative access to the networking device so as to leverage the advanced services provided by the device. To meet different requirements of each tenant, the cloud provider must have the provision to allocate dedicated or shared network resources to the tenants.

By using Juniper Networks FwaaS plug-in, a service provider can allocate dedicated or shared resources (physical or virtual) to the tenants.

For example, the service provider can create and configure firewall as follows according to the requirement of the tenant:

- Economy - Allocates a shared SRX or vSRX for a group of tenants
- Silver - Allocates a dedicated SRX or vSRX per tenant with default specifications
- Gold - Allocates a high-end SRX or vSRX

Figure 8: Firewall Allocation



As seen in the Figure 8 on page 35, you can dedicate SRX/vSRX to a tenant or a group of tenants. This procedure is transparent to the tenant and is done using the supplied CLI tools along with Juniper OpenStack Neutron plug-in.

To allocate a dedicated SRX cluster to a tenant:

1. Allocate the primary SRX device to the tenant:

```
admin@controller:~$ jnpr_allocate_device add -t tenant-project-id -d hostname-
or-device-ip-address
```

```
admin@controller:~$ jnpr_allocate_device add -t
e0d6c7d2e25943c1b4460a4f471c033f  -d 10.20.30.40
```

```
+----------------------------------+---------------+
|            Tenant ID             |   Device IP   |
+----------------------------------+---------------+
| e0d6c7d2e25943c1b4460a4f471c033f | 10.20.30.40   |
+----------------------------------+---------------+
```

2. Define the VRRP cluster and assign a name:

```
admin@controller:~$ jnpr_vrrp_pool add -d hostname-or-device-ip-address -p
pool-name-to-be-assigned
admin@controller:~$ jnpr_vrrp_pool add -d hostname-or-device-ip-address -p
pool-name-to-be-assigned
```

```
admin@controller:~$ jnpr_vrrp_pool add -d 10.20.30.40 -p tenant1_pool1
+----------------------------------+-----------------+
|            Device ID             |   VRRP POOL NAME|
+----------------------------------+-----------------+
| 10.20.30.40                      | tenant1_pool1   |
+----------------------------------+-----------------+

admin@controller:~$ jnpr_vrrp_pool add -d 10.20.30.41 -p tenant1_pool1
+----------------------------------+-----------------+
|            Device ID             |   VRRP POOL NAME|
+----------------------------------+-----------------+
| 10.20.30.40                      | tenant1_pool1    |
+----------------------------------+-----------------+

admin@controller:~$ jnpr_vrrp_pool list
+---------------+----------------+
|   Device ID   | VRRP POOL NAME |
+---------------+----------------+
| 10.20.30.40   | tenant1_pool1  |
| 10.20.30.41   | tenant1_pool1  |
+---------------+----------------+
```

Configuring High Availability and Virtual Router Redundancy Protocol

FwaaS plug-in supports High Availability with Virtual Router Redundancy Protocol (HA with VRRP). In order to use this functionality, you must create a VRRP pool and assign one of the devices in the pool to a tenant by using the **jnpr_allocate_device** command.

To create a VRRP pool and to assign a device to a tenant:

1. Create a VRRP pool:

```
admin@controller:~$ jnpr_vrrp_pool add -d 10.20.30.40 -p tenant1_pool1
+----------------------------------+------------------+
|            Device ID             |   VRRP POOL NAME|
+----------------------------------+------------------+
| 10.20.30.40                      | tenant1_pool1    |
+----------------------------------+------------------+

admin@controller:~$ jnpr_vrrp_pool add -d 10.20.30.41 -p tenant1_pool1
+----------------------------------+------------------+
|            Device ID             |   VRRP POOL NAME|
+----------------------------------+------------------+
| 10.20.30.41                      | tenant1_pool1    |
+----------------------------------+------------------+

admin@controller:~$ jnpr_vrrp_pool list
+---------------+----------------+
|   Device ID   | VRRP POOL NAME |
+---------------+----------------+
| 10.20.30.40   | tenant1_pool1  |
| 10.20.30.41   | tenant1_pool1  |
+---------------+----------------+
```

2. Allocate the primary SRX device of the VRRP pool to the tenant:

```
admin@controller:~$ jnpr_allocate_device add -t tenant-project-id -d hostname-
or-device-ip-address
```

```
admin@controller:~$ jnpr_allocate_device add -t
e0d6c7d2e25943c1b4460a4f471c033f  -d 10.20.30.40
```

```
+----------------------------------+---------------+
|            Tenant ID             |   Device IP   |
+----------------------------------+---------------+
| e0d6c7d2e25943c1b4460a4f471c033f | 10.20.30.40   |
+----------------------------------+---------------+
```

Configuring the VPN-as-a-Service (VPNaaS) Plug-in

Juniper Networks VPN-as-a-Service (VPNaaS) builds on top of the Juniper Networks L3 and FWaaS plug-ins. Use the VPNaaS plug-in to configure site-to-site VPN on SRX and vSRX devices.

Supported Devices

SRX and vSRX series devices

Plug-in Configuration

Before you proceed, ensure that the following pre-requisites are met:

- Topology is set up:
  - Devices are added to jnpr_devices table.
  - Compute NIC – Physical network alias mapping is added to jnpr_nic_mapping table.
  - Compute – Switch connectivity is captured in jnpr_switchport_mapping table, which is needed for L2 VLAN orchestration.
- L2 plug-in is setup. This is optional if you are using a 3rd party ML2 plug-in.
- L3 plug-in is setup to use the SRX/vSRX as the router.
- FwaaS plug-in is setup (optional).

To configure the OpenStack Neutron to use Juniper Networks VPNaaS service plug-in:

1. Update the Neutron configuration file **/etc/neutron/neutron.conf** file and append service_plug-ins with the following:

```
service_plug-ins =
 neutron.services.juniper_l3_router.dmi.l3.JuniperL3Plugin,
neutron_fwaas.services.juniper_fwaas.dmi.fwaas.JuniperFwaaS,
neutron_vpnaas.services.vpn.juniper.vpnaas.JuniperVPNaaS
```

NOTE: The following steps are optional if the FWaaS plug-in is already configured.

