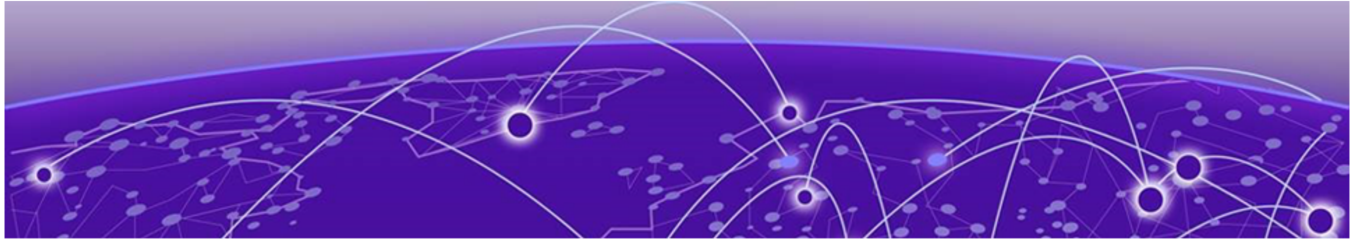
Extreme Fabric Automation 2.3.2

OpenStack Integration Guide

9036771-01 Rev AA
November 2020

Table of Contents



Preface

This section describes the text conventions used in this document, where you can find additional information, and how you can provide feedback to us.

Text Conventions

Unless otherwise noted, information in this document applies to all supported environments for the products in question. Exceptions, like command keywords associated with a specific software version, are identified in the text.

When a feature, function, or operation pertains to a specific hardware product, the product name is used. When features, functions, and operations are the same across an entire product family, such as ExtremeSwitching switches or SLX routers, the product is referred to as *the switch* or *the router*.

Table 1: Notes and warnings

Icon	Notice type	Alerts you to...
💡	Tip	Helpful tips and notices for using the product
🗒	Note	Useful information or instructions
➡	Important	Important features or instructions
⚠	Caution	Risk of personal injury, system damage, or loss of data
⚠	Warning	Risk of severe personal injury

Table 2: Text

Convention	Description
`screen displays`	This typeface indicates command syntax, or represents information as it is displayed on the screen.
The words *enter* and *type*	When you see the word *enter* in this guide, you must type something, and then press the Return or Enter key. Do not press the Return or Enter key when an instruction simply says *type*.
Key names	Key names are written in boldface, for example **Ctrl** or **Esc**. If you must press two or more keys simultaneously, the key names are linked with a plus sign (+). Example: Press **Ctrl**+**Alt**+**Del**
Words in italicized type	Italics emphasize a point or denote new terms at the place where they are defined in the text. Italics are also used when referring to publication titles.
NEW!	New information. In a PDF, this is searchable text.

Table 3: Command syntax

Convention	Description
bold text	Bold text indicates command names, keywords, and command options.
italic text	Italic text indicates variable content.
[ ]	Syntax components displayed within square brackets are optional. Default responses to system prompts are enclosed in square brackets.
{ **x** | **y** | **z** }	A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options.
x | **y**	A vertical bar separates mutually exclusive elements.
< >	Nonprinting characters, such as passwords, are enclosed in angle brackets.
...	Repeat the previous element, for example, *member*[*member*...].
\	In command examples, the backslash indicates a "soft" line break. When a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash.

Documentation and Training

Find Extreme Networks product information at the following locations:

Current Product Documentation

Release Notes

Hardware and software compatibility for Extreme Networks products

Extreme Optics Compatibility

Other resources such as white papers, data sheets, and case studies

Extreme Networks offers product training courses, both online and in person, as well as specialized certifications. For details, visit www.extremenetworks.com/education/.

Getting Help

If you require assistance, contact Extreme Networks using one of the following methods:

Extreme Portal
: Search the GTAC (Global Technical Assistance Center) knowledge base; manage support cases and service contracts; download software; and obtain product licensing, training, and certifications.

The Hub
: A forum for Extreme Networks customers to connect with one another, answer questions, and share ideas and feedback. This community is monitored by Extreme Networks employees, but is not intended to replace specific guidance from GTAC.

Call GTAC
: For immediate support: (800) 998 2408 (toll-free in U.S. and Canada) or 1 (408) 579 2826. For the support phone number in your country, visit: www.extremenetworks.com/support/contact

Before contacting Extreme Networks for technical support, have the following information ready:

- Your Extreme Networks service contract number, or serial numbers for all involved Extreme Networks products
- A description of the failure
- A description of any actions already taken to resolve the problem
- A description of your network environment (such as layout, cable type, other relevant environmental information)
- Network load at the time of trouble (if known)
- The device history (for example, if you have returned the device before, or if this is a recurring problem)
- Any related RMA (Return Material Authorization) numbers

Subscribe to Service Notifications

You can subscribe to email notifications for product and software release announcements, Vulnerability Notices, and Service Notifications.

1. Go to www.extremenetworks.com/support/service-notification-form.
2. Complete the form (all fields are required).
3. Select the products for which you would like to receive notifications.

   Note
   You can modify your product selections or unsubscribe at any time.

4. Select **Submit**.

Providing Feedback

The Information Development team at Extreme Networks has made every effort to ensure the accuracy and completeness of this document. We are always striving to improve our documentation and help you work better, so we want to hear from you. We welcome all feedback, but we especially want to know about:

- Content errors, or confusing or conflicting information.
- Improvements that would help you find relevant information in the document.
- Broken links or usability issues.

If you would like to provide feedback, you can do so in three ways:

- In a web browser, select the feedback icon and complete the online feedback form.
- Access the feedback form at https://www.extremenetworks.com/documentation-feedback/.
- Email us at documentation@extremenetworks.com.

Provide the publication title, part number, and as much detail as possible, including the topic heading and page number if applicable, as well as your suggestions for improvement.

About this Document

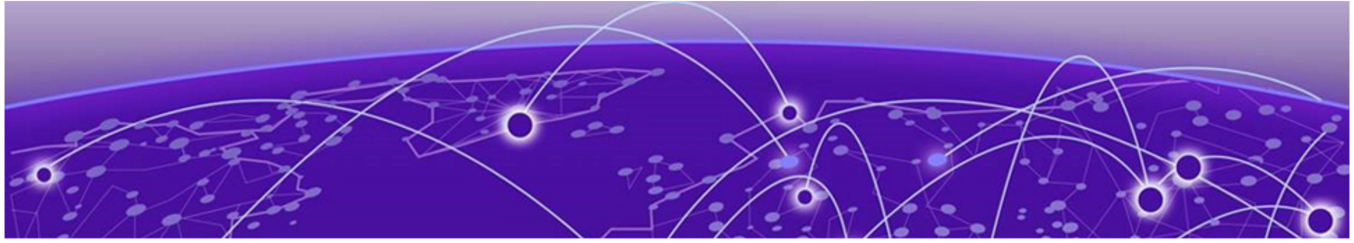
What's New in this Document

The following table describes information added to this guide for the Extreme Fabric Automation 2.3.2 software release.

Table 4: Summary of changes

Feature	Description	Described in
EFA health monitoring	You can use the **`efa-health show`** command to monitor EFA connectivity.	• Verify EFA Health on page 49 • efa-health show on page 59

For complete information on this release, refer to the *Extreme Fabric Automation Release Notes*.



Introduction to OpenStack Integration

Extreme Fabric Automation Overview

Extreme Fabric Automation (EFA) is a micro-services-based scalable fabric automation application.

EFA automates and orchestrates SLX IP Fabrics and tenant networks, with support for the following:

- Building and managing non-Clos small data center Fabrics and 3-stage and 5-stage IP Clos Fabrics
- Managing tenant-aware Layer 2 and Layer 3 networks
- Configuring integration with several ecosystems: VMware vCenter, OpenStack, and Microsoft Hyper-V
- Providing a single point of configuration for your entire Fabric

EFA consists of core K3s containerized services that interact with each other and with other infrastructure services to provide the core functions of Fabric and tenant network automation. For more information, see EFA Microservices on page 13.

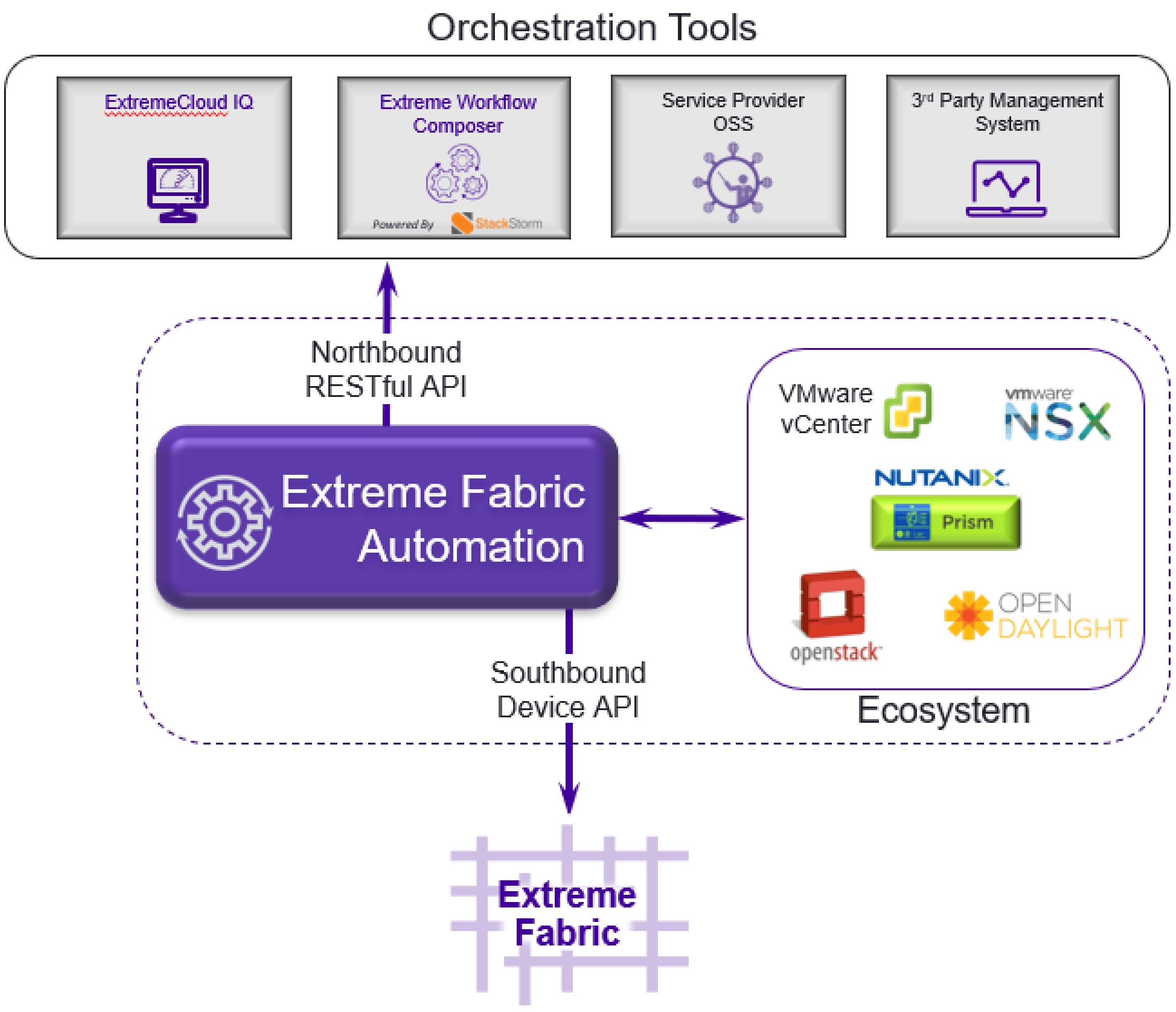


Figure 1: EFA orchestration

EFA Microservices

EFA consists of core K3s containerized microservices that interact with each other and with other infrastructure services to provide the core functions of fabric and tenant network automation.

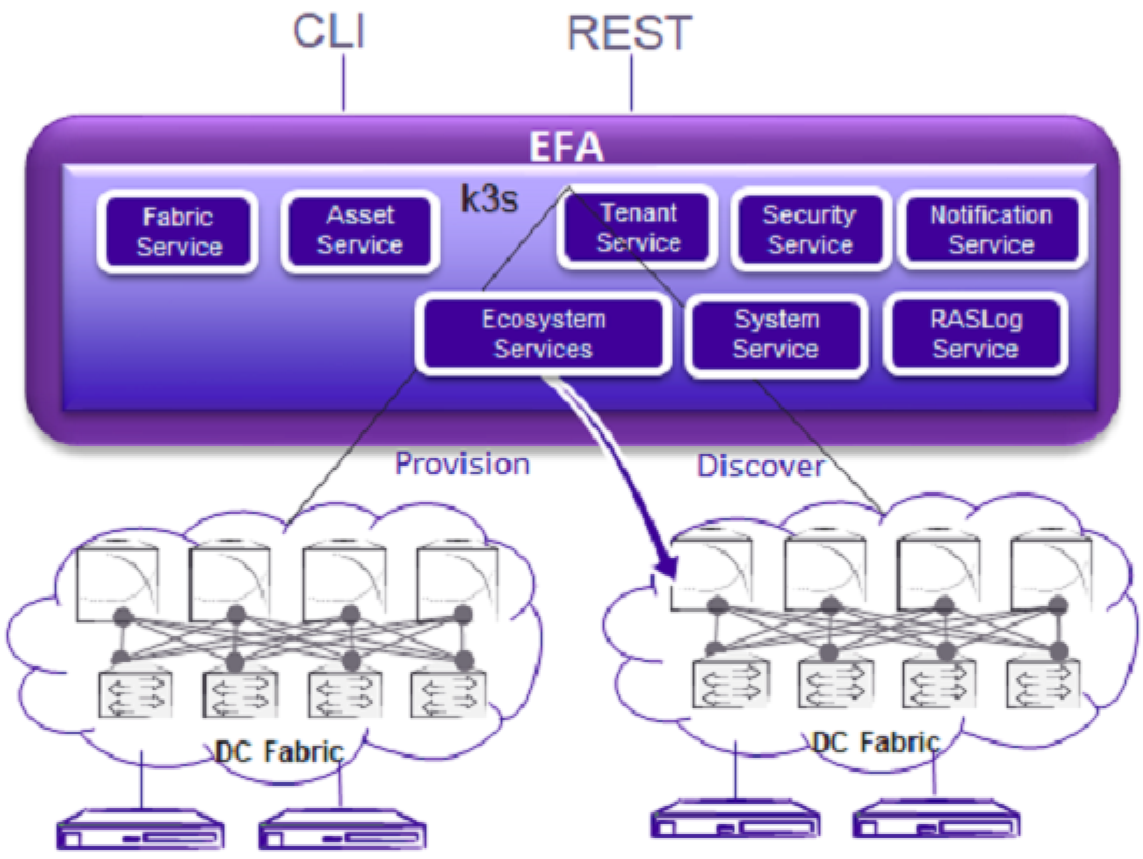


Figure 2: Microservices in the EFA architecture

Fabric Service

The Fabric Service is responsible for automating the fabric BGP underlay and EVPN overlay. By default, the EVPN overlay is enabled but you can disable it before provisioning, if necessary. The Fabric Service exposes the CLI and REST API for automating the fabric underlay and overlay configuration.

The Fabric Service features include:

- Support for small datacenters (non-Clos)
- Support for 3-stage and 5-stage Clos fabrics
- Support for MCT configuration
- Support for ecosystem integration: OpenStack, VMware vCenter, and Microsoft Hyper-V

Underlay automation includes interface configurations (IP numbered), BGP underlay for spine and leaf, BFD, and MCT configurations. Overlay automation includes EVPN and overlay gateway configuration.

Tenant Service

The Tenant Service manages tenants, tenant networks, and end points, fully leveraging the knowledge of assets and the underlying fabric. You can use the CLI and REST API for tenant network configuration on Clos and non-Clos fabrics.

