



# Extreme Fabric Automation OpenStack Integration Guide

Version 2.4.0

9036928-00 Rev AA  
February 2021



Copyright © 2021 Extreme Networks, Inc. All rights reserved.

## Legal Notice

Extreme Networks, Inc. reserves the right to make changes in specifications and other information contained in this document and its website without prior notice. The reader should in all cases consult representatives of Extreme Networks to determine whether any such changes have been made.

The hardware, firmware, software or any specifications described or referred to in this document are subject to change without notice.

## Trademarks

Extreme Networks and the Extreme Networks logo are trademarks or registered trademarks of Extreme Networks, Inc. in the United States and/or other countries.

All other names (including any product names) mentioned in this document are the property of their respective owners and may be trademarks or registered trademarks of their respective companies/owners.

For additional information on Extreme Networks trademarks, see: [www.extremenetworks.com/company/legal/trademarks](http://www.extremenetworks.com/company/legal/trademarks)

## Open Source Declarations

Some software files have been licensed under certain open source or third-party licenses. End-user license agreements and open source declarations can be found at: <https://www.extremenetworks.com/support/policies/open-source-declaration/>



# Table of Contents

---

<b>Preface.....</b>	<b>6</b>
Text Conventions.....	6
Documentation and Training.....	7
Getting Help.....	8
Subscribe to Service Notifications.....	8
Providing Feedback.....	8
<b>About this Document.....</b>	<b>10</b>
What's New in this Document.....	10
<b>Introduction to OpenStack Integration .....</b>	<b>12</b>
Extreme Fabric Automation Overview.....	12
EFA Microservices.....	14
Fabric Service.....	14
Tenant Service.....	15
Inventory Service.....	15
Notification Service.....	15
RASlog Service.....	15
Security Service.....	15
SNMP Service.....	15
Ecosystem Integration Services.....	16
OpenStack Integration Overview.....	17
OpenStack Core Components.....	17
OpenStack Network Nodes.....	18
OpenStack Ecosystem Integration Management.....	19
OpenStack Service Commands.....	20
Limitations.....	20
Dependencies.....	21
<b>OpenStack Plugin Installation.....</b>	<b>22</b>
EFA OpenStack Plugin Package.....	22
System Requirements.....	22
Install EFA OpenStack Neutron Plugin.....	22
Upgrade or Downgrade OpenStack Neutron Plug-in.....	24
Uninstall EFA OpenStack Neutron Plug-in.....	24
<b>OpenStack Configuration.....</b>	<b>25</b>
Setup Overview.....	25
Topology Setup in Neutron.....	26
Tenant Creation.....	27
Configure Neutron to Connect to EFA.....	28
Enabling Mechanism Driver and Plugins.....	29
Enable Journaling.....	29

Neutron Link Mapping.....	31
ML2 Mechanism Driver.....	32
VLAN Provisioning Based on Provider Networks.....	34
Create a Single Segment Virtio VM Network.....	34
Create a Single Segment VFPT VM Network.....	37
Create a Single Segment PFPT VM Network.....	39
Create a Multi Segment Virtio VM, PFPT, and VFPT Network.....	42
L3 Service Plug-in Architecture.....	44
OpenStack Network Workflow Using L3 Service Plug-in.....	45
Centralized Routing.....	47
L3 Flavors.....	48
L3 IPV6 MTU.....	49
L3 IPV6 ND/RA.....	49
<b>Journaling.....</b>	<b>51</b>
Journaling Console CLI Commands .....	51
OpenStack Utilities.....	51
Sync EFA with Neutron.....	52
Verify EFA Health.....	53
Multiple VIM/VPOD Instances.....	54
Scale-in and Scale-out of Compute Nodes.....	57
Virtual Machine Migration.....	58
Enable VMotion.....	58
Migrate Virtual Machines.....	58
Add Certificate to OpenStack Controller.....	60
<b>EFA OpenStack Service Command Reference.....</b>	<b>61</b>
efa openstack debug.....	62
efa openstack execution.....	64
efa openstack network show.....	65
efa openstack network-interface show.....	66
efa openstack router show.....	67
efa openstack router-interface show.....	68
efa openstack subnet show.....	69
efa openstack sync start.....	70
efa-health show.....	71
efa-journal.....	72
efa-sync execute.....	74
openstack network efa-topology-link-map create.....	76
openstack network efa-topology-link-map delete.....	77
openstack network efa-topology-link-map list.....	78
<b>Appendix: Neutron REST Endpoints.....</b>	<b>79</b>
Neutron REST Endpoints.....	79
EFA Topology Neutron Extension.....	79
<b>Appendix: SR-IOV and Multi-Segment Support.....</b>	<b>81</b>
SR-IOV Network.....	81
Create PCI Passthrough Allowed List.....	82
Configure SR-IOV Agent.....	82
Configure Nova Scheduler.....	83
Configure Mechanism Drivers for SR-IOV.....	83