2. Add a firewall to the topology:

```
admin@controller:~$ jnpr_device add -d device-name -c firewall -u root-user -
p root-password
```

3. Define the downlink trunk port on the SRX device on which the RVIs are created by the plug-in.

   Update the plug-in database with the port on the SRX device to which the Aggregation Switch is connected:

```
admin@controller:~$ jnpr_device_port -d srx-device-name-or-switch-ip-address -
p port-on-the-srx -t port-type
```

   For example:

```
admin@controller:~$ jnpr_device_port add -d srx1 -p ge-0/0/1 -t Downlink
```

4. Allocate the firewall to a tenant or as a default for all tenants:

```
admin@controller:~$ jnpr_allocate_device add -t project-id -d srx-or-vsrx-ip-
address
```

   To allocate the firewall as a default to all the tenants who do not have a firewall allocated to them:

```
admin@controller:~$ jnpr_allocate_device add -t default -d srx-or-vsrx-ip-
address
```

5. After completing the FWaaS plug-in configuration, restart the following:

   - Neutron-Server
     - Ubuntu - service neutron-server restart
     - CentOS - systemctl restart neutron-server
   - Apache (restarts Horizon)
     - Ubuntu - service apache2 restart
     - CentOS - systemctl restart httpd

6. From the Horizon GUI, create a VPN IPSEC site connection along with its associated IKE, IPSEC and VPNService components. You can view the corresponding IKE, IPSEC, IKE GW configurations that are activated on the SRX/vSRX.

Configuring OpenStack Extension for Physical Topology

Physical Topology extension provides a dashboard for the OpenStack administrator to manage physical network connections. For example, Host NIC to Switch Port mapping. The Physical topology API exposes these physical network connections.

Juniper neutron plug-in currently manages topology information by using the **jnpr_switchport_mapping** command.

```
admin@controller:~$ jnpr_switchport_mapping list
+---------------+------+---------------+-----------+-----------+
|      Host     | Nic  |     Switch    |    Port   | Aggregate |
+---------------+------+---------------+-----------+-----------+
| 10.107.65.101 | eth1 | 10.107.52.136 | ge/0/0/10 |           |
| 10.107.65.102 | eth1 | 10.107.52.136 | ge/0/0/20 |           |
| 10.107.65.105 | eth2 | 10.107.52.137 | ge/0/0/20 |           |
| 10.107.65.106 | eth1 | 10.107.52.137 | ge/0/0/30 |           |
| 10.108.10.100 | eth1 | 10.107.52.137 |  ge/0/0/5 |           |
+---------------+------+---------------+-----------+-----------+
```

Plug-in Configuration

To configure the physical topology extension:

1. On OpenStack Controller, update the ML2 configuration file **etc/neutron/plugins/ml2/ml2_conf.ini** as follows:

   NOTE: If the file is updated to enable EVPN driver for VXLAN, skip this step.

```
[ml2]
type_drivers = vlan
```

```
tenant_network_types = vxlan
mechanism_drivers = openvswitch,juniper
```

2. Update the Neutron configuration file **/etc/neutron/neutron.conf** as follows:

   core_plugin = neutron.plugins.ml2.plugin_pt_ext.Ml2PluginPtExt

3. Enable Physical topology dashboard as follows:

   - CentOS

     cp /usr/lib/python2.7/site-packages/juniper_horizon_physical-topology/

     openstack_dasboard/enabled/_2102_admin_topology_panel.py

     /usr/share/openstack-dashboard/openstack_dashboard/enabled/

   - Ubuntu

     cp /usr/lib/python2.7/dist-packages/juniper_horizon_physical-topology/

     openstack_dasboard/enabled/_2102_admin_topology_panel.py

     /usr/share/openstack-dashboard/openstack_dashboard/enabled/

4. Restart Neutron and Horizon services:

   - Neutron-Server
     - Ubuntu - service neutron-server restart
     - CentOS - systemctl restart neutron-server
   - Apache (restarts Horizon)
     - Ubuntu - service apache2 restart
     - CentOS - systemctl restart httpd

5. After the installation is complete, the physical topology dashboard is available at **Admin > System > Physical Networks**.

   From the Physical Networks dashboard, you can perform the following tasks:.

Figure 9 on page 42 shows how to view physical topologies.

Figure 9: View Physical topologies

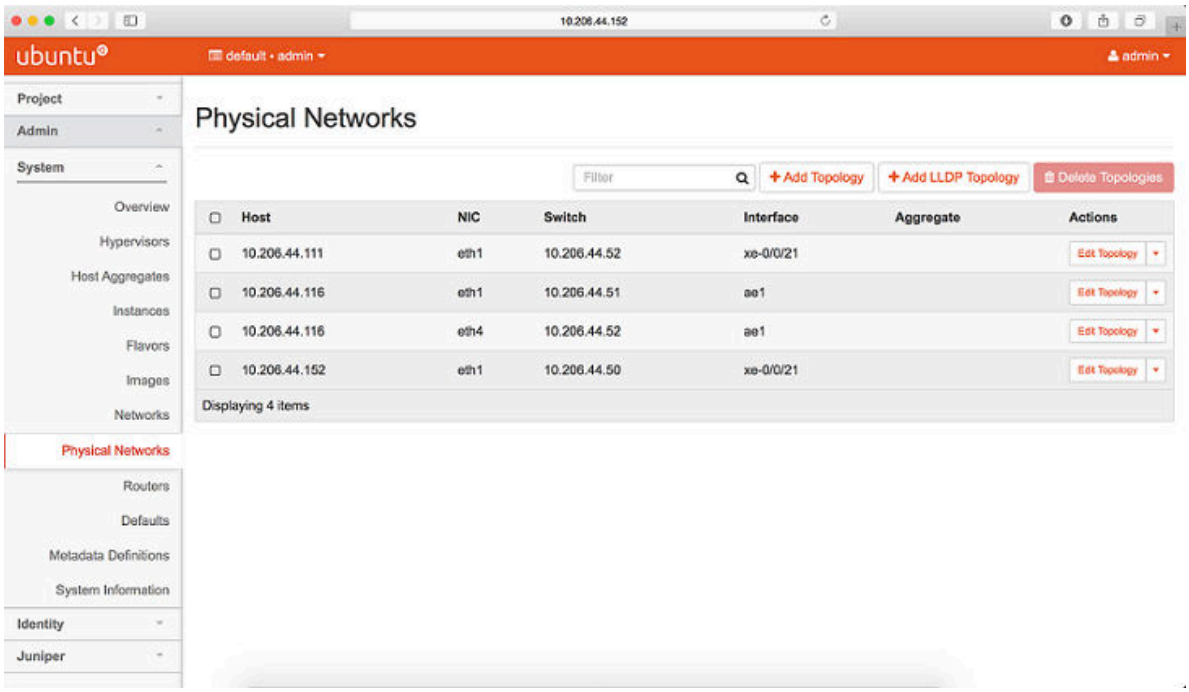


Figure 10 on page 43 shows how to add a physical topology.