Tenant network configuration includes VLAN, BD, VE, EVPN, VTEP, VRF, and router BGP configuration on fabric devices to provide layer-2 extension, layer-3 extension across the fabric, layer-2 handoff, and layer-3 handoff at the edge of the fabric.

Inventory Service

The Inventory Service acts as an inventory of all the necessary physical and logical assets of the fabric devices. All other EFA services rely on asset data for their configuration automation. The Inventory Service is a REST layer on top of device inventory details, with the capability to filter data based on certain fields. The Inventory Service securely stores the credentials of devices in encrypted form and makes those credentials available to different components such as the Fabric and Tenant services.

The Inventory Service supports the **`execute-cli`** option for pushing configuration and exec commands to devices. Examples include configuring SNMP parameters or OSPF configurations. This means you can use EFA for SLX-OS commands and push the same configuration to multiple devices.

The Asset Service provides the secure credential store and deep discovery of physical and logical assets of the managed devices. The service publishes the Asset refresh and change events to other services.

Notification Service

The Notification Service sends events, alerts, and tasks to external entities. Notifications sent from EFA are derived from the syslog events received from the devices that EFA manages. Alerts are notifications that services in EFA send for unexpected conditions. Tasks are user-driven operations or timer-based tasks such as device registration or fabric creation.

RASlog Service

The RASlog Service processes syslog messages from devices, processes SNMP traps from devices, and forwards notifications to subscribers.

Security Service

The Security Service consists of authentication and authorization features that enforce a security boundary between northbound clients and downstream operations between EFA and SLX devices. The service also validates users and their credentials through Role-based Access Control (RBAC) and supports local and remote (LDAP) login.

Ecosystem Integration Services

EFA provides one-touch integration with these ecosystems, providing deep insight into VMs, vSwitches, port groups, and hosts, and the translation of these into IP fabric networking constructs.

VMware vCenter Service

The vCenter integration provides connectivity between EFA and vCenter using a REST API. EFA does not connect to individual ESXi servers. All integration is done through vCenter. For more information, see the *Extreme Fabric Automation VMware vCenter Integration Guide, 2.3.0*. Integration support includes the following:

- Registration or deregistration of one or more vCenter servers in EFA
- Updates for vCenter asset details
- Lists of information about vCenter servers
- Inventory integration
- Dynamic updates about Tenant Service integration from vCenter and from EFA services

Hyper-V

The Hyper-V integration supports networking configuration for Hyper-V servers in a datacenter, manual and automated configuration updates when VMs move, and visibility into the VMs and networking resources that are deployed in the Hyper-V setup. For more information, see *Extreme Fabric Automation Hyper-V Integration Guide, 2.3.0*. Integration support includes the following:

- SCVMM (System Center Virtual Machine Manager) server discovery
- SCVMM server update
- Periodic polling of registered SCVMM servers
- SCVMM server list
- SCVMM server delete and deregister
- Network event handling

OpenStack Service

The OpenStack service integrates Extreme OpenStack plug-ins with the rest of the EFA foundation services in an IP fabric. For more information, see the *Extreme Fabric Automation OpenStack Integration Guide, 2.3.0*. Integration support includes the following:

- Create, read, update, delete (CRUD) operations on networks and ports
- LAG support
- Provider network (default, PT)
- VLAN trunking
- Network operations using single-root I/O virtualization (SR-IOV), physical and virtual functions
- East-west traffic using Virtio ports
- VMotion (virtual machine migration)
- Single-homed connections to the edge port

OpenStack Integration Overview

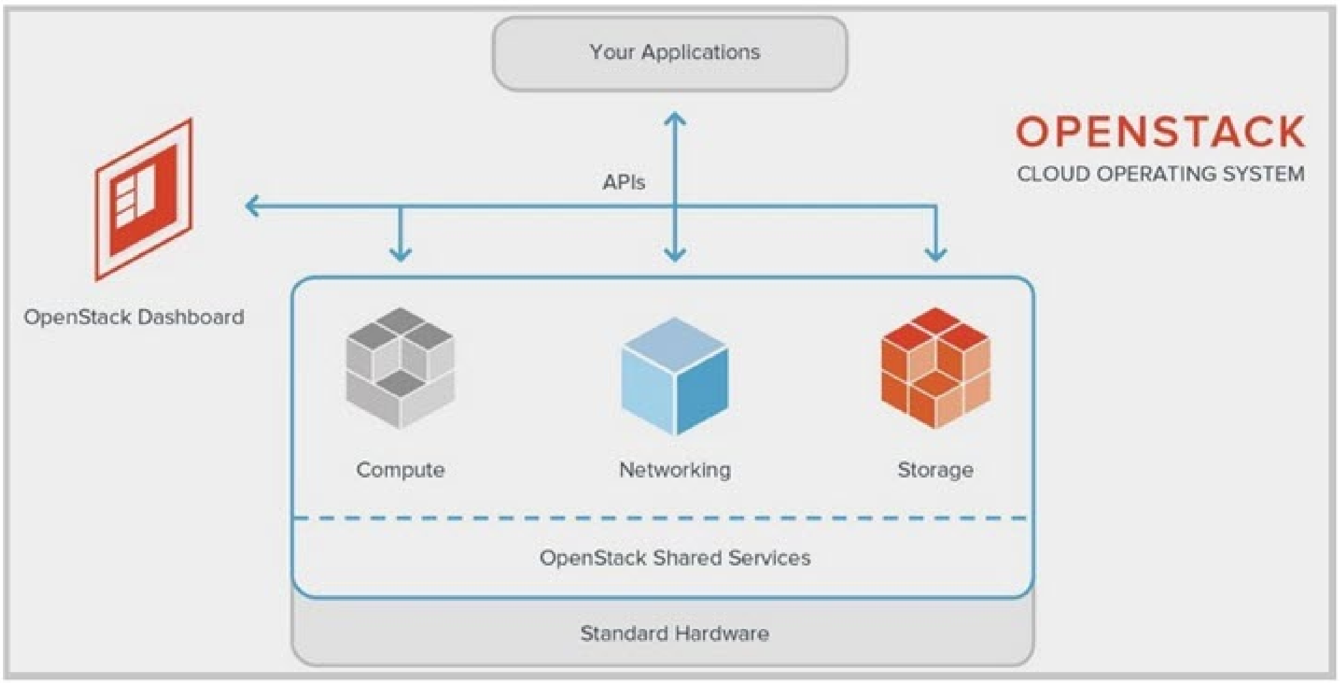
OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a Data Center. OpenStack Integration enables System administrators to manage and provision resources through APIs and web interface.

OpenStack Core Components

The following table lists the OpenStack core components.

Component	Name	Description
Compute	Nova	Compute service that enables provisioning of compute instances or virtual servers and supports creating virtual machines and external Linux servers.
Networking	Neutron	Networking service that provides layer 2 network connectivity for virtual devices.
Dashboard	Horizon	OpenStack project that provides an extensible, unified, and web-based user interface for all OpenStack services.
Storage	Cinder	OpenStack Block Storage service for providing volumes to Nova virtual machines.

Figure 3: OpenStack core components



OpenStack Network Nodes

The EFA OpenStack network consists of Controller and Compute nodes.

Most of the shared OpenStack services and other tools run on the Controller node. The Controller node supplies API, scheduling, and other shared services for the cloud. The Controller node includes dashboard, image store, and identity service. Nova Compute management service and Neutron server are also configured on the Controller node.

The VM instances or Nova Compute instances are installed on the Compute node.

The following figure shows an overview of the EFA OpenStack Integration.

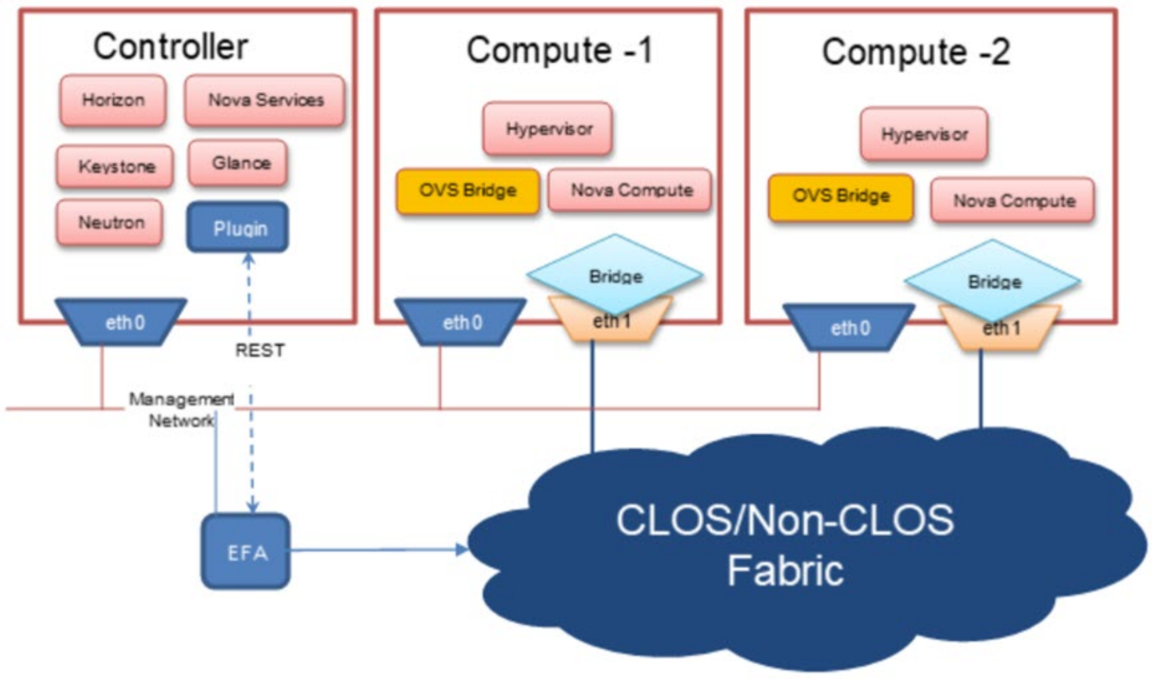


Figure 4: Overview of EFA OpenStack Network

OpenStack Ecosystem Integration Management

OpenStack Service is an ecosystem service that provides integration of Extreme OpenStack plug-ins with the rest of the of the EFA foundation services in an IP fabric network.

Foundation Service	Description
Fabric Service	This service provides mechanisms to create: • NON-CLOS/CLOS Fabric • CLOS Fabrics can be 3-Stage or 5-Stage
Tenant Service	This service provides mechanisms to create: • Layer 2 Networks • Layer 3 Networks
Asset or Inventory Service	This service provides mechanisms to manage: • Physical and Logical assets of the switches in the Fabric • Manage the credentials of the switches

The following figure shows an overview of OpenStack ecosystem integration management.

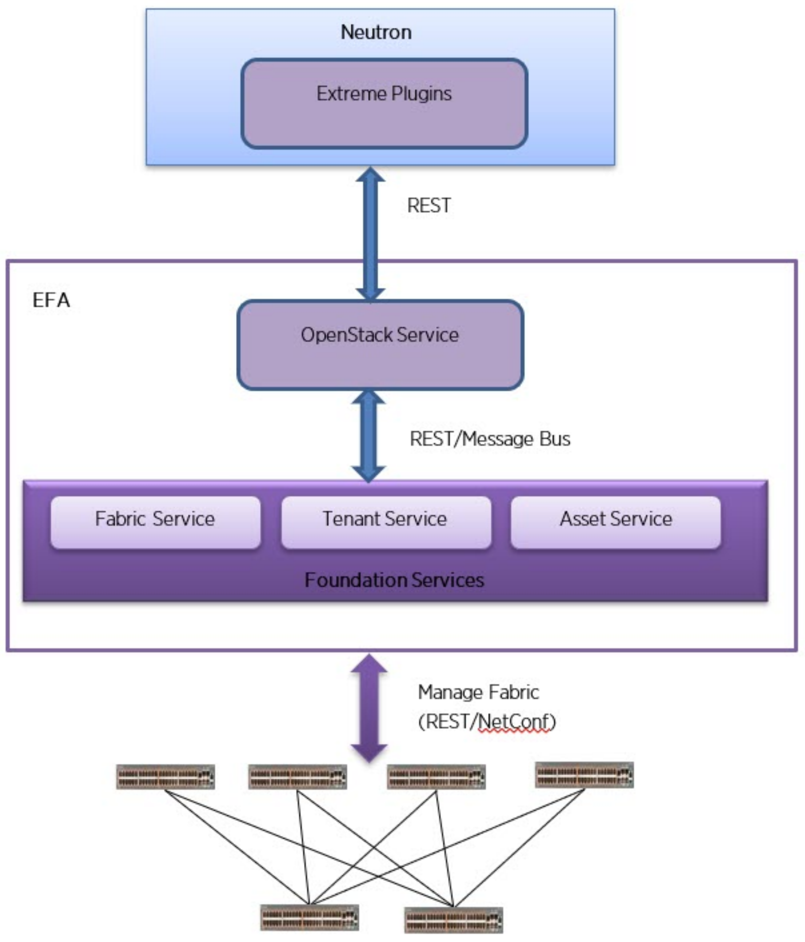


Figure 5: Overview of OpenStack ecosystem integration management

OpenStack Service Commands

OpenStack Service commands show the Neutron constructs and their provisioning status on EFA. For more information about OpenStack Service commands, see EFA OpenStack Service Command Reference on page 51.

For EFA commands and supported parameters, see *Extreme Fabric Automation Command Reference, 2.3.0*.

Limitations

OpenStack Ecosystem Integration limitations are as follows:

- Only green field deployment is supported.

Dependencies

OpenStack Ecosystem Integration dependencies are as follows:

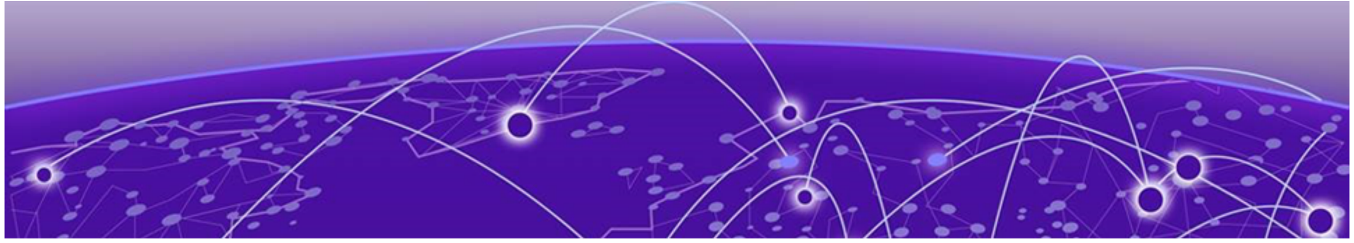
- Non-CLOS (2-Node, 4-Node, and 8-Node)

- CLOS
    - 3-stage CLOS (Leaf / Spine nodes)
    - 5-stage CLOS (Leaf / Spine / Super-Spine nodes)
- Compute nodes (connected to every Leaf node in MCT and Non-MCT configurations)

Supported Platforms

OpenStack Ecosystem Integration is supported on the following platforms.

- SLX 9640 (Border leaf devices)
- SLX 9150 (Spine and leaf devices)
- SLX 9150T (Spine and leaf devices)
- SLX 9250 (Spine and leaf devices)



OpenStack Plugin Installation

EFA OpenStack Plug-in Package

OpenStack Integration requires Modular Layer 2 (ML2) Mechanism Driver to interact with Extreme Fabric Automation (EFA). The Neutron drivers and plug-ins communicate with EFA over the REST interface.

The EFA plug-in for OpenStack is packaged as `deb` files for Ubuntu Linux.