Create Network for VF-PT.....	83
Create Network for PF-PT.....	83
Create Virtual Machines.....	84
Create SR-IOV Direct Ports.....	84
Create SR-IOV Direct-Physical Port.....	84
Delete SR-IOV Entities.....	84
Multi-Segment Network.....	84
Configure Segments in Neutron.....	85
Configure Multi-Segment Network.....	85
Create Multi-Segment Network.....	86
Rename Multi-Segment Network.....	86
Create Subnet on Segment.....	86
Create a Port on SR-IOV Segment.....	86
Create a VM Using the Port.....	86



# 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 <i>enter</i> and <i>type</i>	When you see the word <i>enter</i> 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 <i>type</i> .
<b>Key</b> names	Key names are written in boldface, for example <b>Ctrl</b> or <b>Esc</b> . If you must press two or more keys simultaneously, the key names are linked with a plus sign (+). Example: Press <b>Ctrl+Alt+Del</b>
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.
<b>NEW!</b>	New information. In a PDF, this is searchable text.

**Table 3: Command syntax**

Convention	Description
<b>bold</b> text	Bold text indicates command names, keywords, and command options.
<i>italic</i> text	Italic text indicates variable content.
[ ]	Syntax components displayed within square brackets are optional. Default responses to system prompts are enclosed in square brackets.
{ <b>x</b>   <b>y</b>   <b>z</b> }	A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options.
<b>x</b>   <b>y</b>	A vertical bar separates mutually exclusive elements.
< >	Nonprinting characters, such as passwords, are enclosed in angle brackets.
...	Repeat the previous element, for example, <i>member</i> [ <i>member</i> ...].
\	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/](http://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](http://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](http://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](mailto: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](#) on page 10

## What's New in this Document

---

The following table describes information added to this guide for the Extreme Fabric Automation 2.4.0 software release.

**Table 4: Summary of changes**

Feature	Description	Described in
Journaling	Journaling for L2 and L3	<a href="#">Journaling</a> on page 51
ND/RA Support	<ul style="list-style-type: none"><li>• IPV6 ND MTU support</li><li>• IPV6 No-AutoConfig support</li></ul>	<a href="#">L3 IPV6 ND/RA</a> on page 49
Centralized Routing	Routers can be configured either in centralized mode or distributed mode.	<a href="#">Centralized Routing</a> on page 47
L3 Service Plugin	The Extreme L3 Service Plugin proxies the Neutron API calls for network management toward the Openstack Service.	<a href="#">L3 Service Plug-in Architecture</a> on page 44
L3 Flavors Support	L3 flavors allow multiple backends in a single Neutron deployment.	<a href="#">L3 Flavors</a> on page 48
Enhancements to EFA Health tool	This facility is used to monitor the health of EFA.	<a href="#">Verify EFA Health</a> on page 53

**Table 4: Summary of changes (continued)**

Feature	Description	Described in
Enhancements to the EFA Sync tool	When EFA goes out of sync with Neutron, this tool can be executed to resync all entries.	<a href="#">Sync EFA with Neutron</a> on page 52
New and modified OpenStack commands	Syntax was changed for the following: <ul style="list-style-type: none"> <li>• efa-sync</li> <li>• efa-journal</li> <li>• efa-health</li> <li>• openstack network efa-topology-link-map</li> </ul> Output was changed for the following: <ul style="list-style-type: none"> <li>• efa openstack network show</li> <li>• efa openstack subnet show</li> <li>• efa openstack router show</li> </ul>	<a href="#">EFA OpenStack Service Command Reference</a> on page 61