Figure 10: Add Physical Topology

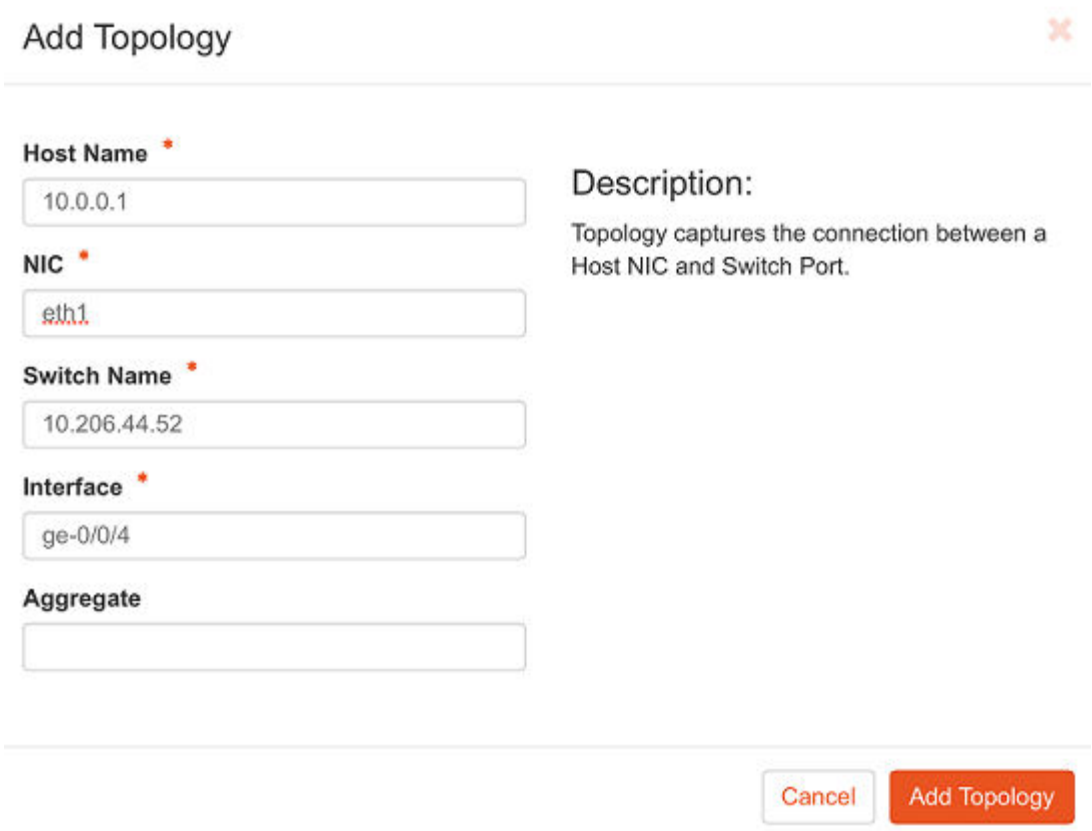


Figure 11 on page 44 shows how to add a physical topology from LLD.

Figure 11: Add Topology from LLDP

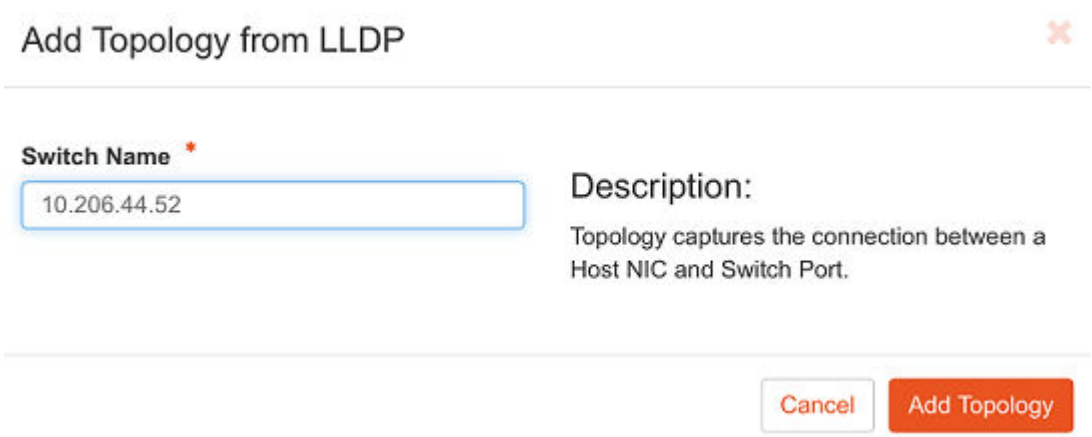


Figure 12 on page 45 shows how to edit a physical topology.