The EFA OpenStack plug-in package supports the following features:

- Modular Layer 2 (ML2) Mechanism Driver (Neutron)
- Topology
- Single Root I/O Virtualization (SR-IOV)
- Multi-Segment
- Virtual Machine Migration (VMotion)
- OpenStack Service
- Sync EFA with Neutron

System Requirements

System requirements for EFA OpenStack Integration are as follows:

Node	
CPU cores	4
Storage	50 G
RAM	16 GB
OpenStack	Pike
EFA	2.3.0
Operating System	Ubuntu 16.04
Python	2 and 3

Install EFA OpenStack Neutron Plug-in

This section provides information required to install Extreme Fabric Automation OpenStack Neutron plug-in on Ubuntu.

Before You Begin

The prerequisites for installing EFA OpenStack Neutron plug-in are as follows:

- Working knowledge of Ubuntu Linux
- Experience in OpenStack deployment
- Experience in managing EFA 2.x.x
- Network nodes are prepared as required
- All OpenStack compute, network, and controller node hostname name resolutions are setup (fully qualified Host names (fqdn) are not supported for beta release)
- All OpenStack nodes are configured with unique hostnames
- Working EFA Fabric using `efa cli/rest`
- OpenStack nodes are connected to the leaf switches either in direct mode, VPC, or bonded mode
- Bonding setup is done using 802.3ad

Note

EFA OpenStack Neutron plug-in supports only Extreme specific OpenStack services. If other OpenStack services are required, install the respective plug-ins prior to EFA OpenStack Neutron plug-in installation.

All network and server connection settings or mappings can be saved to `csv` files for bulk configuration using the `startup` file option in the `ml2_conf_extreme.ini` file.

Procedure

1. Install the EFA OpenStack plug-in debian packages.

   ```
   # dpkg -i networking_extreme*deb
   ```

2. Note the Neutron configuration file layout.
   - Neutron configuration: `/etc/neutron/neutron.conf`
   - ML2 plug-in configuration: `/etc/neutron/plugins/ml2/ml2_conf.ini`
   - Extreme EFA Mechanism driver or topology configuration: `/etc/neutron/plugins/ml2/ml2_conf_extreme.ini`
3. Configure the `Ml2 core_plugin` in the `neutron.conf` file.

   ```
   [DEFAULT]
   core_plugin=ml2
   service_plugins = trunk, segments, efa_topology_plugin
   ```

   Do not enable the reference router plug-in.
4. Copy the `ml2_conf_extreme.ini` file provided with TAR to the `/etc/neutron/plugins/ml2/ml2_conf_extreme.ini` file.
5. Enable Neutron in the `ml2_conf_extreme.ini` file to communicate with EFA.

   ```
   [ml2_extreme]
   efa_rest_token = <efa_api_token for VIM_1>
   efa_cert_file = /root/gcla/extreme-ca-chain.crt
   efa_secure_mode = True
   efa_port = 443
   ```

```
efa_host =efa.extremenetworks.com
region_name = VIM_1
region_shared = SHARED_TENANT
#SHARED_TENANT is the name of the shared tenant created on EFA
fabric_name = CNCF
```

6. Enable the Neutron EFA extension plug-in in the `ml2_conf_extreme.ini` file to build initial physical topology between OpenStack Compute nodes and TOR switches.

   ```
   [efa_topology]
   efa_link_mapping_file = /home/ubuntu/link.csv
   ```

7. Enable Extreme EFA mechanism drivers in `Ml2_conf.ini`.

   ```
   Ml2_conf.ini
   [ml2]
   tenant_network_types = vlan
   type_drivers = vlan
   mechanism_drivers = openvswitch,extreme_efa
   [ml2_type_vlan]
   network_vlan_ranges = physnet1:100:500 (Required vlan range)
   [ovs]
   bridge_mappings = physnet1:br0 (bridge used for datapath)
   ```

8. Modify the system unit file to start Neutron with `ml2_conf_extreme.ini`.

   ```
   # ExecStart = /usr/local/bin/neutron-server --config-file /etc/neutron/neutron.conf --
   config-file /etc/neutron/plugins/ml2/ml2_conf.ini  --config-file /etc/neutron/
   plugins/ml2/ml2_conf_extreme.ini

   # systemctl daemon-reload
   ```

   On DevStack installation, modify the `/etc/systemd/system/devstack@q-svc.service` file.

9. Restart the Neutron server.

   ```
   # systemct1 restart
   ```

   On Opensource installation, use the **`sudo service neutron-* restart`** command.

   On DevStack installation, use the **`sudo systemctl restart devstack@q-svc.service`** command.

10. Verify if the status of the Neutron server is `Active` and confirm the following:

    - Neutron service started with the `ml2_confi_extreme.ini` file.
    - **`efa-topology`** extension is loaded using **`OpenStack extension show efa-topology`**.

    ```
    # sudo systemctl status devstack@q-svc.service
    ```

    On Opensource stack installation, use the **`sudo service neutron-* status`** command.

Upgrade or Downgrade OpenStack Neutron Plug-in

Procedure

Upgrade or downgrade EFA plug-in debian packages from the network node.

```
# dpkg -i networking_extreme*deb
```

Uninstall EFA OpenStack Neutron Plug-in

Procedure

Uninstall the EFA OpenStack Neutron plug-in.

```
# dpkg -P networking-extreme
```

OpenStack Configuration

Multiple VIM/VPOD Instances

Each VIM/VPOD instance or OpenStack is mapped to a separate EFA Tenant. A VIM Instance can be segregated as follows:

- Grouping all physical Interfaces together through topology specification
- Restricting the VLAN range details in the Neutron initialization file

The following figure shows an example of multiple VIM/VPOD instances managing the same IP fabric.

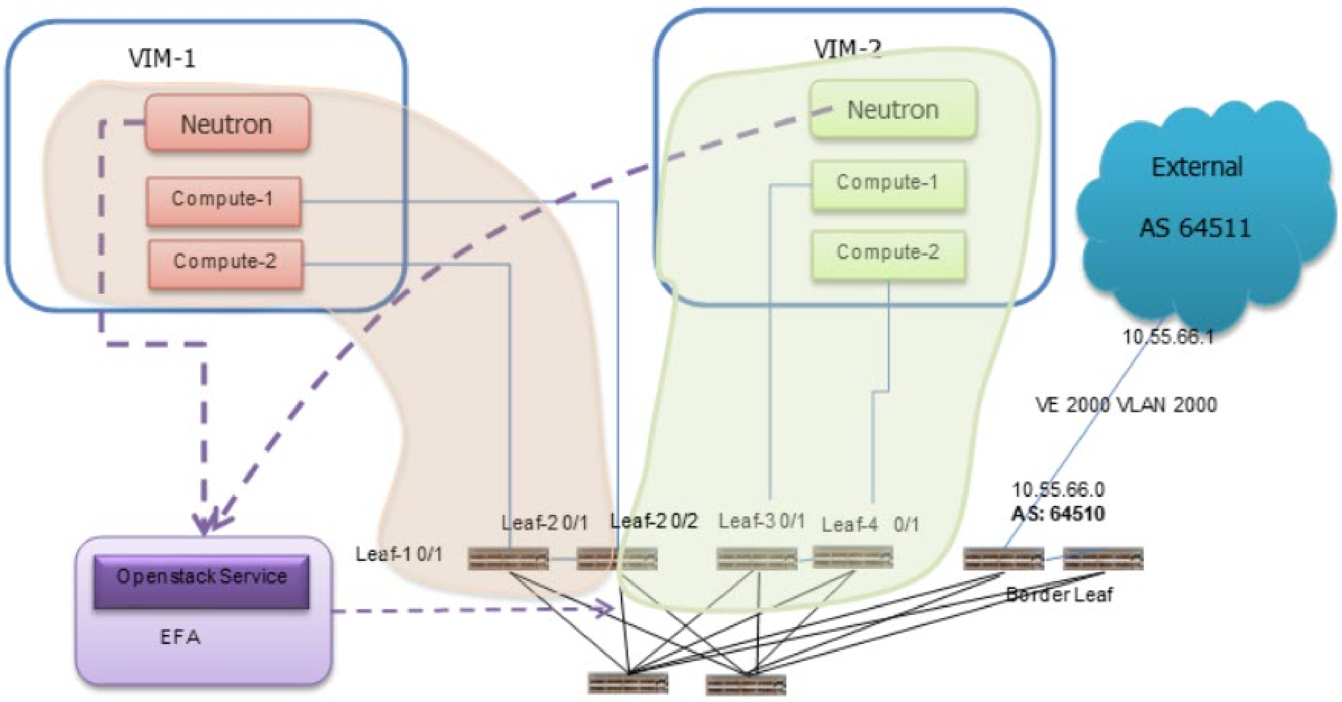


Figure 6: Multiple VIM or VPOD instances

VIM	EFA Tenant	BD	Properties
VIM1	OS-Tenant-VIM1	BD is disabled CTAG mapped to VLAN on IP Fabric	VLAN-Range = 2-4080 Specified in ml2_conf.ini `network_vlan_ranges = physnet1:2:4080` Physical interfaces can be specified using the topology.
VIM2	OS-Tenant-VIM2	BD is enabled CTAG mapped to BD on IP Fabric	LAN-Range = 2-4080 Specified in ml2_conf.ini `network_vlan_ranges = physnet1:2:4080` Tenant created with **–bd-enable flag**. Physical interfaces can be specified using the topology.

The following example shows multiple VIM/VPOD instances.

```
# efa tenant create --name VIM1 --port 10.24.80.111[0/1],10.24.80.112[0/1] --vlan-range
2-4080  --vrf-count 200

# efa tenant create --name VIM2 --port 10.24.80.112[0/2],10.24.80.113[0/1],
10.24.80.114[0/1]  --enable-bd  --vlan-range 2-4080  --vrf-count 200

# efa tenant show
+----------------+------------+------------+-----------+----------+----------
+-------------------------------------+
|      Name      | L2VNI-Start | L3VNI-Start | VLAN-Range | VRF-Count | Enable-BD
|              Ports                    |
+----------------+-------------+-------------+------------+-----------+-----------
+-----------------------------------+
| default-tenant | *           | *           | *          | *         | *         |
*                                   |
+----------------+-------------+-------------+------------+-----------+-----------
+-----------------------------------+
| VIM1           | 0           | 0           | 2-4080     | 200       | False     |
10.24.80.111[0/1]                   |
|                |             |             |            |           |           |
10.24.80.112[0/1]                   |
+----------------+-------------+-------------+------------+-----------+-----------
+-----------------------------------+
| VIM2           | 0           | 0           | 2-4080     | 200       | True      |
10.24.80.112[0/2]                   |
|                |             |             |            |           |           |
10.24.80.113[0/1]                   |
|                |             |             |            |           |           |
10.24.80.114[0/1]                   |
+----------------+-------------+-------------+------------+-----------+-----------
+-----------------------------------+
*: VNIs will be allocated from the default-tenant VNI pool, for new tenant and default-
tenant network.
--- Time Elapsed: 23.9927ms ---
```

Configure EFA Mode

Procedure

1. Determine the EFA mode.
   - Secure mode - `https`
   - Standard mode - `http`
2. Configure the required mode in the `ml2_conf_extreme.ini` file.

```
efa_rest_token = eyJhbGciOiJSUzI1NiIsImtpZCI6IjEuMCIsInR5cCI6IkpXVCJ9 <<<<<Complete
token String>>>>>>
efa_cert_file = /usr/local/share/ca-certificates/extreme-ca-chain.crt
efa_secure_mode = True
#efa_secure_mode = False
#https
efa_port = 443
#http
#efa_port = 80
efa_host = efa.extremenetworks.com

region_name = RegionOne
fabric_name = Fabric1
[efa_topology]
efa_link_mapping_file = /home/administrator/networking-extreme/link.csv
```

   For description of tokens, see Token Description on page 27.

Token Description

Token	Description
`efa_rest_token`	Authentication token used for EFA
`efa_cert_file`	The location of the CA Cert Chain for EFA
`efa_secure_mode`	True (for secure connections)
`efa_port`	Port running the HTTP service
`efa_host`	IP address of EFA service
`region_name`	Used as the Tenant name by EFA
`region_shared`	Name of the Shared Tenant as created on EFA
`efa_link_mapping_file`	Mapping of Physical NICs to Extreme Switch Ports

Add Certificate to OpenStack Controller

Procedure

1. Copy certificate from EFA to OpenStack Controller.

```
administrator@osController-55:$ su
Password:
root@osController-55:# cd /usr/local/share/ca-certificates
root@osController-55:#
root@osController-55:/usr/local/share/ca-certificates# scp root@10.21.88.130://usr/
local/share/ca-certificates/extreme-ca-chain.crt .
root@10.21.88.130's password:
extreme-ca-chain.crt
100% 4488     4.4KB/s   00:00
root@osController-55:/usr/local/share/ca-certificates#
```

2. Restart Neutron to apply the token and certificate.

```
# sudo service neutron-* restart
```

Configure Neutron to Connect to EFA

Procedure

1. Configure the topology plugin in `/etc/neutron/neutron.conf`.

```
[ml2]
service_plugins = efa_topology_plugin,trunk,segments
```

2. Configure the `mechanism_drivers` in `/etc/neutron/plugins/ml2/ml2_conf.ini`.

```
[ml2]
tenant_network_types = vlan
type_drivers = vlan
mechanism_drivers = openvswitch, extreme_efa
[ml2_type_vlan]
network_vlan_ranges = physnet1:2:500
[ovs]
bridge_mappings = physnet1:br1
```

3. Configure `efa_rest_token`, `region_name`, and `region_shared` in `/etc/neutron/plugins/ml2/ml2_conf_extreme.ini`.

```
[ml2_extreme]
efa_rest_token = <efa_api_token for VIM_1>
```

```
efa_cert_file = /root/gcla/extreme-ca-chain.crt
efa_secure_mode = True
efa_port = 443
efa_host = efa.extremenetworks.com
region_name = VIM_1
region_shared = SHARED_TENANT
#SHARED_TENANT is the name of the shared tenant created on EFA
fabric_name = CNCF
[efa_topology]
efa_link_mapping_file = /home/ubuntu/link.csv
```

For description of tokens, see Token Description on page 27.

Topology Setup in Neutron

OpenStack CLI enables you to manage the topology between compute nodes and SLX devices. Physical Network Topology extension provides a Neutron CLI for the OpenStack administrator to manage links between SLX devices and Compute NICs. Neutron has an abstract for physical network called Provider Network.

Extreme Topology plug-in provides **`efa-link-mapping`** command to configure and setup the topology. The efa-link-mapping feature enables scaling in and scaling out of compute nodes. For more information about scaling in and scaling out of compute nodes, see Scale-in and Scale-out of Compute Nodes on page 44.