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



# Introduction to OpenStack Integration

---

[Extreme Fabric Automation Overview](#) on page 12

[EFA Microservices](#) on page 14

[OpenStack Integration Overview](#) on page 17

[OpenStack Ecosystem Integration Management](#) on page 19

[Limitations](#) on page 20

## 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 14.

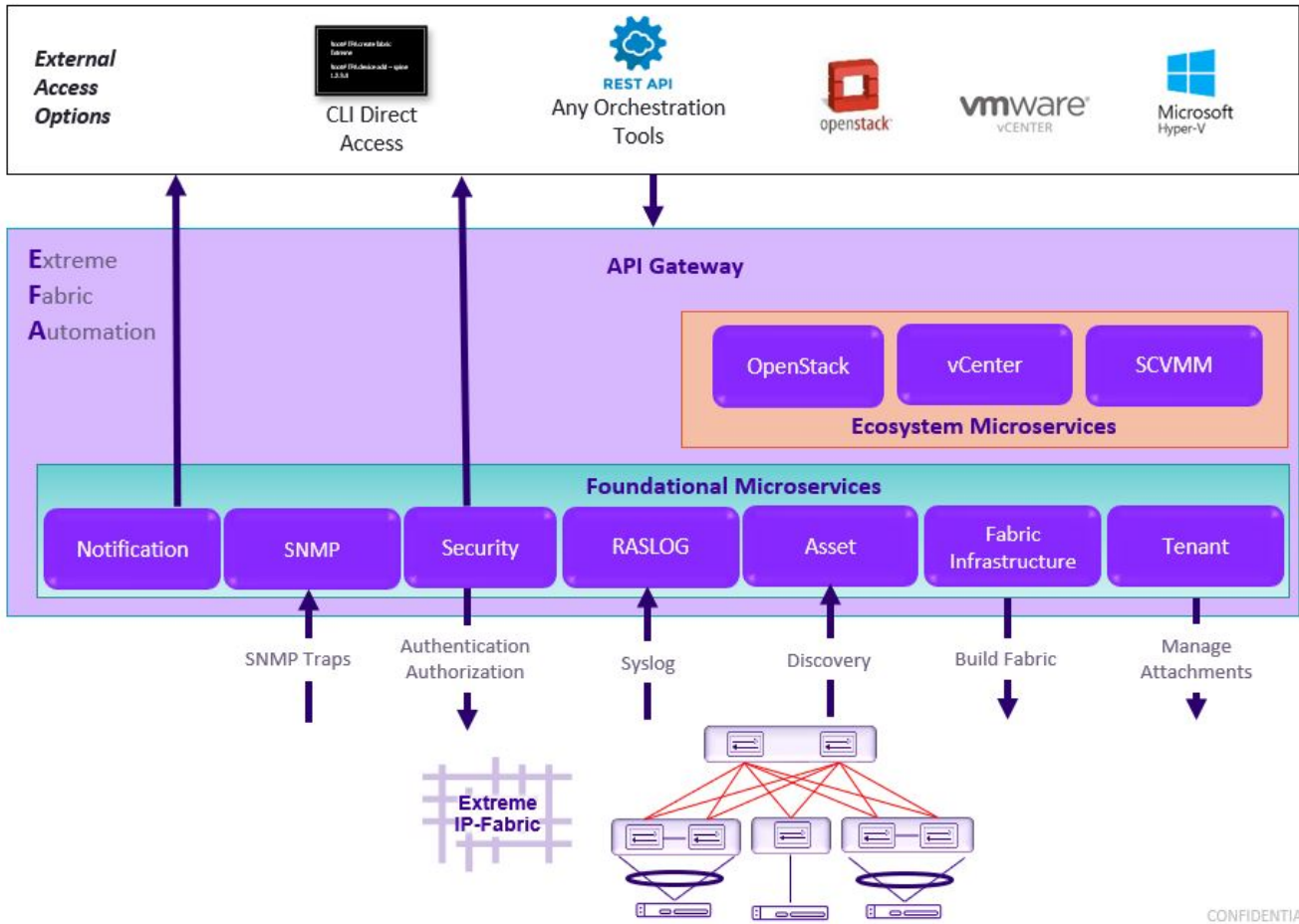
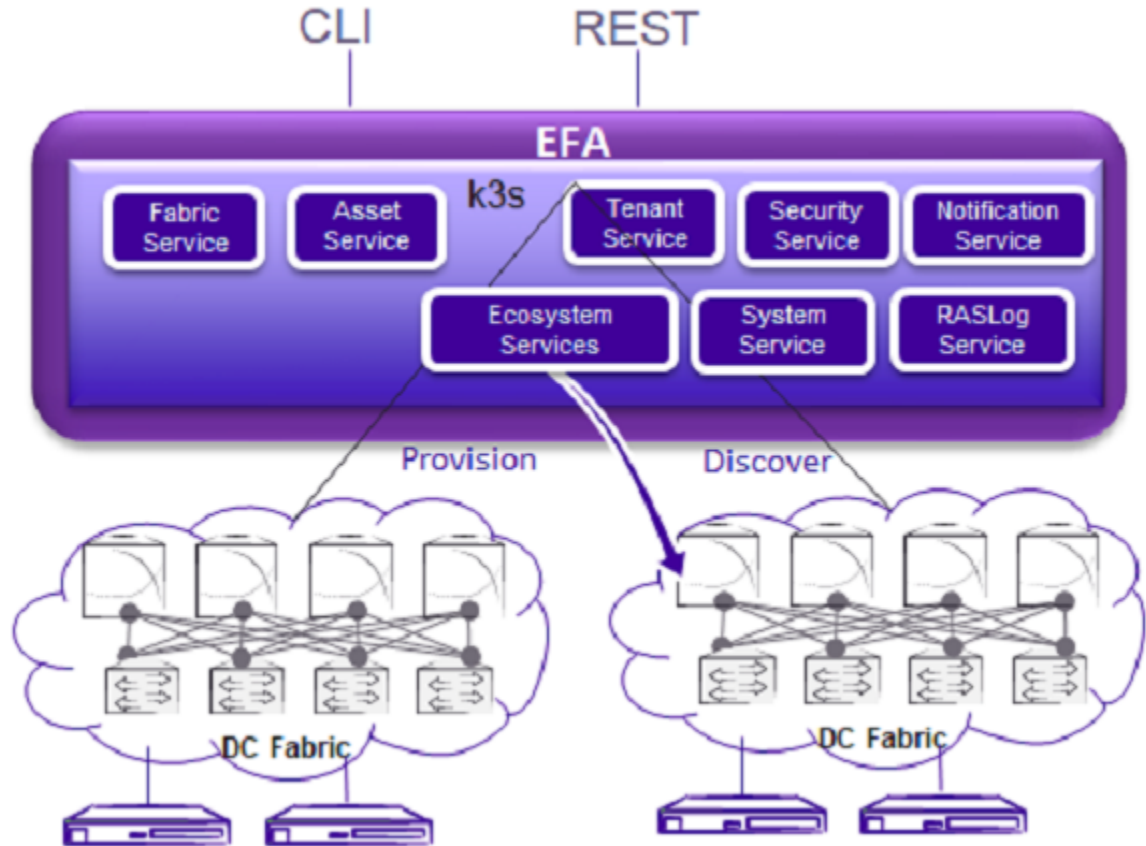