Figure 12: Edit Physical Topology

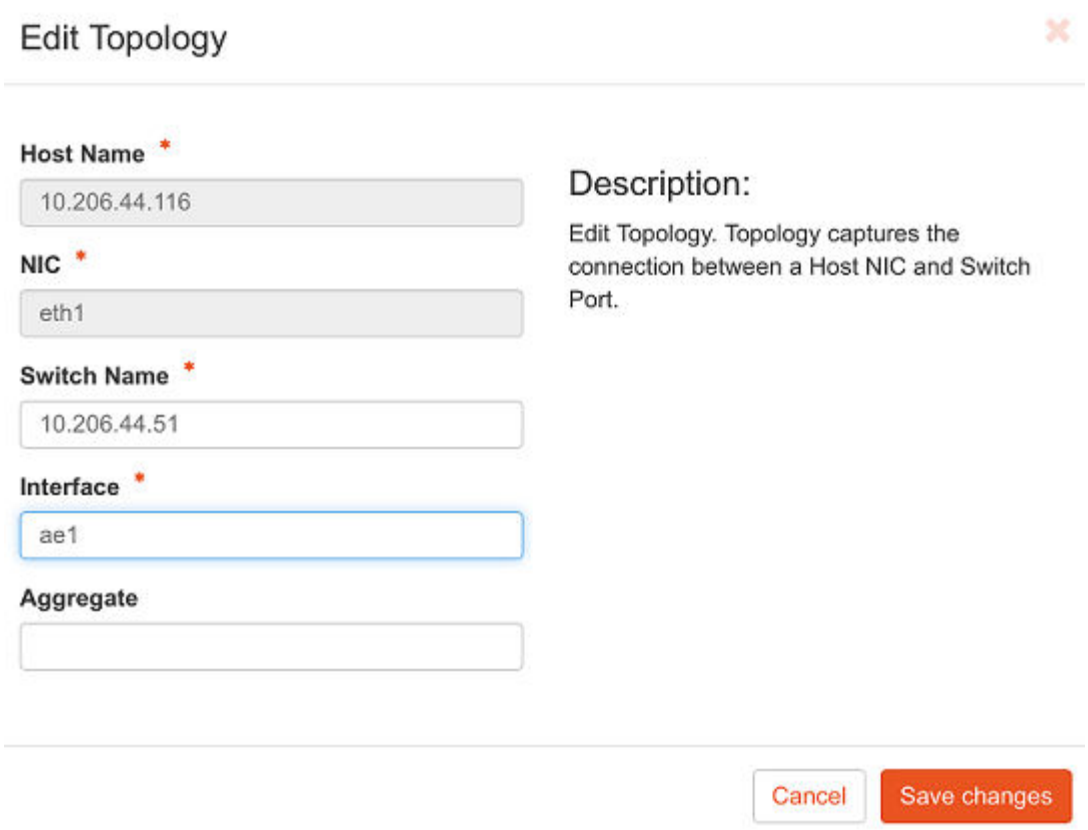


Figure 13 on page 46 shows how to delete a physical topology.

Figure 13: Delete a Physical Topology

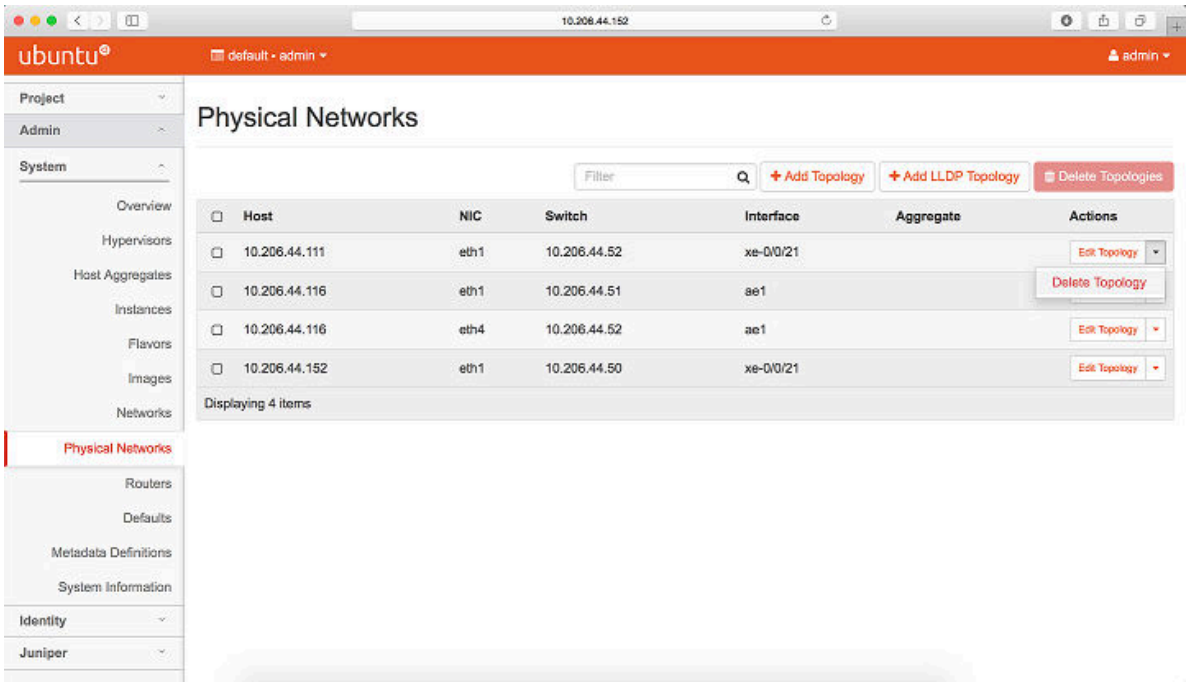
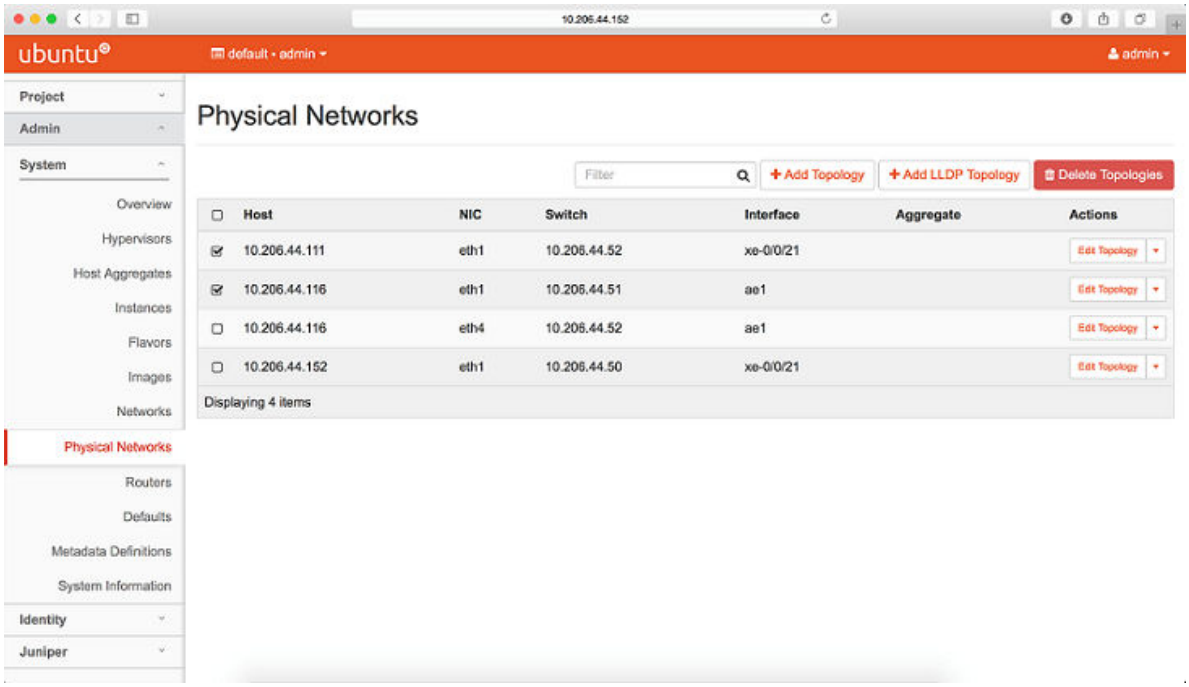