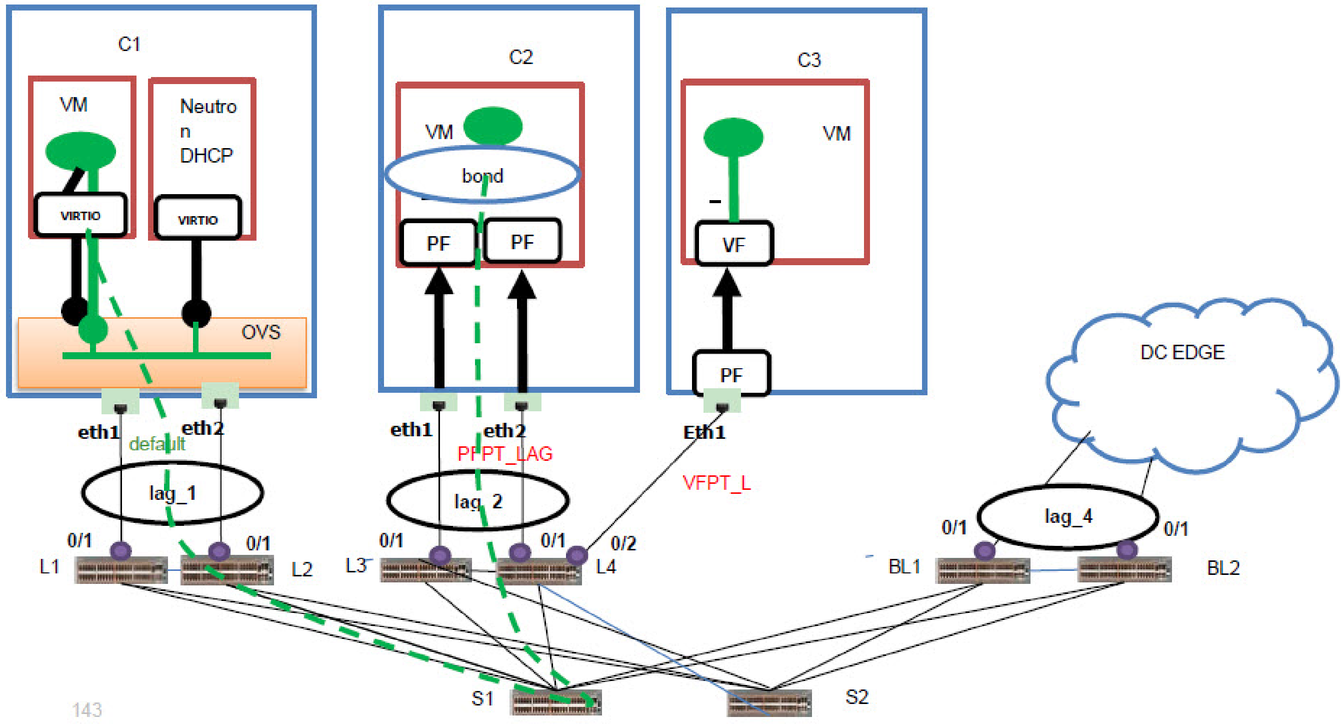


Figure 7: Topology setup in Neutron

Node	IP Address / Description
C1	10.24.51.115
C2	10.24.51.116

Node	IP Address / Description
C3	10.24.51.117
L1	10.24.14.133
L2	10.24.14.134
L3	10.24.14.135
L4	10.24.14.136
BL1	10.24.14.191
BL2	10.24.14.192

Extend Provider Network to External Gateway

About This Task

Perform this procedure to extend provider network to external gateway using port based method. To extend the network using multi-segment based method, see Single and Multi-Segment Networks with DC Edge on page 34.

Procedure

1. Create a provider network.

```
# openstack network create provider-physical-network default --provider-network-type
vlan --provider-segment 1234 dcgw_network1
```

   If the **`provider-physical-network`** is `default`, regular `virtio` ports are brought up on the provider and all corresponding end-points are provisioned.

2. Extend the provider network to external gateway.

```
# openstack port create --network dcgw_network1 --device-owner network:dc_edge --host
DCGW-1  dcgw1_port
```

Configure Topology

About This Task

To configure topology using OpenStack CLI, see Create Topology Using OpenStack CLI on page 30.

Note

All network and server connection settings or mappings can be saved to `csv` files for bulk configuration using the `startup` file option in the `ml2_conf_extreme.ini` file.

Network provisioning and de-provisioning on EFA depends on physnet mapping for non-default physnets. When a node is added or deleted from the mapping file, the corresponding EFA operations are updated.

Procedure

1. Configure the topology.
   - For bulk node configuration, proceed to the next step.
   - For individual node configuration, go to step 4.

2. Save the network settings to a `csv` file.

```
ubuntu@Openstack114:~$ cat /home/ubuntu/link.csv
Openstack115,eth1,default,10.24.14.133,0/1,lag_1
Openstack115,eth2,default,10.24.14.134,0/1,lag_1
Openstack116,eth1,PFPT_LAG,10.24.14.135,0/1,lag_2
Openstack116,eth2,PFPT_LAG,10.24.14.136,0/1,lag_2
Openstack117,eth1,VFPT_L10.24.14.136,0/1
DC-GW1, ,EXT1,10.24.14.191,0/1,lag_4
DC-GW1, ,EXT1,10.24.14.192,0/1,lag_4
```

- The number of comma delimiters used in each line within the csv file must be the same.
- For `DC-Edge` entries, `--nic` must be empty.
- Only links corresponding to VLAN based physnets must be added.
- Only VLAN based Physnet entries must be added.
- Only host names must be added for `--host`. Example: `DC-GW1`

3. Perform bulk configuration.

```
# efa-link-mapping add --file /home/ubuntu/link.csv
```

4. Configure the nodes.

```
# efa-link-mapping add --host Openstack115 --nic eth1 --pn default --switch
10.24.14.133 --port 0/1 --po-name lag_1
# efa-link-mapping add --host Openstack115 --nic eth2 --pn default --switch
10.24.14.134 --port 0/1 --po-name lag_1
# efa-link-mapping add --host Openstack116 --nic eth1 --pn PFPT_LAG --switch
10.24.14.135 --port 0/1 --po-name lag_2
# efa-link-mapping add --host Openstack116 --nic eth2 --pn PFPT_LAG --switch
10.24.14.136 --port 0/1 --po-name lag_2
# efa-link-mapping add --host Openstack117 --nic eth1 --pn VFPT_L --switch
10.24.14.136 --port 0/2
# efa-link-mapping add --host DC-GW1 --pn EXT1 --switc10.24.14.191 --port 0/1 --po-
name lag_4
# efa-link-mapping add --host DC-GW1 --pn EXT1 --switch 10.24.14.192 --port 0/1 --po-
name lag_4
```

Note

If the `LAG` and `PO` are not created on EFA, leave the `--po-name` empty.

The `--host name` in the CSV file and the command must match the `hypervisor hostname` from the **`openstack hypervisor list`** command. If `hypervisor hostname` is a FQDN, then the same must be used here as well.

What to Do Next

Connect OpenStack Instance to EFA Tenant on page 48

Create Topology Using OpenStack CLI

About This Task

The OpenStack CLI commands use Neutron REST extensions provided by EFA.

Procedure

Create the topology link.

```
# openstack network efa-topology-link-map create --host Openstack115--nic eth1 --pn
default --switch 10.24.14.133 --port 0/1 --po-name lag_1
```

Display Link Mapping

About This Task

To display topology link mapping using OpenStack CLI, see Display Topology Mapping Using OpenStack CLI on page 31.

Procedure

Verify link mapping.

```
# efa-link-mapping list
```

Example

```
# sudo efa-link-mapping list
+--------------+------+------------------+-------------+---------+---------+
| Host         | Nic  | provider-network |Switch       | Port    | Po-Name |
+--------------+------+------------------+-------------+---------+----------
|Openstack115  | eth1 | default          |10.24.14.133 |0/1      | lag_1   |
+--------------+------+------------------+-------------+---------+---------+
```

Display Topology Mapping Using OpenStack CLI

About This Task

The OpenStack CLI commands use Neutron REST extensions provided by EFA.

Procedure

Display topology link.

```
# openstack network efa-topology-link-map list

administrator@osController-55:~$ openstack network efa-topology-link-map list
+---------------------------------------+------------------+------------+------------------
+-------------+--------+----------+
| ID                                    | Host             | Nic        | Provider Network
| Switch      | Port   | Po Name  |
+---------------------------------------+------------------+------------+------------------
+-------------+--------+----------+
| 08bb90b6-ac37-4b7f-aa80-d1d08de7e54b | Openstack115     | eth1            |
default          | 10.24.14.133 | 0/1      | lag_1   |
| 012b90b6-ac12-427f-a220-d1d08de7e123 | Openstack115     | eth2            |
default          | 10.24.14.134 | 0/1      | lag_1   |
+---------------------------------------+------------------+------------+------------------
+-------------+--------+----------+
```

Remove Link Mapping

About This Task

To delete topology link mapping using OpenStack CLI, see Delete Topology Link Using OpenStack CLI on page 32.

Procedure

Remove link mappings.

- Remove all the mappings.

  ```
  # sudo efa-link-mapping remove
  ```

- Remove all the mappings for the selected host.

  ```
  # sudo efa-link-mapping remove -H compute1/ip-address
  ```

- Remove the selected mapping.

  ```
  # sudo efa-link-mapping remove -H compute1 -n eth0
  ```

Delete Topology Link Using OpenStack CLI

About This Task

The OpenStack CLI commands use Neutron REST extensions provided by EFA.

Procedure

Delete topology link.

```
# openstack network efa-topology-link-map delete 08bb90b6-ac37-4b7f-aa80-d1d08de7e54b
012b90b6-ac12-427f-a220-d1d08de7e123
```

- Multiple link entries can be removed by providing multiple IDs.
- Multiple IDs can be saved to `csv` file for bulk deletion using the `--file` option. The `--file` option is available only in the console version of the command.

ML2 Mechanism Driver

Modular Layer 2 (ML2) Neutron plug-in allows OpenStack networking to simultaneously utilize multiple layer 2 networking technologies such as 802.1Q and VXLAN for virtual instances.

External networks are managed using Mechanism Drivers. The Mechanism Driver is responsible for taking the information established by the Type Driver and ensuring that it is properly applied given the specific networking mechanisms that are enabled. A single OpenStack installation can use multiple ML2 Mechanism Drivers.

Extreme ML2 Mechanism driver within Neutron initiates Neutron API calls for network management. OpenStack Service in EFA translates the Neutron network management calls to appropriate tenant API calls and provisions the fabric with appropriate L2 networking constructs.

The following figure shows an overview of the ML2 Mechanism Driver within Neutron.

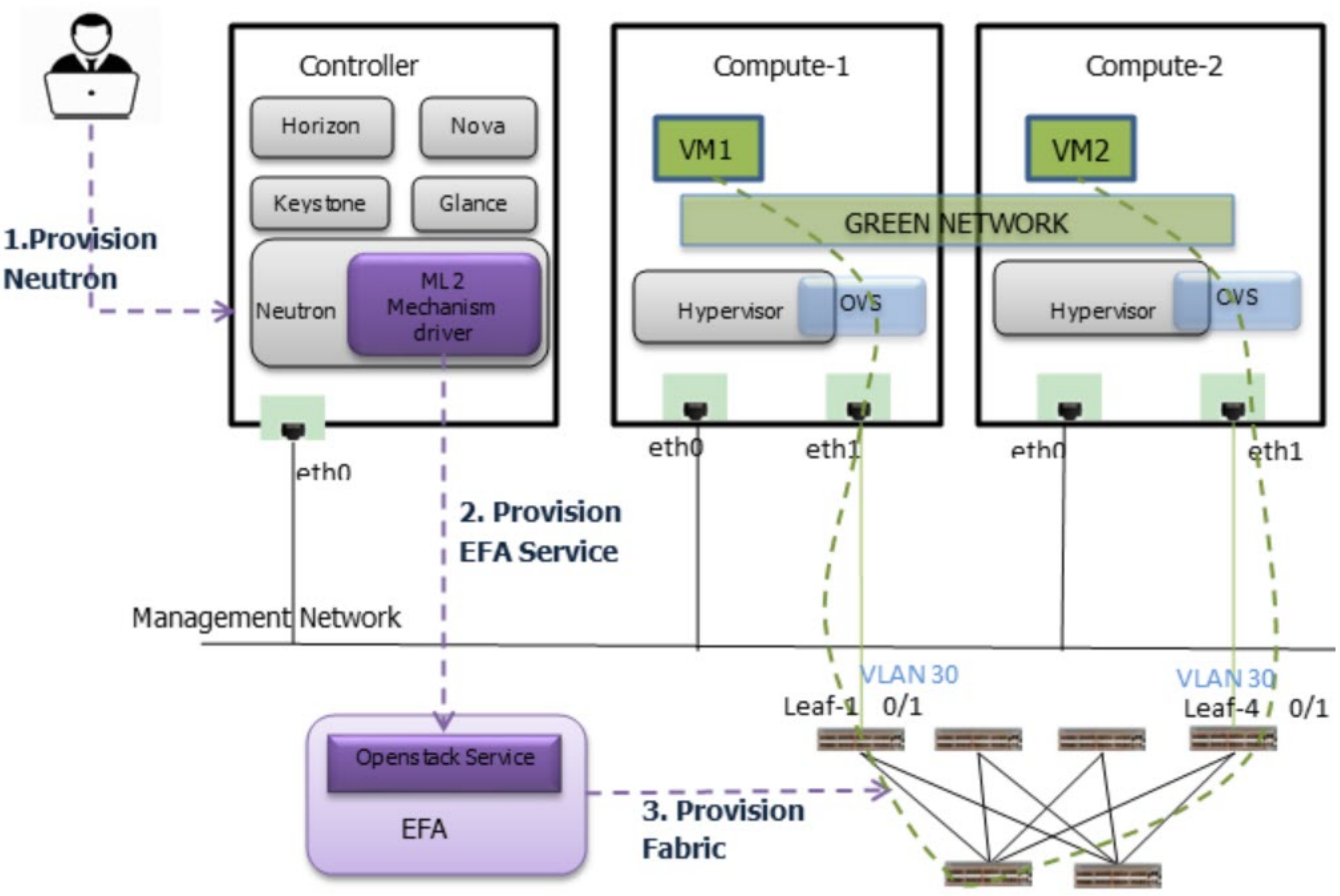


Figure 8: Overview of ML2 Mechanism Driver within Neutron

The following table shows an example mapping of the OpenStack Networks to the Tenant Service EPG constructs.

EPG provisioning on the fabric creates a L2 network on the fabric spanning VM1 and VM2 with necessary fabric mappings. This creates the necessary constructs to establish an end-to-end connectivity between DB1 and DB2.

The following table shows VLAN provisioning based on Provider Networks (Physnet).

Table 5: VLAN provisioning based on Provider Networks (Physnet)

Provider Network (Physnet)	VLAN Provisioning by ML2
default	VLAN provisioning of end-points is done when Neutron ports are bound to a host or compute node. Example: • Neutron port is bound to a host when VM is launched. • Neutron port is bound to a host (Neutron controller) when DHCP starts.
default (device-owner=dc-edge –host=<DCGW>)	VLAN provisioning for default physnet is done during port creation based on additional parameters passed during port create call.
non-default (PFPT_L, EXT1)	VLAN provisioning is done during network creation or segment creation of single segment or multiple segment.

The Endpoints are de-provisoned during the following negation operations:

- `Virtio Ports` are unbounded from a host
- `Port Delete` operations on `device-owner=dc-edge` qualified ports
- Deletion of Networks or Segments on non-default physnets

Single and Multi-Segment Networks with DC Edge

The following sections describe how to create single and multi-segment networks that extend to DC Edge.

Create Single Segment Virtio VM Network

About This Task

Perform this procedure to create a single segment Virtio VM network on default physnet and extend the network to DC Edge.

Note

VLAN provisioning depends on the following:

- Neutron Virtio ports are bound to a host during VM launch
- DHCP ports are bound during subnet creation
- DC Edge extension is achieved using explicit port-create

The following figure shows an overview of a single segment Virtio VM Network with DC Edge.

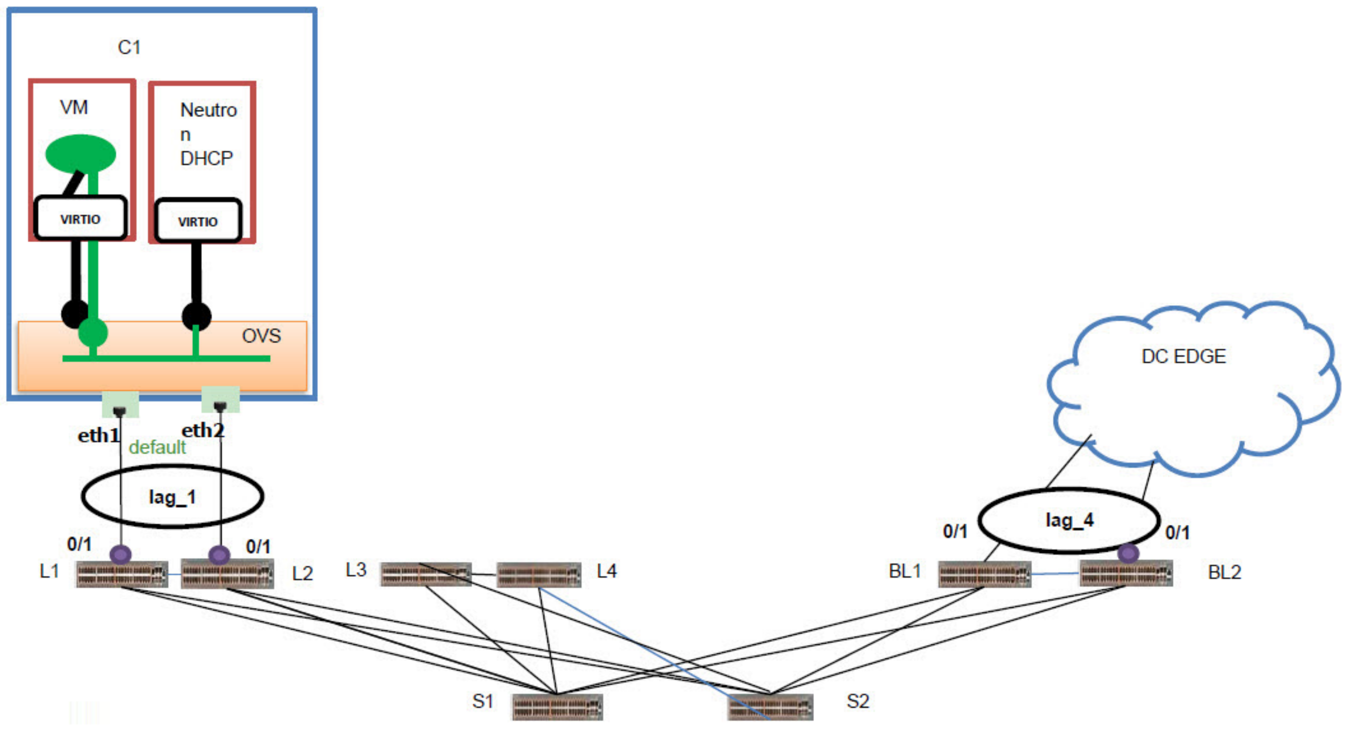


Figure 9: Overview of single segment Virtio VM network

Procedure

1. Create an OpenStack network on default physnet.

   ```
   # openstack network create --provider-network-type vlan --provider-physical-network
   default --provider-segment 3320 ss1network1
   ```

   EPG is created for `ss1network`:

   - Name: 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d
   - CTAG: 3320
   - Neutron UUID allocated for the EPG: 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d

2. Create DHCP End Points on EPG corresponding to `ss1network1`.

   ```
   # openstack subnet create ss1subnet1 --network ss1network1 --subnet-range 10.1.1.0/24
   ```

   VLAN is provisioned.

3. Update EPG.

   ```
   # openstack subnet create ss1subnet1ipv6 --network ss1network1 --ip-version 6 --ipv6-
   address-mode=dhcpv6-stateful --subnet-range fd00:10:0:57::1000/64
   ```

   EPG is updated and port `lag_1` is added.

4. Create OpenStack ports and network trunk.

   ```
   # openstack port create ss1VirtIoTrunkPort1 --network ss1network1

   #openstack network trunk create --parent-port ss1VirtIoTrunkPort1 ss1VirtIoTrunk1

   # openstack port create ss1VirtIoSubport1 --network ss1network1
   ```

5. Create EPG for `SS1network2`.

   ```
   # openstack network create --provider-network-type vlan --provider-physical-network
   default --provider-segment 3321 ss1network2
   ```

   EPG is created for `ss1network2`:

   - Name: 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d
   - CTAG: 3321
6. Create DHCP End Points on EPG corresponding to ss1network2.

   ```
   # openstack subnet create ss1subnet2 --network ss1network2 --subnet-range 11.1.1.0/24
   ```

   VLAN is provisioned.
7. Update EPG.

   ```
   # openstack subnet create ss1subnet2ipv6 --network ss1network2 --ip-version 6 --ipv6-
   address-mode=dhcpv6-stateful --subnet-range fd00:11:0:57::1000/64
   ```

   EPG is updated:

   - Name: 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d
   - Port: lag_1 (added)
8. Create an OpenStack port.

   ```
   # openstack port create ss1VirtIoTrunkPort2 --network ss1network2
   ```
9. Create an OpenStack network trunk.

   ```
   # openstack network trunk create --parent-port ss1VirtIoTrunkPort2 ss1VirtIoTrunk2
   ```
10. Create an OpenStack network trunk set.

    ```
    # openstack network trunk set --subport port=ss1VirtIoSubport1,segmentation-
    type=vlan,segmentation-id=3801 ss1VirtIoTrunk2
    ```
11. Create an OpenStack server.

    ```
    # openstack server create --flavor m1.large --image ubuntu --port $(neutron port-list
    | grep -w 'ss1VirtIoTrunkPort1' | awk '{print $2}') ss1VirtIoVM1 --availability-zone
    nova:Openstack116
    ```

    Endpoint corresponding to `ss1VirtIoTrunkPort1` is added to EPG (ss1nework1) and VLAN is provisioned.

    EPG is updated:

    - Name: 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d
    - Port: lag_1

12. Update EPG.

    ```
    # openstack server create --flavor m1.large --image ubuntu --port $(neutron port-list
    | grep -w 'ss1VirtIoTrunkPort2' | awk '{print $2}') ss1VirtIoVM2 --availability-zone
    nova:Openstack117
    ```

    Endpoint corresponding to `ss1VirtIoTrunkPort2` is added to `EPG(ss1nework2)` and VLAN is provisioned.

    EPG is updated:

    - Name: 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d
    - Port: lag_1

    Endpoint corresponding to `ss1VirtIoSubport1` is added to `EPG(ss1nework1)` and VLAN is provisioned.

    EPG is updated:

    - Name: 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d
    - Port: lag_1

13. Extend `ss1network1` to DC Edge.

    ```
    # openstack port create ss2DcGwPort1 --device-owner network:dc_edge --host DCGW-1 --
    network ss1network1
    ```

    Endpoint corresponding to `host DCGW-1` is added to `EPG (ss1network1)` and VLAN is provisioned.

    EPG is updated:

    - Name: 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d
    - Port: lag_1, lag_4 (added)

14. Extend `ss1network2` to DC Edge.

    ```
    # openstack port create ss2DcGwPort1 --device-owner network:dc_edge --host DCGW-1 --
    network ss1network2
    ```

    Endpoint corresponding to `host DCGW-1` is added to `EPG (ss1network2)` and VLAN is provisioned.

    EPG updated:

    - Name: 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d
    - Port: lag_1, lag_4 (added)

Create Single Segment VFPT VM Network

About This Task

Perform this procedure to create a single segment VFPT VM network on non-default physnet and extend the network to DC Edge.

Note

VLAN provisioning depends on the following:

- VFPT ports are bound to a host when the VFPT Subport is attached to the trunk port
- DC Edge extension is achieved using explicit port-create

The following figure shows an overview of single segment VFPT VM network with DC Edge.

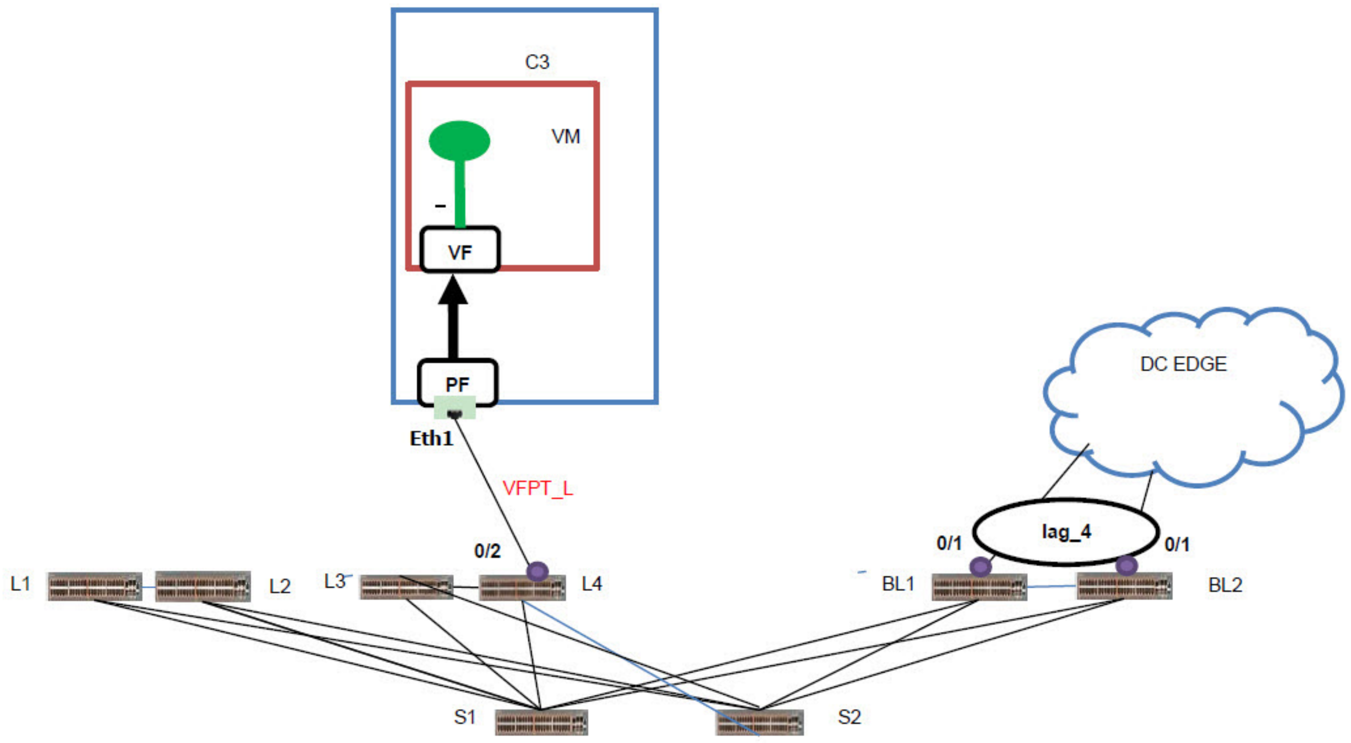


Figure 10: Overview of single segment VFPT VM network with DC Edge

Procedure

1. Create OpenStack networks, `VFPT_L` and `VFPT_R`.

```
# openstack network create --provider-physicalnetwork VFPT_L --provider-network-type
flat ss9_vfpt_flat_left

# openstack network create --provider-physicalnetwork VFPT_R --provider-network-type
flat ss9_vfpt_flat_right
```

   No EFA impact as network type is `flat`.

2. Create an OpenStack subnet.

```
# openstack subnet create ss8flatleftsubnet --network ss9_vfpt_flat_left --no-dhcp --
subnet-range 107.1.1.0/24
```

3. Create an OpenStack port.

```
# openstack port create --network ss9_vfpt_flat_left --vnic-type direct
ss9_port_vfpt_left1 openstack network trunk create --parent-port ss9_port_vfpt_left1
ss9SriovTrunkLag1
```

4. Create an OpenStack server.

```
# openstack server create --flavor myhuge --image ubuntu --port $(neutron port-list |
grep -w 'ss9_port_vfpt_left1' | awk '{print $2}') ss9SriovVM1 --availability-zone
nova:compute-0-10.domain.tld --poll
```

5. Create an OpenStack network.

   ```
   # openstack network create --provider-network-type vlan --provider-physical-network
   VFPT_L --providersegment 3390 ss9network
   ```

   EPG is created for `ss9network`:

   - Name: 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d
   - CTAG: 3390
   - Neutron UUID allocated for the EPG: 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d
6. Create OpenStack subnets, `ss9subnet1` and `ss9subnet1ipv6`.

   ```
   # openstack subnet create ss9subnet1 --network ss9network1 --no-dhcp --subnet-range
   90.1.1.0/24

   # openstack subnet create ss9subnet1ipv6 --network ss8network1 --ip-version 6 --no-
   dhcp --subnet-range fd00:90:0:57::1000/64
   ```

   No EFA impact as `--no-dhcp` option is used.
7. Create an OpenStack port.

   ```
   # openstack port create ss9SriovSubPort1 --network ss9network1 --mac-address <same-mac-
   asss9_port_vfpt_left1> --vnic-type direct --fixed-ip subnet=ss9subnet1,ip-
   address=90.1.1.10 --fixed-ip subnet=ss9subnet1ipv6,ip-address=fd00:90:0:57::10
   ```
8. Create an OpenStack network trunk set.

   ```
   openstack network trunk set --subport
   port=ss9SriovSubPort1,segmentationtype=vlan,segmentation-
   id=3390ss9SriovTrunkLagopenstack port setss9SriovSubPort1 --device-owner compute:nova
   --host compute-0-10.domain.tld --device <samedeviceid-as-ss9_port_vfpt_left1> --
   binding-profile pci_slot=<same-as-slot-of-ss9_port_vfpt_left1> --binding-profile
   pci_vendor_info=<same-as-vendorof-ss9_port_vfpt_left1> --binding-profile
   physical_network=<same-as-physnet-ofss9_port_vfpt_left1>
   ```

   End point corresponding to `ss9SriovSubPort1` is added to `EPG(ss9network1)` and VLAN is provisioned.

   EPG is updated:

   - Name: 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d
   - Port: L4[0/2] (added)
9. Extend `ss9network1` to DC Edge.

   ```
   # openstack port create ss9DcGwPort --network ss9network1 --device-owner
   network:dc_edge --host DCGW-1
   ```

   End Point corresponding to `host DCGW-1` is added to `EPG (ss9network1)` and VLAN is provisioned.

   EPG is updated:

   - Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d
   - Port = L4[0/2]. lag_4 (added)

Create Single Segment PFPT VM Network

About This Task

Perform this procedure to create a single segment PFPT VM network on non-default physnet and extend the network to DC Edge.

The following figure shows an overview of single segment PFPT VM network with DC Edge.