Figure 14 on page 47 shows how to delete multiple physical topologies.

Figure 14: Delete Multiple Physical Topologies



Configuring the Static Route Extension

Static route extension provides Horizon dashboard and Neutron REST API for configuring static routes with preferences. The Horizon dashboard is available at the following location: **Project > Network > Routers > Static Routes**.

Supported Devices

EX, QFX, SRX, and vSRX series devices

Configuring the Static Route Extension

To configure the static route extension:

1. Update the Neutron configuration file **/etc/neutron/neutron.conf** as follows:

```
service plugins = neutron.services.juniper_l3_router.dmi.l3.JuniperL3Plugin
```

2. Restart Neutron and Horizon services.

   Neutron-Server

   a. Ubuntu - service neutron-server restart

   b. Centos - systemctl restart neutron-server

   Apache (restarts Horizon)

   a. Ubuntu - service apache2 restart

   b. CentOS - systemctl restart httpd

3. From the OpenStack Dashboard, you can perform the following tasks using OpenStack tenant:

   Figure 15 on page 48 shows how to view static routes.

Figure 15: View Static routes

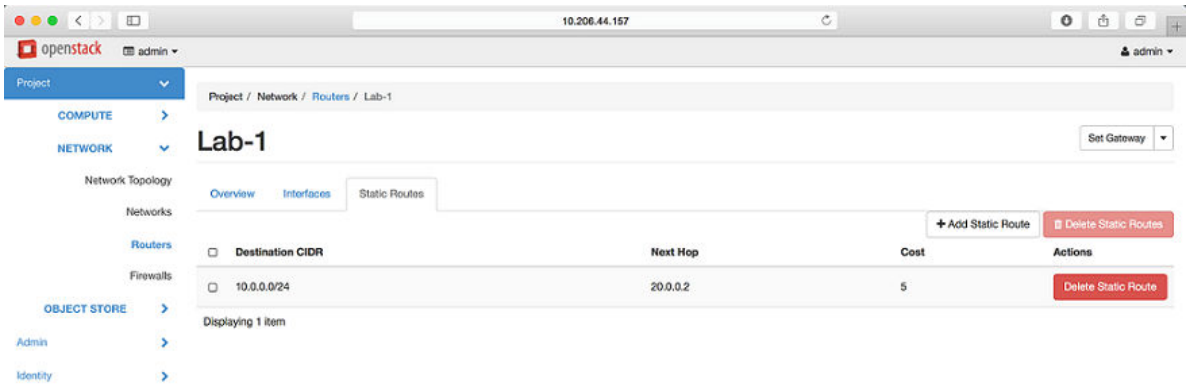


Figure 16 on page 49 shows how to add a static route.

Figure 16: Add a Static route

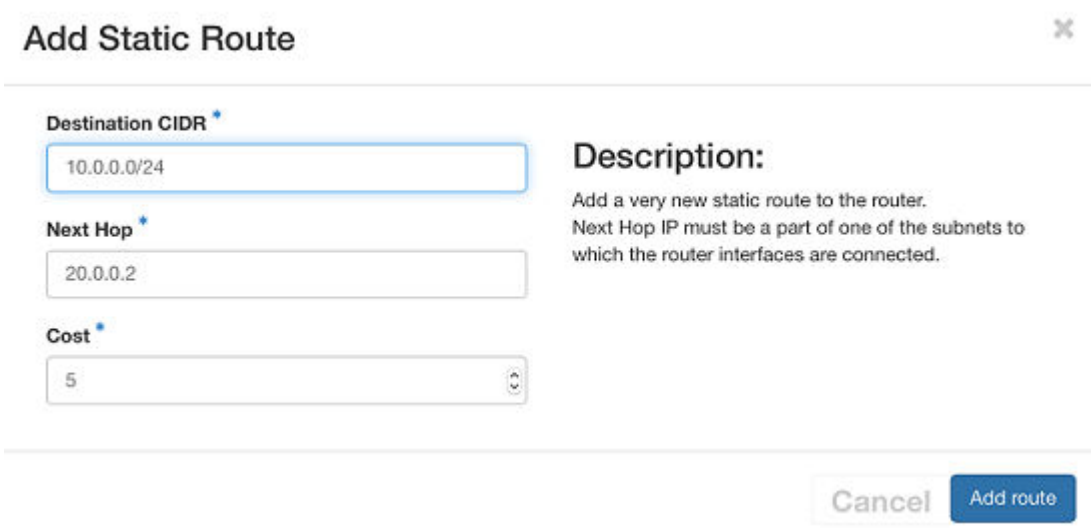
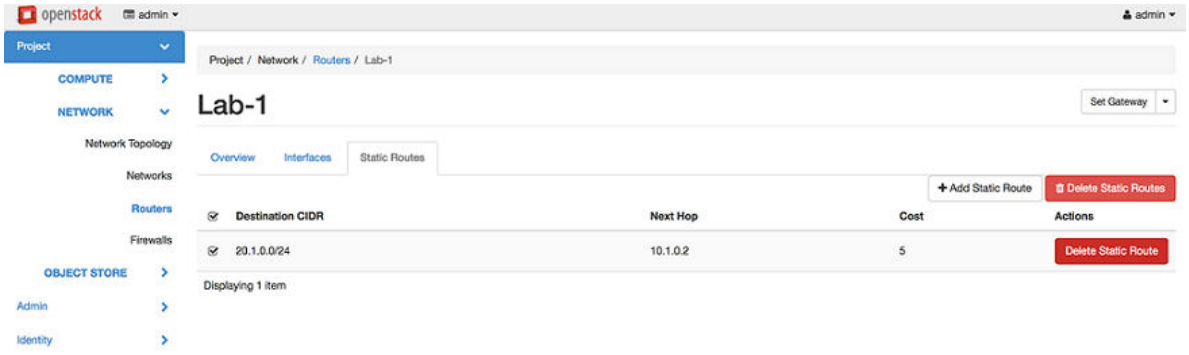


Figure 17 on page 49 shows how to delete a static route.