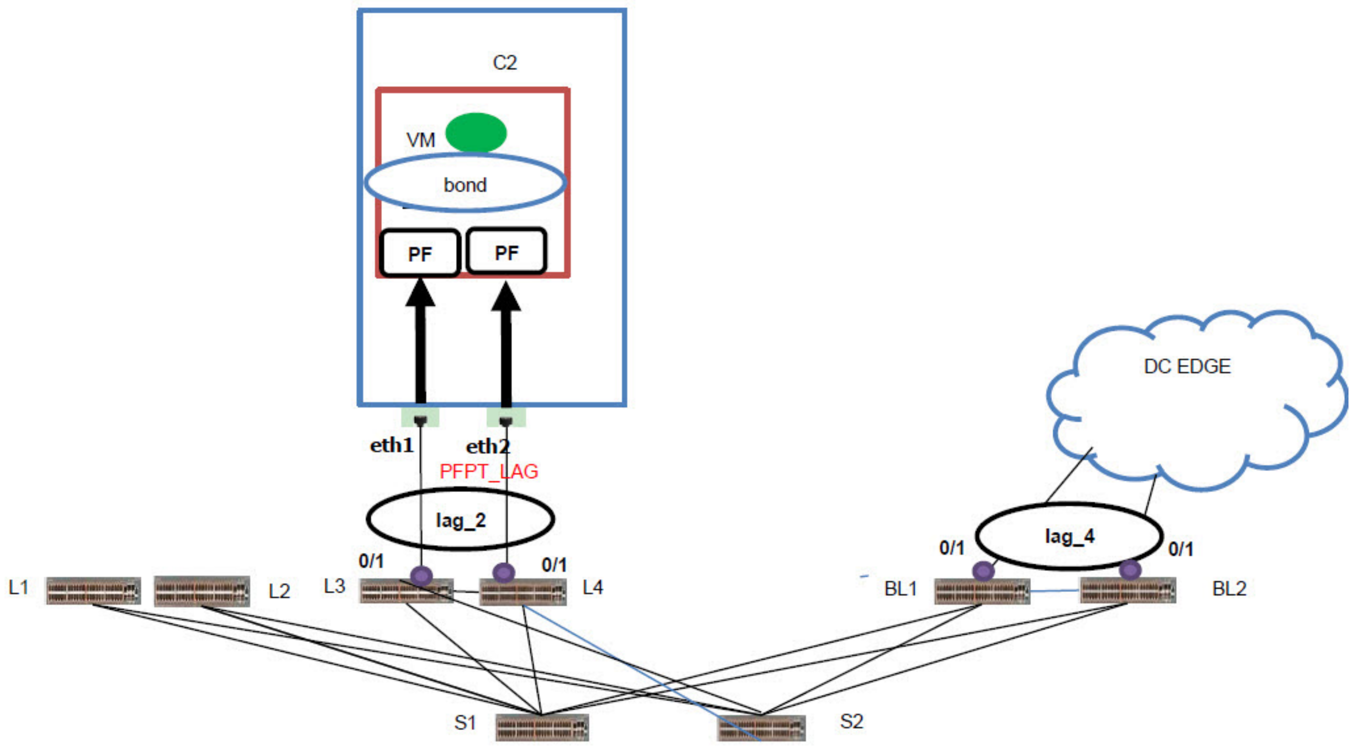


Figure 11: Overview of single segment PFPT VM network with DC Edge

Procedure

1. Create OpenStack physnet networks, `PFPT_L` and `PFPT_R`.

```
# openstack network create --provider-physical-network PFPT_L --provider-network-type
flat ss7_pfpt_flat_left

# openstack network create --provider-physical-network PFPT_R --provider-network-type
flat ss7_pfpt_flat_right
```

2. Create OpenStack subnets.

```
# openstack subnet create ss7flatleftsubnet --network ss7_pfpt_flat_left --no-dhcp --
subnet-range 107.1.1.0/24

# openstack subnet create ss7flatrightsubnet --network ss7_pfpt_flat_right --no-dhcp --
subnetrange 108.1.1.0/24
```

3. Create OpenStack ports.

```
# openstack port create --network pfpt_flat_left --vnic-type direct-physical
ss7_port_pfpt_left

# openstack port create --network pfpt_flat_right --vnic-type direct-physical
ss7_port_pfpt_right
```

4. Create an OpenStack network trunk.

```
# openstack network trunk create --parent-port ss7_port_pfpt_left ss7PFPTTrunkLag1
```

5. Create an OpenStack server.

```
# openstack server create --flavor myhuge --image ubuntu --port $(neutron port-list |
grep -w 'ss7_port_pfpt_left' | awk '{print $2}') --port $(neutron port-list | grep -w
'ss7_port_pfpt_right' | awk '{print $2}') ss7PFPTLAGVM1 --availability-zone
nova:compute-0-10.domain.tld --poll
```

6. Create an OpenStack network, `ss7network1`.

   ```
   # openstack network create --provider-network-type vlan --provider-physical-network
   PFPT_LAG --provider-segment 3370 ss7network1
   ```

   EPG is created for `ss7network1`:

   - Name: 74cbf489-f3d9-41c7-bbb2-6cb7df33da6
   - CTAG: 3370
   - Neutron UUID allocated for the EPG: 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d

7. Create OpenStack subnets `ss7subnet1` and `ss7subnet1ipv6`.

   ```
   # openstack subnet create ss7subnet1 --network ss7network1 --no-dhcp --subnet-range
   70.1.1.0/24

   # openstack subnet create ss7subnet1ipv6 --network ss7network1 --ip-version 6 --no-
   dhcp --subnet-range fd00:70:0:57::1000/64
   ```

   No EFA impact as `--no-dhcp` option is used.

8. Create an OpenStack port.

   ```
   # openstack port create ss7PFPTSubPort1 --network ss7network1 --mac-address <same-mac-
   asss7_port_pfpt_left> --vnic-type direct-physical --fixed-ip subnet=ss8subnet1,ip-
   address=70.1.1.10 --fixed-ip subnet=ss7subnet1ipv6,ip-address=fd00:70:0:57::10
   ```

9.
   ```
   #openstack network trunk set --subport port=ss7PFPTSubPort1,segmentation-
   type=vlan,segmentation-id=3370
   ```

   Endpoint corresponding to `ss7PFPTSubPort1` is added to `EPG(ss7network1)` and VLAN is provisioned.

   EPG Updated:

   - Name: 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d
   - Port: lag_2 (added)

10. Extend `ss9network1` to DC Edge.

    ```
    # openstack port create ss7DcGwPort --network ss7network1 --device-owner
    network:dc_edge --host DCGW-1 --fixed-ip subnet=ss7subnet1,ip-address=70.1.1.30 --
    fixed-ip subnet=ss7subnet1ipv6,ipaddress=fd00:70:0:57::30
    ```

    Endpoint corresponding to `host DCGW-1` is added to `EPG(ss9network1)` and VLAN is provisioned.

    EPG Updated:

    - Name: 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d
    - Port: lag_2, lag_4 (added)

Create Virtio VM, PFPT, and VFPT Segment Network

About This Task

Perform this procedure to create a network on default physnet with Virtio VM, PFPT, and VFPT segments and extend the network to DC Edge.

The following figure shows an overview of Virtio VM, PFPT, and VFPT segment network with DC Edge.

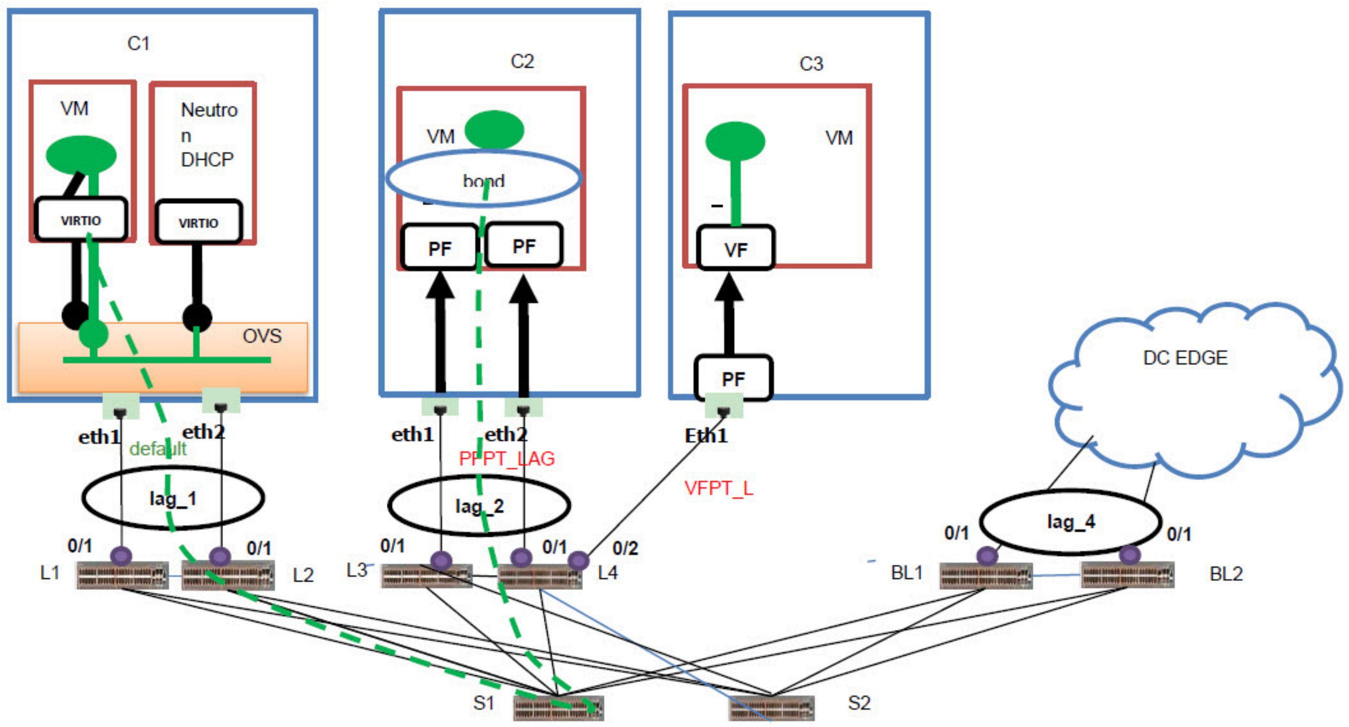


Figure 12: Overview of Virtio VM, PFPT, and VFPT segment network with DC Edge

Procedure

1. Create an OpenStack network, `VFPT_L`.

```
# openstack network create --provider-network-type vlan --provider-physical-network
VFPT_L --provider-segment 3710 ms10Network1
```

   EPG is created for `ms10Network1`:

   - Name: 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d
   - CTAG: 3710
   - Neutron UUID allocated for the EPG: 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d

2. Create an OpenStack network segment.

```
# openstack network segment create --network-type vlan --physical-network EXT1 --
segment 3710 --network ms10Network1 ms10Network1DcgwSegment
```

   Endpoint corresponding `EXT1` added to `EPG(ms10Network1)` and VLAN is provisioned.

   EPG is updated:

   - Name: 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d
   - Port = L4[0/2] , lag_4 (added)

3. Create an OpenStack network segment.

   ```
   # openstack network segment create --network-type vlan --physical-network PFPT_LAG --
   segment 3710 --network ms10Network1 ms10NetworkPFPTSegment
   ```

   Endpoint corresponding `PFPT_LAG` added to `EPG(ms10Network1)` and VLAN is provisioned.

   EPG is updated:

   - Name: 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d
   - Port = L4[0/2], lag_4, lag_2 (added)
4. Create an OpenStack network segment on default physnet.

   ```
   # openstack  network segment create --network-type vlan --physical-network default  --
   segment 3710 --network ms10Network1 ms10Network1DefaultSegment
   ```
5. Create OpenStack subnets.

   ```
   # openstack subnet create ms10subnet1 --network ms10Network1 --subnet-range
   130.1.1.0/24

   # openstack subnet create ms10subnet1ipv6 --network ms10Network1 --ip-version 6 --ipv6-
   address-mode=dhcpv6-stateful --subnet-range fd00:0130:0:57::1000/64
   ```
6. Create OpenStack ports, `ms10VirtIoPort1`, `ms10SriovPort2`, and `ms10PFPTport3`.

   ```
   openstack port create ms10VirtIoPort1 --network ms10Network1 --vnic-type normal

   # openstack port create ms10SriovPort2 --network ms10Network1 --vnic-type direct

   # openstack port create ms10PFPTport3 --network ms10Network1 --vnic-type direct-
   physical
   ```
7. Create OpenStack Server, `ms10VirtIoVM1`.

   ```
   # openstack server create --flavor myhuge --image ubuntu --port $(neutron port-list |
   grep -w 'ms10VirtIoPort1' | awk '{print $2}') ms10VirtIoVM1 --availability-zone
   nova:compute-0-5.domain.tld
   ```

   Endpoint corresponding to default physnet is added to EPG(ms10Network1) and VLAN is provisioned.

   EPG Updated:

   - Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d
   - Port = L4[0/2],lag_4,lag_2 (added)
8. Create OpenStack Servers, `ms10SrIovVM2` and `ms10PFPTVM3`.

   ```
   # openstack server create --flavor myhuge --image ubuntu --port $(neutron port-list |
   grep -w 'ms10SriovPort2' | awk '{print $2}') ms10SrIovVM2 --availability-zone
   nova:compute-0-1.domain.tld

   # openstack server create --flavor myhuge --image ubuntu --port $(neutron port-list |
   grep -w 'ms10PFPTport3' | awk '{print $2}') ms10PFPTVM3 --availability-zone
   nova:compute-0-7.domain.tld
   ```

Scale-in and Scale-out of Compute Nodes

Table 6: Scale-in of compute nodes

Scale-in	EFA Impact
Link removal from default physical network	VLAN provisioning for end-points is done when Neutron ports are unbound to a host/compute. Hence, removing a link which belongs to default physical network does not have any impact, only link-mapping table is updated.
Link removal from non-default physical networks	For non-default physical networks, VLAN provisioning is done at network level. Hence, the links that are removed from non-default networks are automatically removed from the existing networks. On EFA, interface is removed from the corresponding EPGs. If there are any errors during EFA unbinding, the operation succeeds. This behavior is consistent with delete port operation.
Links bound to port-channels on non-default physical networks	Removing a link which maps to LAG interface does not affect EPG unless the last member of the LAG is removed in link mapping. For example, LAGs have two entries in the link mapping. Removing only one entry does not affect the existing EPG. Only when the last entry for that LAG is removed, EFA removes the LAGs from corresponding EPGs.

Table 7: Scale-out of compute nodes

Scale-out	EFA Impact
Link addition to default physical network	VLAN provisioning of end-points is done when Neutron ports are bound to a host/Compute. Hence, adding a link which belongs to default physical network does not have any impact, Only link-mapping table is updated.
Link addition to non-default physical networks	For non-default physical networks, VLAN provisioning is done during network-create time. Hence, the new links that are added to non-default networks are automatically added to the existing networks. On EFA, the interface is added to the corresponding EPGs. If there are any errors during EFA binding, the whole operation fails.

Table 7: Scale-out of compute nodes (continued)

Scale-out	EFA Impact
Links bound to port-channels on non-default physical networks	If the switch link is a LAG interface, as soon as the first link to LAG is added, the corresponding LAG is added to the EPGs. Further link addition on that LAG is no-op from EFA perspective.
File option in efa-link-add	File option enables saving link mappings in a file for bulk configuration. If there is an error in adding a link during file replay, the operation is aborted. You can correct the errors and replay it.

Virtual Machine Migration

Virtual Machine Migration (VMotion) enables migration of live virtual machines from one OpenStack Compute server to another. You can use VMotion to migrate VMs during planned maintenance or redistribute load on the server.

In non-live or cold migration, the VM instances are shutdown before migrating them to another server resulting in disruption of services. In live migration, the VM instances continue to run during migration without disrupting any services.

Enable VMotion

Procedure

1. Set the following parameters in `nova.conf` on all Compute nodes. Ensure that `instances_path` and `restorecon` are same for all Compute nodes.

   Note
   This setting allows VNC clients from any IP address to connect to instance consoles. Ensure to take additional measures to secure networks.

   ```
   vncserver_listen = 0.0.0.0
   instances_path = /var/lib/nova/instances
   restorecon = /etc/hosts
   ```
2. Enable password-less SSH.
   The libvirt daemon that runs as root uses SSH protocol to copy the instance to the destination.
3. Configure the firewalls to allow libvirt to communicate with Compute nodes.
   The default libvirt TCP port range is 49152 to 49261 for copying memory and disk contents.