Figure 17: Delete Static Routes



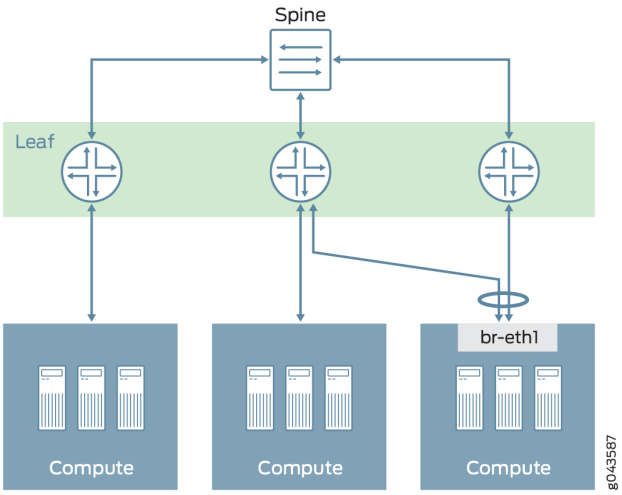
Configuring EVPN Multi-homing

To achieve network redundancy and load balancing, OpenStack nodes can be connected to more than one leaf switches that are capable of VXLAN-EVPN network. Juniper ML2 VXLAN-EVPN driver plug-in provisions the multi-homed peer devices with an identical Ethernet Segment Identification (ESI) number

and identical VLAN, VNI encapsulation details. This enables EVPN multi-homing functionality on the device.

OpenStack nodes can utilize all the multi-homing feature enabled uplinks to send traffic. This provides load balancing and redundancy in case of any failures. The uplink interface must be an aggregated interface.

Figure 18: EVPN Multi-homing



If more than one device connection is added for a particular OpenStack node using the **jnpr_switchport_mapping** command, the node is considered as multi-homing enabled. The interface must be an Aggregated Ethernet interface. This triggers an ESI ID generation and configures it to the aggregated switch interfaces.

Supported Devices and JUNOS Version

EX, QFX, SRX, and vSRX series devices

Configuration of ML2 VXLAN EVPN plug-in is a prerequisite for this plug-in. For more details on configuring the ML2 VXLAN EVPN plug-in, refer to Configuring ML2 VXLAN Plug-in with EVPN.

Additionally, the **jnpr_switchport_mapping** command creates the required physical topology name that is derived from the ESI ID and the bridge mapping details based on the topology inputs.

To update the configuration details:

1. Update the configuration details in the Open vSwitch Agent configuration file of the OpenStack nodes to which the switch is connected:

```
admin@controller:~$ jnpr_switchport_mapping add -H 10.206.44.116 -n eth3 -s
10.206.44.50 -p ae2
+---------------+------+-------------+------+-----------+
|      Host     | Nic  |    Switch   | Port | Aggregate |
+---------------+------+-------------+------+-----------+
| 10.206.44.116 | eth3 | 10.206.44.50 | ae2  |           |
+---------------+------+-------------+------+-----------+
================================================================
If you are using evpn driver, please update ovs l2 agent
config file /etc/neutron/plug-ins/ml2/openvswitch_agent.ini on
node 10.206.44.116 with bridge_mappings = 000000000010206044116:br-eth1
```

2. Update the physical_topology name with VLAN ranges in the neutron ML2 plug-in configuration file ml2_conf.ini as follows:

```
[ml2]
type_drivers = flat,vlan,vxlan,vxlan_evpn
tenant_network_types = vxlan_evpn
mechanism_drivers = jnpr_vxlan_evpn,openvswitch
#extension_drivers = port_security

[ml2_type_vlan]
network_vlan_ranges=10.206.44.50:10:1000,000000000010206044116:10:1000,10.206.4
4.56:10:1000

[ml2_type_vxlan]
vni_ranges = 10:5000
```

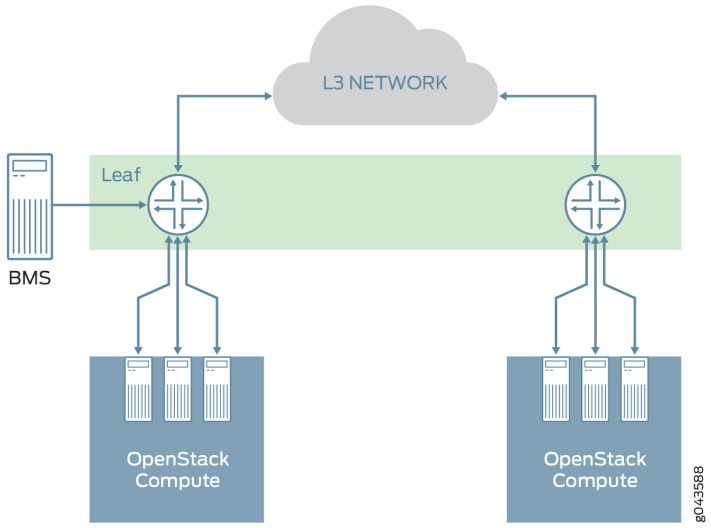
3. To verify whether the EVPN multi-homing plug-in is functioning properly, restart your neutron server, create networks, and VMs associated to the networks. The multi-homing enabled VMs are reachable when a redundant link is disabled.

Configuring EVPN BMS

Juniper VXLAN-EVPN plug-in supports integration of Bare Metal Server (BMS) into VXLAN-EVPN network. BMS communicates with the OpenStack VMs when it is connected through an OpenStack network.

Juniper plug-in supports integration of BMS into the OpenStack network. This provides accessibility to traditional physical devices from the OpenStack VM. Based on the plug-in configuration, BMS can be integrated into VLAN, VXLAN-EVPN network.

Figure 19: EVPN BMS Support



Supported Devices and JUNOS Version

Juniper EVPN BMS functionality is supported on QFX5100 leaf device with version 14.1X53-D40.

Plugin Configuration

To integrate BMS into the OpenStack network:

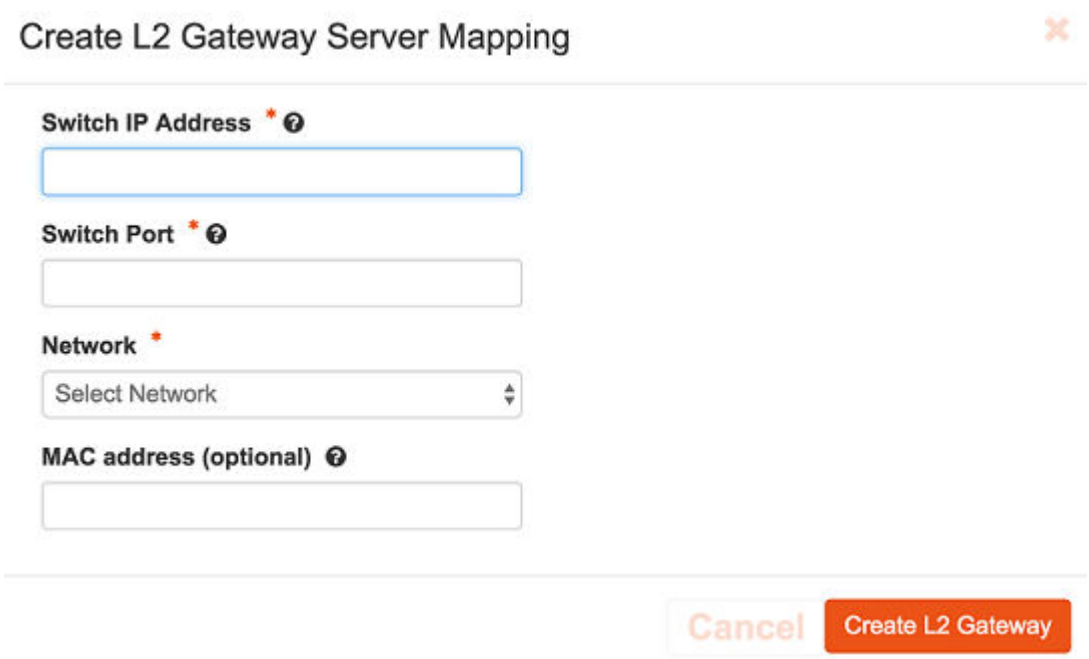
NOTE: Configuration of ML2 VXLAN EVPN plug-in is a prerequisite for this plug-in. For more information on how to configure ML2 VXLAN EVPN, refer Configuring ML2 VXLAN Plug-in with EVPN.

NOTE: The BMS must be connected to the Leaf device.

1. From the OpenStack Horizon GUI, select the OpenStack network to be linked with the BMS and provide the switch interface details. This provisions the necessary VLAN configuration on the device interface to establish connection to the OpenStack network and BMS.

   By providing the BMS MAC address, an IP address is allocated by the OpenStack DHCP server. By enabling DHCP client on BMS interface, allocated IP address can be obtained from OpenStack.

Figure 20: EVPN BMS Plugin Configuration



2. Type the following commands to enable BMS GUI on the Horizon dashboard:

- On Ubuntu:

```
usr/lib/python2.7/dist-packages/juniper_bms/openstack_dashboard/enabled/
_50_juniper.py
/usr/share/openstack-dashboard/openstack_dashboard/enabled/
```

```
sudo cp /usr/lib/python2.7/dist-packages/juniper_bms/openstack_dashboard/
enabled/_50_juniper.py
/usr/share/openstack-dashboard/openstack_dashboard/enabled/
```

- On Centos:

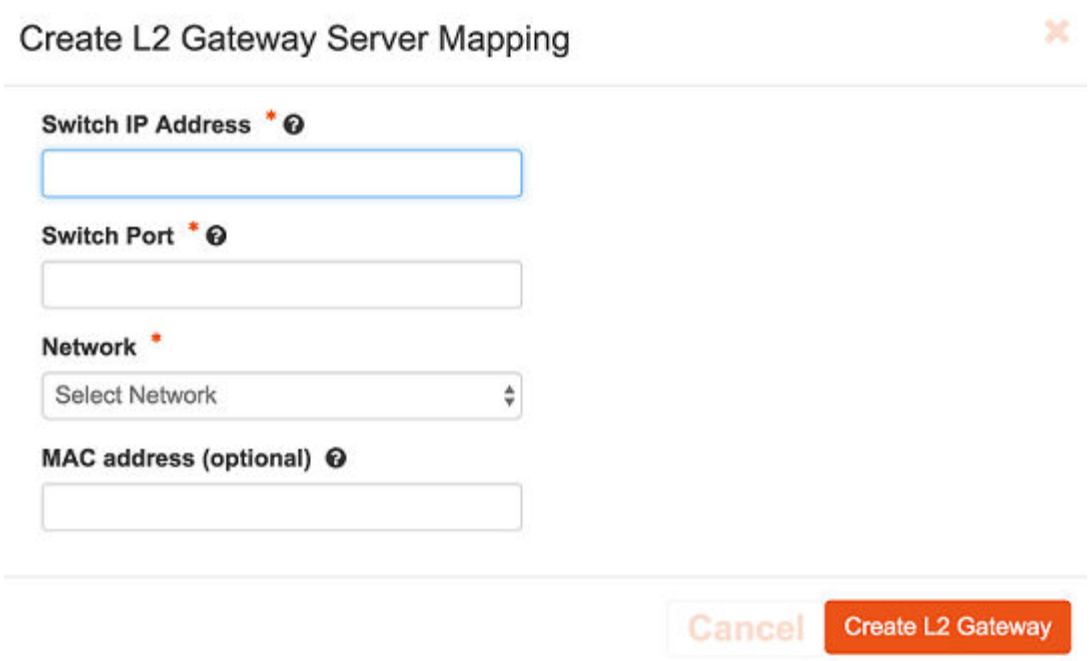
```
cp /usr/lib/python2.7/site-packages/juniper_bms/openstack_dashboard/
enabled/_50_juniper.py
/usr/share/openstack-dashboard/openstack_dashboard/enabled/
```

```
sudo cp /usr/lib/python2.7/site-packages/juniper_bms/openstack_dashboard/
enabled/_50_juniper.py
/usr/share/openstack-dashboard/openstack_dashboard/enabled/
```

3. Restart the Horizon dashboard.

On the Horizon dashboard, a new **Juniper->L2 Gateway** tab is displayed as shown in the Figure 21 on page 55 when the user logs in as **admin**.