Migrate Virtual Machines

Procedure

1. View the OpenStack Server list to determine the VM to be migrated.
   ```
   # openstack server list
   ```

```
+--------------------------------------+------+--------+-----------------+------------+
| ID                                   | Name | Status | Networks        | Image Name |
+--------------------------------------+------+--------+-----------------+------------+
| d1df1b5a-70c4-4fed-98b7-423362f2c47c | vm1  | ACTIVE | private=a.b.c.d | ...        |
| d693db9e-a7cf-45ef-a7c9-b3ecb5f22645 | vm2  | ACTIVE | private=e.f.g.h | ...        |
+--------------------------------------+------+--------+-----------------+------------+
```

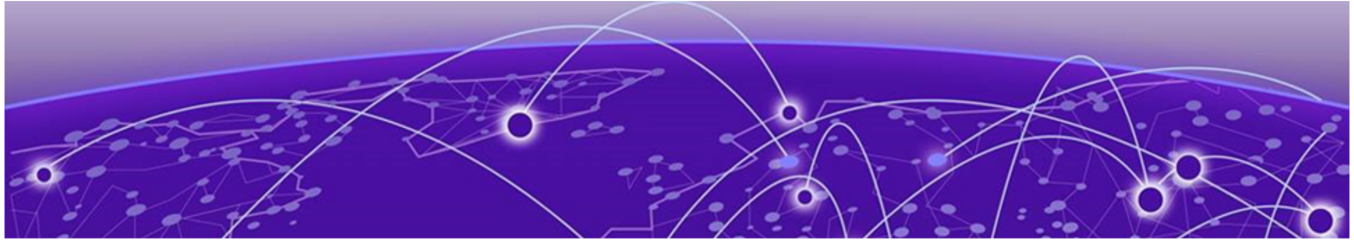
2. Select the destination host.
   - For manual selection of the destination host, proceed to the next step.
   - For automatic selection of the destination host, go to Step 6.
3. View the server details of the selected VM.

```
# openstack server show d1df1b5a-70c4-4fed-98b7-423362f2c47c

+----------------------+--------------------------------------+
| Field                | Value                                |
+----------------------+--------------------------------------+
| ...                  | ...                                  |
| OS-EXT-SRV-ATTR:host | HostB                                |
| ...                  | ...                                  |
| addresses            | a.b.c.d                              |
| flavor               | m1.tiny                              |
| id                   | d1df1b5a-70c4-4fed-98b7-423362f2c47c |
| name                 | vm1                                  |
| status               | ACTIVE                               |
| ...                  | ...                                  |
+----------------------+--------------------------------------+
```

4. Determine the destination host to migrate the selected VM.

```
# openstack compute service list

+----+------------------+-------+----------+---------+-------
+----------------------------+
| ID | Binary           | Host  | Zone     | Status  | State | Updated
At                  |
+----+------------------+-------+----------+---------+-------
+----------------------------+
|  3 | nova-conductor   | HostA | internal | enabled | up    |
2017-02-18T09:42:29.000000 |
|  4 | nova-scheduler   | HostA | internal | enabled | up    |
2017-02-18T09:42:26.000000 |
|  5 | nova-consoleauth | HostA | internal | enabled | up    |
2017-02-18T09:42:29.000000 |
|  6 | nova-compute     | HostB | nova     | enabled | up    |
2017-02-18T09:42:29.000000 |
|  7 | nova-compute     | HostC | nova     | enabled | up    |
2017-02-18T09:42:29.000000 |
+----+------------------+-------+----------+---------+-------
+----------------------------+

# openstack host show HostC

+-------+------------+-----+-----------+---------+
| Host  | Project    | CPU | Memory MB | Disk GB |
+-------+------------+-----+-----------+---------+
| HostC | (total)    |  16 |     32232 |     878 |
| HostC | (used_now) |  22 |     21284 |     422 |
| HostC | (used_max) |  22 |     21284 |     422 |
| HostC | p1         |  22 |     21284 |     422 |
| HostC | p2         |  22 |     21284 |     422 |
+-------+------------+-----+-----------+---------+
```

5. Migrate the VM instance.
   - Manual selection of the destination host:
     ```
     # openstack server migrate d1df1b5a-70c4-4fed-98b7-423362f2c47c --live HostC
     ```
   - Automatic selection of the destination host:
     ```
     # nova live-migration d1df1b5a-70c4-4fed-98b7-423362f2c47c
     ```
6. View the host server details of the migrated VM to confirm the status of migration. If migration fails, go to Step 8.
   ```
   # openstack server show d1df1b5a-70c4-4fed-98b7-423362f2c47c

   +----------------------+--------------------------------------+
   | Field                | Value                                |
   +----------------------+--------------------------------------+
   | ...                  | ...                                  |
   | OS-EXT-SRV-ATTR:host | HostC                                |
   | ...                  | ...                                  |
   +----------------------+--------------------------------------+
   ```
7. (Optional) View the log files on the Controller and Compute nodes for more information about migration failure.
   - `nova-scheduler`
   - `nova-conductor`
   - `nova-compute`
8. (Optional) Stop the migration manually.
   ```
   # nova live-migration-abort INSTANCE_ID MIGRATION_ID
   ```
9. (Optional) Force complete the migration.
   ```
   # nova live-migration-force-complete INSTANCE_ID MIGRATION_ID
   ```



EFA Configuration

Configure IP Fabric

About This Task

For more information about EFA configuration, see *Extreme Fabric Automation Administration Guide, 2.3.0*. For more information about EFA Commands and parameters, see *Extreme Fabric Automation Command Reference, 2.3.0*.

Procedure

1. Log in to EFA.
2. Add devices to the fabric.

```
# efa fabric device add-bulk --name --ip device-ip --leaf --border-leaf <hostname> --
three-stage-pod --five-stage-pod --spine --super-spine --username --password --rack
```

3. Configure the fabric.

```
# efa fabric configure --name default
```

4. Create the tenant.

```
# efa tenant create --name --description --type < private | shared > --l2-vni-range
--l3-vni-range --vlan-range --vrf-count --enable-bd --port
```

5. Update the tenant.

```
# efa tenant update --name tenant-name --description tenant-description --l2-vni-range
--l3-vni-range --vlan-range --vrf-count --enable-bd --operation desc-update < vni-
update | port-add | port-delete | vlan-add | vlan-delete | vlan-update | num-vrf-
update | enable-bd-update > --port--force
```

Connect OpenStack Instance to EFA Tenant

Procedure

1. Create tenant `VIM_1`.

```
# efa tenant create --name VIM_1 --vlan-range 2-4090 --l2-vni-range 1-5000 --port
10.24.14.133.[0/1], 10.24.14.134.[0/1],10.24.14.135.[0/1],10.24.14.136.[0/1-2]
```

2. Create `lag_1` on Compute 1.

```
# efa tenant po create --name lag_1 --tenant VIM_1 --port 10.24.14.133.[0/1],
10.24.14.134.[0/1] --speed 10Gbps --negotiation active
```

3. Create `lag_2` on Compute 2.

```
# efa tenant po create --name lag_2 --tenant VIM_1 --port 10.24.14.135.[0/1],
10.24.14.136.[0/1] --speed 10Gbps --negotiation active
```

4. Create `SHARED_TENANT` on the border leaf.

```
# efa tenant create --name SHARED_TENANT --port BL-1[0/1], BL-2[0/1] --role shared

# efa tenant create --name SHARED_TENANT --port 10.24.14.191.[0/1], 10.24.14.192.[0/1]
-type shared
```

5. Create a border leaf LAG, `lag_4` on the shared tenant.

```
# efa tenant po create --name lag_4 --port BL1[0/1],BL2[0/1] --speed  10Gbps --
negotiation active --tenant SHARED_TENANT --lacp-timeout short

# efa tenant po create --name lag_4 --tenant SHARED_TENANT --port 10.24.14.133.[0/1],
10.24.14.134.[0/1] --speed 10Gbps --negotiation active
```

6. Register `VIM_1` tenant for OpenStack access.

```
# efa auth client register --name VIM_1 --type openstack
```

7. Generate the `auth apikey` using the Client ID generated in client register.

```
# efa auth apikey generate --client-id <client_id>
```

8. Configure the generated `auth apikey` as `efa_rest_token` in OpenStack.

Verify EFA Health

You can use the **`efa-health show`** command to verify connectivity with EFA.

About This Task

Tip
Running this command creates a VRF for testing the status. The output of the **`efa tenant show`** command shows the extra VRF in the VRF-Count field.

Procedure

Verify EFA connectivity.

```
$> efa-health show

EFA Host : efa.extremenetworks.com
EFA Status : UP
```

Example

This example shows the output when the status is DOWN.

```
$ efa-health show

EFA Host : efa.extremenetworks.com
EFA Status : DOWN
Reason : EFA host is not reachable
```

Sync EFA with Neutron

The efa-sync tool on Neutron provides the **`efa-sync`** command.

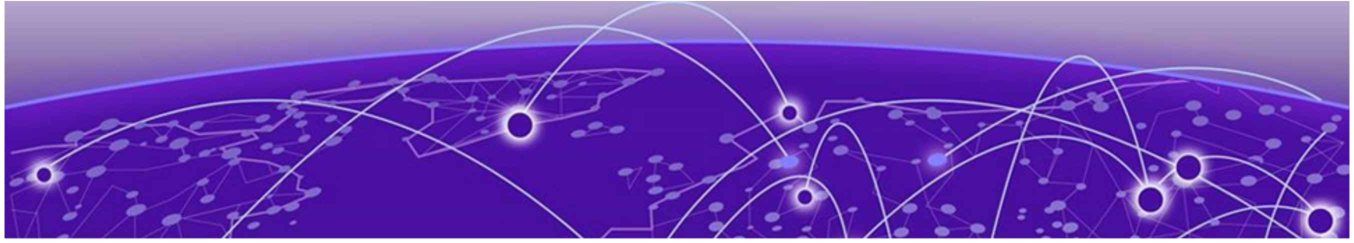
About This Task

If there are stale or missing entries in EFA, use the **`efa-sync`** command to sync all entries from Neutron to EFA.

Procedure

Sync EFA with Neutron:

```
# efa-sync execute
```



EFA OpenStack Service Command Reference

efa openstack debug

Displays OpenStack debug information.

Syntax

efa openstack debug [**network** | **network-interface** | **tenant** | **router** | **router-interface**]

efa openstack debug network delete *--neutron-id*

efa openstack debug network-interface delete *--neutron-id*

efa openstack debug router delete *--router-id*

efa openstack debug router-interface delete --router-id *<router-id>* **--subnet-id** *<subnet-id>*

efa openstack debug tenant cleanup --name *tenant name*

Parameters

cleanup *--name*

Cleans up all OpenStack assets associated to a tenant

delete *--neutron-id* | *--router-id* | *subnet-id*

Deletes the selected network element

network

Specifies the name of the network

network-interface

Specifies the name of the network-interface

tenant

Specifies the name of the tenant

router

Specifies the name of the router

router-interface

Specifies the name of the router-interface

efa openstack execution

Provides OpenStack execution commands.

Syntax

efa openstack execution delete [**--days int32** | **--help**]

efa openstack execution show [**--help** | **--id** | **--limit int32** | **--status**]

Parameters

delete

Deletes execution entries older than the specified days.

--days int32

Deletes execution entries older than the specified days (default 30).

--id

Filters the executions based on execution id. "limit" and "status" flags are ignored when "id" flag is given.

--help

Provides help for execution.

--limit int32

Limits the number of executions to be listed. Value "0" will list all the executions (default 10).

show

Lists all Networks and its summary information.

--status

Filters the executions based on the status (failed/succeeded/all) (default "all").

efa openstack network show

Displays OpenStack network information.

Syntax

efa openstack network show

Parameters

network

Lists all networks

Examples

```
# efa openstack network show
+--------------------------------------+-----------+------+
|               Neutron ID             | Tenant    | CTAG |
+--------------------------------------+-----------+------+
| 123e4567-e89b-12d3-a456-426655440001 | RegionOne | 900  |
+--------------------------------------+-----------+------+
| 123e4567-e89b-12d3-a456-426655440002 | RegionOne | 950  |
+--------------------------------------+-----------+------+
```

efa openstack network-interface show

Displays OpenStack network-interface information.

Syntax

efa openstack network-interface show

Parameters

network-interface

Lists all network-interfaces

Examples

```
# efa openstack network-interface show

+--------------------------------------+--------------------------------------
+--------------+------------------+
|           Neutron Port ID            |          Neutron Network ID          |  Switch
IP   | Switch Interface |
+--------------------------------------+--------------------------------------
+--------------+------------------+
| 123e4567-e89b-12d3-a456-426655440001 | 123e4567-e89b-12d3-a456-426655440001 |
10.24.80.134 | 0/9              |
+--------------------------------------+--------------------------------------
+--------------+------------------+
| 123e4567-e89b-12d3-a456-426655440003 | 123e4567-e89b-12d3-a456-426655440001 |
10.24.80.133 | 0/9              |
+--------------------------------------+--------------------------------------
+--------------+------------------+
| 123e4567-e89b-12d3-a456-426655440005 | 123e4567-e89b-12d3-a456-426655440002 |
10.24.80.134 | 0/9              |
+--------------------------------------+--------------------------------------
+--------------+------------------+
| 123e4567-e89b-12d3-a456-426655440007 | 123e4567-e89b-12d3-a456-426655440002 |
10.24.80.133 | 0/9              |
+--------------------------------------+--------------------------------------
+--------------+------------------+
```

efa openstack router show

Displays OpenStack router information.

Syntax

efa openstack router show

Parameters

Router

Lists all routers

Examples

```
# efa openstack router show

+--------------------------------------+-----------+
|              Router ID               | Tenant    |
+--------------------------------------+-----------+
| 523e4567-e89b-12d3-a456-426655440001 | RegionOne |
+--------------------------------------+-----------+
```

efa openstack router-interface show

Displays OpenStack router-interface information.

Syntax

`efa openstack router-interface show`

Parameters

`router-interface`

Lists all router-interfaces

Examples

```
# efa openstack router-interface show

+--------------------------------------+--------------------------------------+
|              Subnet ID               |              Router ID               |
+--------------------------------------+--------------------------------------+
| 323e4567-e89b-12d3-a456-426655440001 | 523e4567-e89b-12d3-a456-426655440001 |
+--------------------------------------+--------------------------------------+
| 323e4567-e89b-12d3-a456-426655440002 | 523e4567-e89b-12d3-a456-426655440001 |
+--------------------------------------+--------------------------------------+
```

efa openstack subnet show

Displays OpenStack subnet information.

Syntax

efa openstack subnet show

Parameters

subnet

Lists all subnets

Examples

```
# efa openstack subnet show

+--------------------------------------+--------------------------------------
+---------------+------------+
|              Subnet ID               |              Network ID              |
CIDR      | Gateway IP |
+--------------------------------------+--------------------------------------
+---------------+------------+
| 323e4567-e89b-12d3-a456-426655440001 | 123e4567-e89b-12d3-a456-426655440001 |
20.32.45.0/24 | 20.32.45.1 |
+--------------------------------------+--------------------------------------
+---------------+------------+
| 323e4567-e89b-12d3-a456-426655440002 | 123e4567-e89b-12d3-a456-426655440002 |
10.32.45.0/24 | 10.32.45.1 |
+--------------------------------------+--------------------------------------
+---------------+------------+
```

efa-health show

Verifies connectivity with EFA.

Syntax

`efa-health show`

Examples

This example shows EFA status as UP.

```
$ efa-health show
EFA Host : efa.extremenetworks.com
EFA Status : UP
```

This example shows EFA status as DOWN.

```
$ efa-health show
EFA Host : efa.extremenetworks.com
EFA Status : DOWN
Reason : EFA host is not reachable
```

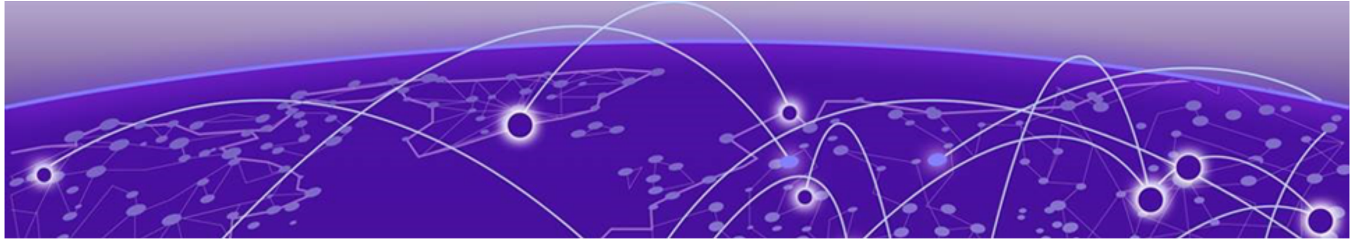
efa-sync execute

Syncs EFA with Neutron.

Syntax

`efa-sync execute`

```
# sudo efa-sync execute
```

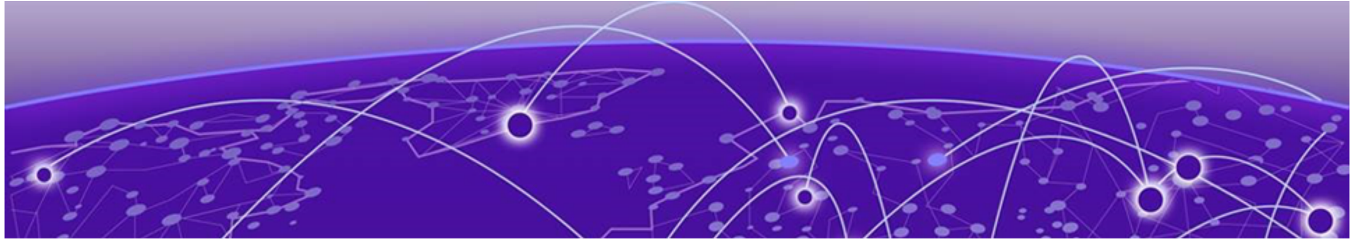


Event Logging

Event Logging

OpenStack services use standard logging levels such as TRACE, DEBUG, INFO, AUDIT, WARNING, ERROR, and CRITICAL. The log messages appear in the logs only if they are more severe than the particular log level. The log files are saved to the respective sub-directory at `/var/log directory`.