Figure 21: EVPN BMS Plugin Configuration



Users can create an L2 Gateway by specifying the switch IP address, wherein the switch must be present in the topology, the interface connecting the BMS, the OpenStack network to be connected to, and optionally the BMS MAC address (required for dhcp to allocate IP address).

4. Verify whether the BMS is able to communicate with the the OpenStack guest VMs.

Configuring Network Address Translation

Network Address Translation (NAT) is a process for modifying the source or destination addresses in the headers of an IP packet while the packet is in transit. In general, the sender and receiver applications are not aware that the IP packets are manipulated.

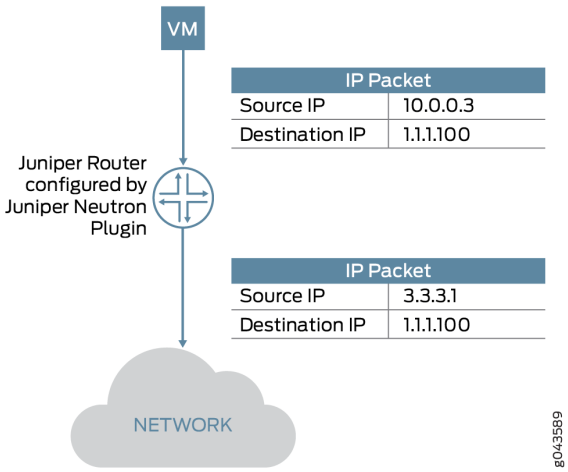
In OpenStack, external network provides Internet access for instances. By default, this network only allows Internet access from instances using Source Network Address Translation (SNAT). In SNAT, the NAT router modifies the IP address of the sender in IP packets. SNAT is commonly used to enable hosts with private addresses to communicate with servers on the public Internet.

OpenStack enables Internet access to an instance by using floating IPs. Floating IPs are not allocated to instances by default. Cloud users should get the floating IPs from the pool configured by the OpenStack administrator and then attach them to their instances. Floating IP is implemented by Destination Network Address Translation (DNAT). In DNAT, the NAT router modifies the IP address of the destination in IP headers.

SNAT – Internet Access from VM

Figure 22 on page 56 describes an OpenStack instance accessing Internet.

Figure 22: OpenStack Accessing the Internet

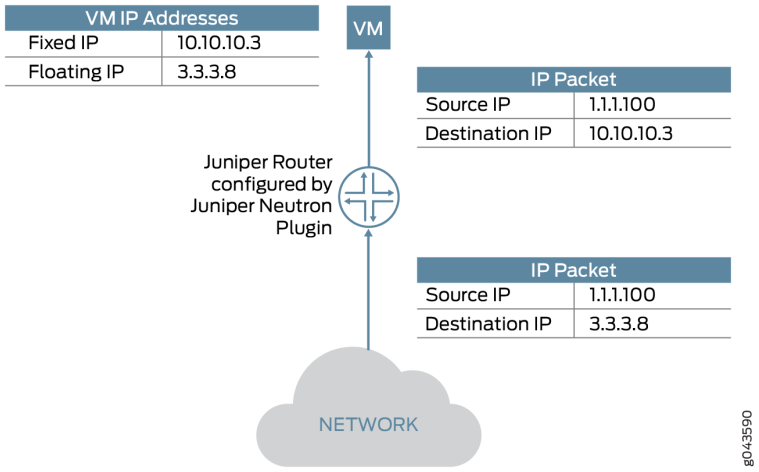


To enable Internet access from VMs, the network to which VM is connected should be connected to a router. This router must have its gateway set to the external network created by the administrator. Juniper Networks Neutron plug-in configures SNAT on the router to modify source address of all the VMs to the address of the interface that is connected to external network.

DNAT – Internet Access to VM

Figure 23 on page 57 describes accessing OpenStack instance from the Internet.

Figure 23: Accessing OpenStack from the Internet



To enable Internet access to VMs, a floating IP is allocated to the VM from a pool of IP addresses allocated to the tenant by administrator. The floating IP should be a routable IP. Juniper Networks Neutron plug-in configures the external facing interface of the router to proxy ARP for this IP address and DNATs for floating IP of the VM.

Plugin Configuration

To configure the plug-in:

1. Update the Neutron configuration file **/etc/neutron/neutron.conf** as follows:

```
service plug-ins = neutron.services.juniper_l3_router.dmi.l3.JuniperL3Plugin
```

2. Restart the Neutron service as follows:

```
service neutron-server restart
```

Configuring an External Network Access

To configure an external network access:

1. Create a network.
2. Launch an instance on the network created.

3. Create a router.
4. Add the new network to the router.
5. Set gateway of the router to the external network.
6. Ping from VM to any IP in the external network.

Configuring Access to VM from an External Network

To configure access to a VM from an external network:

1. Associate a floating IP from the floating IP pool of the external network to the instance that is created.
2. Configure security rules in security group to allow traffic from the external network. For example, ICMP ALLOW ALL for both ingress and egress traffic.
3. You can access the VM through the floating IP.

3
CHAPTER

References

Troubleshooting

The Juniper plug-in logs are available on Network node under the **folder /var/log/neutron/juniper***.

Referenced Documents

Table 3: References

Bookmark	Title / Author / Link	Revision
Release note	Juniper Networks Plug-In for OpenStack Neutron	

Glossary

Table 4: Terminologies and Acronyms

Term	Definition
ML2	Neutron Modular plugin 2
SNAT	Source Network Address Translation
DNAT	Destination Network Address Translation
VPNaaS	VPN-as-a-Service
ESI	Ethernet Segment Identification number

Table 4: Terminologies and Acronyms *(Continued)*

Term	Definition
BMS	Bare Metal Server
TOR	Top of Rack