OpenTracing is enabled on the Extreme Fabric Automation application to provide end-to-end traceability. The EFA logs are available as part of SupportSave.



Appendix: Neutron REST Endpoints

Neutron REST Endpoints

The following section shows API handled Extreme M12 drivers. For more information on OpenStack APIs, refer to OpenStack Networking API Guide.

EFA Topology Neutron Extension

```
GET
/v2.0/efa_topologies
Shows details of the topology

Normal response codes: 200
```

Table 8: Response parameters

Name	IN	Type	Description
id	Body	UUID	Topology ID
host	Body	String	Host name of the Compute
nic	Body	String	Nic on the Compute
provider_network	Body	String	Provide Network to which the NIC belongs
switch	Body	String	IP address of the switch to which the NIC is connected
Port	Body	String	Switch Interface to which the NCI is connected
po_name	Body	String	LAG as created on EFA for the NICs

```
DELETE

/v2.0/efa_topologies{id}
```

```
Deletes a toology link from its asscociate resources
```

Table 9: Response parameters

Name	IN	Type	Description
id	Path	UUID	Topology ID

```
POST

/v2.0/efa_topologies

Create a link on the efa_topology object
```

Table 10: Response parameters

Name	IN	Type	Description
host	Body	String	Host name of the Compute
nic	Body	String	Nic on the Compute
provider_network	Body	String	Provide Network to which the NIC belongs
switch	Body	String	IP address of the switch to which the NIC is connected
Port	Body	String	Switch Interface to which the NCI is connected
po_name	Body	String	LAG as created on EFA for the NICs

```
e.g body = {‘efa_topology’;:
 { 'host': 'Openstack115',
 'nic':  'eth0'
 'switch': "10.24.35.225',
 'provider_network': "PFPT_LAG',
 'port': '0/1',
 'po_name': 'lag_1', }}
```

Appendix: SR-IOV and Multi-Segment Support

SR-IOV Network

Single Root I/O Virtualization (SR-IOV) allows a single physical NIC to appear as multiple physical NICs. All NICs in the VM must be tagged to the appropriate VLAN.

The two SR-IOV functions are as follows:

- Physical Function (PF) - Physical Ethernet controller that supports SR-IOV.
- Virtual Function (VF) - Virtual device created from a physical Ethernet controller.

The following figure shows a network with:

- ens3f1 (Compute-1) and ens3f0 (Compute-2) as part of network for PF-PT
- ens3f0 (Compute-1) and ens2f0 (Compute-2) as part of network for VF-PT

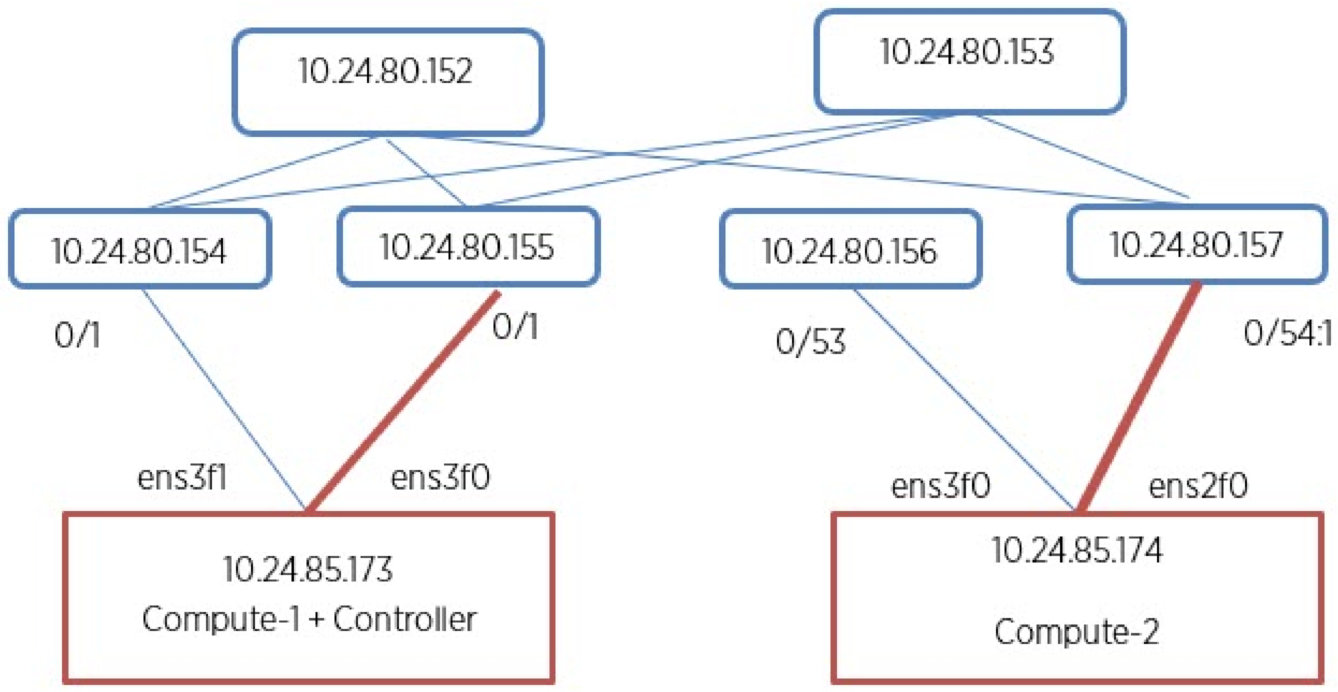


Figure 13: SR-IOV network (VF passthrough)

Create PCI Passthrough Whitelist

Procedure

1. Configure the PCI passthrough whitelist in the `/etc/nova/nova.conf` and `/etc/nova/nova-cpu.conf` files on Compute node 1.

```
[default]
pci_passthrough_whitelist =[{"devname": "ens3f1", "physical_network":
"sriovnet2","device_type": "type-PF"},{ "devname": "ens3f0", "physical_network":
"sriovnet1"}]
```

2. Restart the Nova server.
3. Repeat the procedure for Compute node 2.

Configure SR-IOV Agent

Procedure

1. Configure the SR-IOV NIC Agent in the `/etc/neutron/plugins/ml2/sriov_agent.ini` file on Compute node 1.

```
[securitygroup]
firewall_driver = neutron.agent.firewall.NoopFirewallDriver
[sriov_nic]
physical_device_mappings = sriovnet1:ens3f0,sriovnet2:ens3f1
exclude_devices =
```

2. Run the SR-IOV NIC Agent on Compute node 1.

```
/usr/local/bin/neutron-sriov-nic-agent --config-file /etc/neutron/neutron.conf --
config-file /etc/neutron/plugins/ml2/sriov_agent.ini
```

3. Repeat the procedure for Compute node 2.

Configure Nova Scheduler

Procedure

1. Configure the Nova Scheduler in the `/etc/nova/nova.conf` file on both Controller and Compute nodes.

```
enabled_filters = ...,PciPassthroughFilter
available_filters = nova.scheduler.filters.all_filters
```

2. Restart the Nova Scheduler.

Configure Mechanism Drivers for SR-IOV

Procedure

1. Configure the `sriovnicswitch` mechanism driver in the `ml2_conf.ini` file on each controller.

```
[ml2]
tenant_network_types = vlan
type_drivers = vlan
mechanism_drivers = openvswitch, sriovnicswitch, extreme_efa
[ml2_type_vlan]
network_vlan_ranges = physnet1:2:500
```

2. Ensure that `sriovnet` is configured for the selected network type in the `ml2_conf.ini` file on each controller.

```
[ml2_type_vlan]
network_vlan_ranges = sriovnet1:100:500,sriovnet2:100:500
```

3. Restart the Neutron.

Create Network for VF-PT

Procedure

Create a network for VF-PT.

```
# openstack network create --provider-physical-network sriovnet1 --provider-network-type
vlan --provider-segment 101 \ sriov-net

# openstack subnet create sriov-subnet --network sriov-net --subnet-range 10.65.217.0/24
--gateway 10.65.217.254
```

Create Network for PF-PT

Procedure

Create a network for PF-PT.

```
# openstack network create --provider-physical-network sriovnet2 --provider-network-type
vlan --provider-segment 102  \ pt-net

# openstack subnet create pt-subnet --network pt-net --subnet-range 10.65.217.0/24 --
gateway 10.65.217.254 pt_net_id=$(openstack network show pt-net -c id -f value)
```

All corresponding end-points are provisioned.

Create Virtual Machines

Procedure

1. Create a virtual machine on Compute node 1.

```
# openstack server create --flavor ds512M --image vlan-capable-image --nic port-id=
$port_id  --availability-zone nova:Compute-1  sriov-instance-1
```

2. Repeat the procedure for Compute node 2.

Create SR-IOV Direct Ports

Procedure

1. Create the SR-IOV direct port.

```
net_id=$(openstack network show sriov-net -c id -f value)

# openstack port create --network $net_id --vnic-type direct \ sriov-port port_id=$
(openstack port show sriov-port -c id -f value)
```

2. Repeat the procedure for the second port.

Create SR-IOV Direct-Physical Port

Procedure

1. Create a SR-IOV Direct-Physical port.

```
# openstack port create --network $pt_net_id --vnic-type direct-physical pt-port pt_id=
$ (openstack port show pt-port -c id -f value)
```

2. Repeat the procedure to create the second SR-IOV Direct-Physical port.

Delete SR-IOV Entities

Procedure

Delete the SR-IOV entities.

```
# openstack server delete sriov-instance-1
# openstack server delete sriov-instance-2
# openstack port delete sriov-port2
# openstack port delete sriov-port
# openstack subnet delete sriov-subnet
# openstack network delete sriov-net
```

Multi-Segment Network

A network segment is an isolated Layer 2 segment within a network. A network can contain multiple network segments.

The following figure shows a network with:

- ens3f1 (Compute-1) and ens3f0 (Compute-2) configured as virtio ports in physnet1
- ens3f0 (Compute-1) and ens2f0 (Compute-2) configured as SRIOV ports in sriovnet1

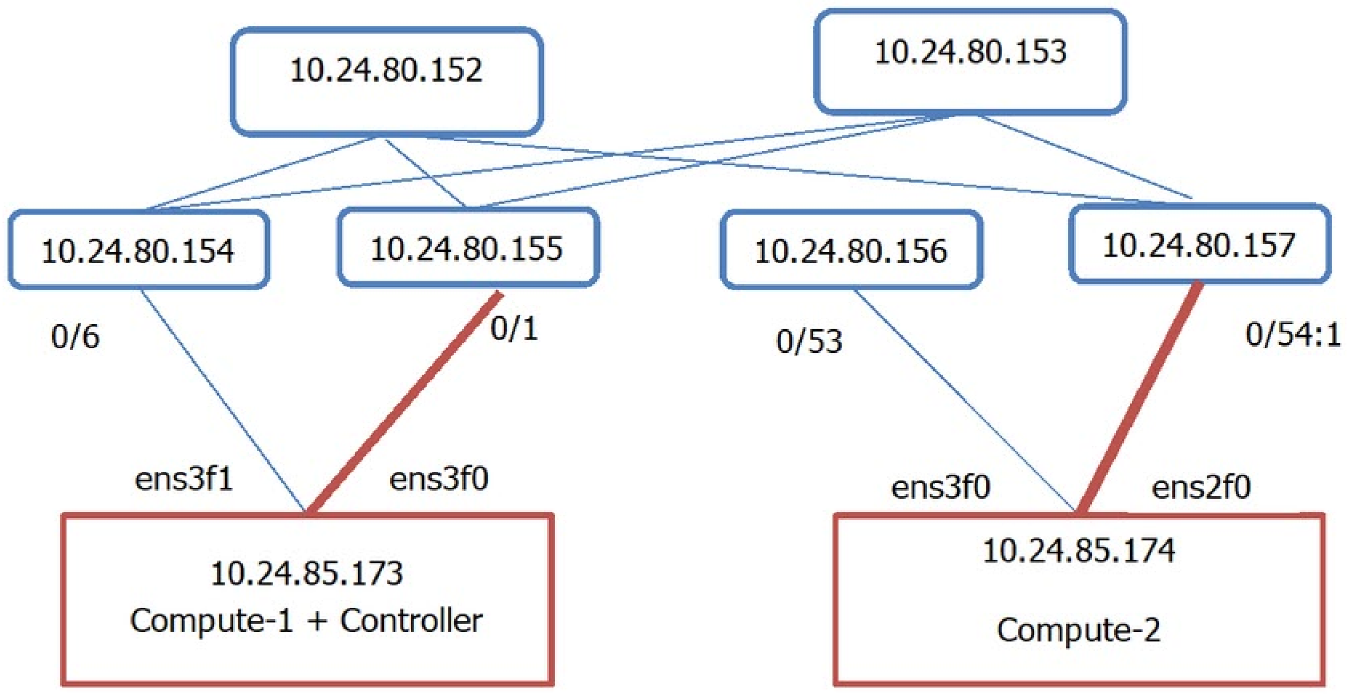


Figure 14: Overview of multi-segment network

Configure Segments in Neutron

Procedure

1. Configure segments in the `/etc/neutron/neutron.conf` file.

   Ensure that the placement IP address points to the Nova server on the Controller node.

   ```
   [DEFAULT]
   # ...
   service_plugins = ..., segments

   [placement]
   auth_url = http://10.24.85.173/identity
   project_domain_name = Default
   project_name = service
   user_domain_name = Default
   password = apassword
   username = nova
   auth_url = http://10.24.85.173/identity_admin
   auth_type = password
   region_name = RegionOne
   ```
2. Restart the Neutron server.

Configure Multi-Segment Network

Procedure

Configure the segment network.

```
# openstack network create --share --provider-physical-network sriovnet2 --provider-
network-type vlan --provider-segment 2016 multisegment
```

Create Multi-Segment Network

Procedure

1. Create a multi-segment network.

   ```
   # openstack network create --provider-network-type vlan --provider-segment 333 msnet1
   ```

2. (Optional) Create a segment for external connectivity through DC Gateway (DC GW).

   ```
   # openstack network segment create --physical-network EXT_A --segment 333 --network-
   type vlan --network msnet1 extseg
   ```

   Regular virtio ports can be created on a separate segment of the same network.

   All corresponding end-points are provisioned for SR-IOV PF-PT and DC GW after segment creation.

Rename Multi-Segment Network

Network must be formed with SR-IOV provider network.

Procedure

Rename the multi-segment network.

```
# segment1=openstack network segment list -c ID -f value openstack network segment set --
name segment1 $virtio_segment
```

Create Subnet on Segment

Procedure

Create a subnet on the segment.

```
# openstack subnet create --network multisegment1 --network-segment dhcp_segment --ip-
version 4 --subnet-range 203.0.113.0/24 \ multisegment-segment1-v4
```

Create a Port on SR-IOV Segment

Procedure

Create a port on SR-IOV segment.

```
# openstack port create --network multisegment --vnic-type direct-physical sriov_port
```

Create a VM Using the Port

Procedure

Create a virtual machine using the port.

```
# openstack server create --flavor ds512M --availability-zone nova:Compute-1 --image
xenial-server-amd64 --nic port-id=sriov_port sriov_vm

# ip link add link ens4 name ens4.2016 type vlan id 2016  sudo dhclient ens4.2016

# sudo dhclient ens4.2016
```