Figure 1: EFA orchestration

## 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 data centers (non-Clos)
- Support for 3-stage and 5-stage Clos fabrics
- Support for MCT configuration

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 endpoints, 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 hand-off, and Layer 3 hand-off 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 and forwards notifications to subscribers. For more information, see [#unique\\_20](#).

## 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.

## SNMP Service

The SNMP Service processes SNMP traps from devices and forwards notifications to subscribers. For more information, see [#unique\\_21](#).

## 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, 2.4.0](#). 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, 2.4.0](#). 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, 2.4.0](#). 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
- vMotion (virtual machine migration)
- ML2 driver with support for:
  - Network and segment provisioning for non-default physnets
  - DC-owner-based I2 extension for DC gateway
- Topology changes for port-based extension of DC gateway addition and deletion of topology entries and its changes on EFA EPGs
- Single-homed connections to the edge port
- Multi-segment support
- Journaling support for L2 and L3



- L3 service plugin:
  - Routing feature support using VRF
  - Flavor (service provider) support
  - Centralized routing
  - IPv6 support (dual stack)
- Layer 3 flavors
- Neighbor Discovery and Router Advertisement support:
  - IPv6 ND MTU support
  - IPv6 No-Autoconfig support

## OpenStack Integration Overview

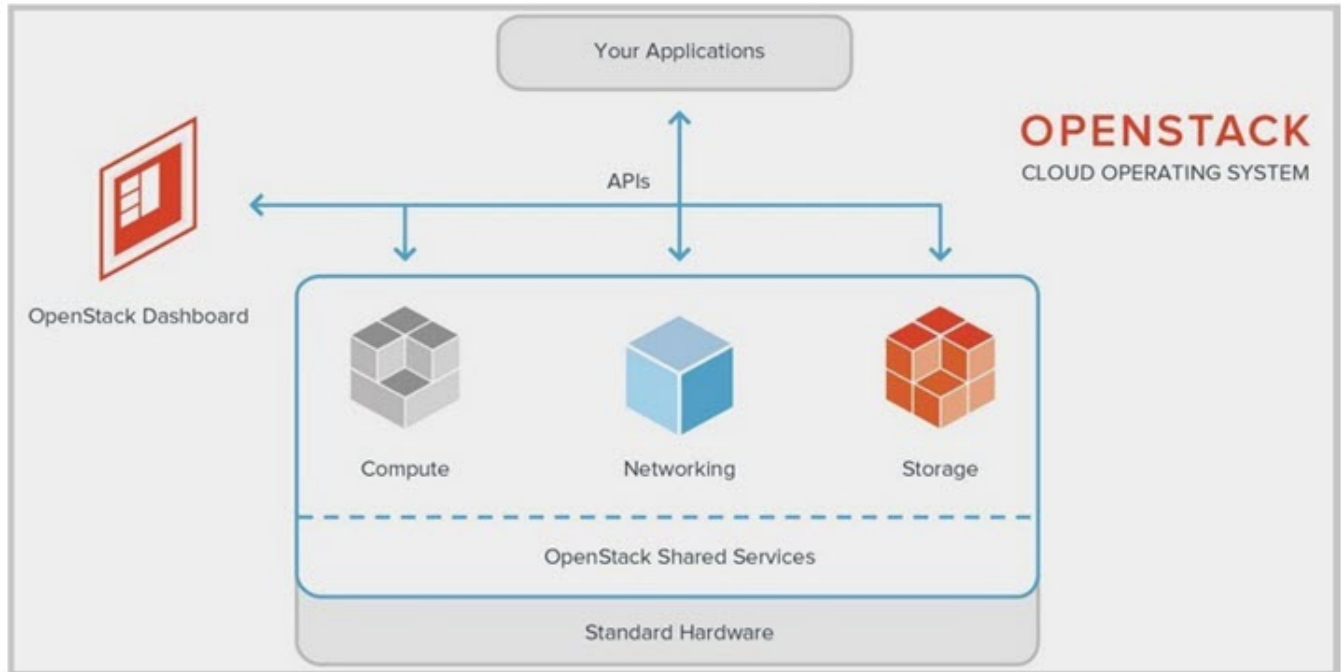
OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a Data Center. OpenStack Integration enables System administrators to manage and provision resources through APIs and web interface.

### 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**



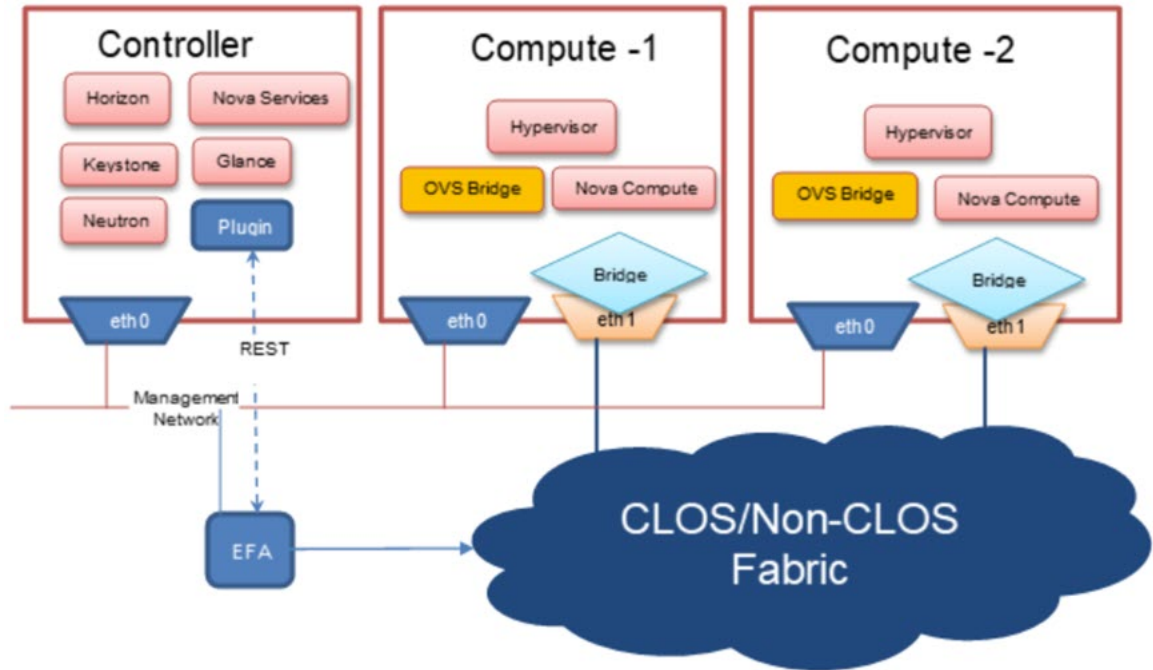
## OpenStack Network Nodes

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.



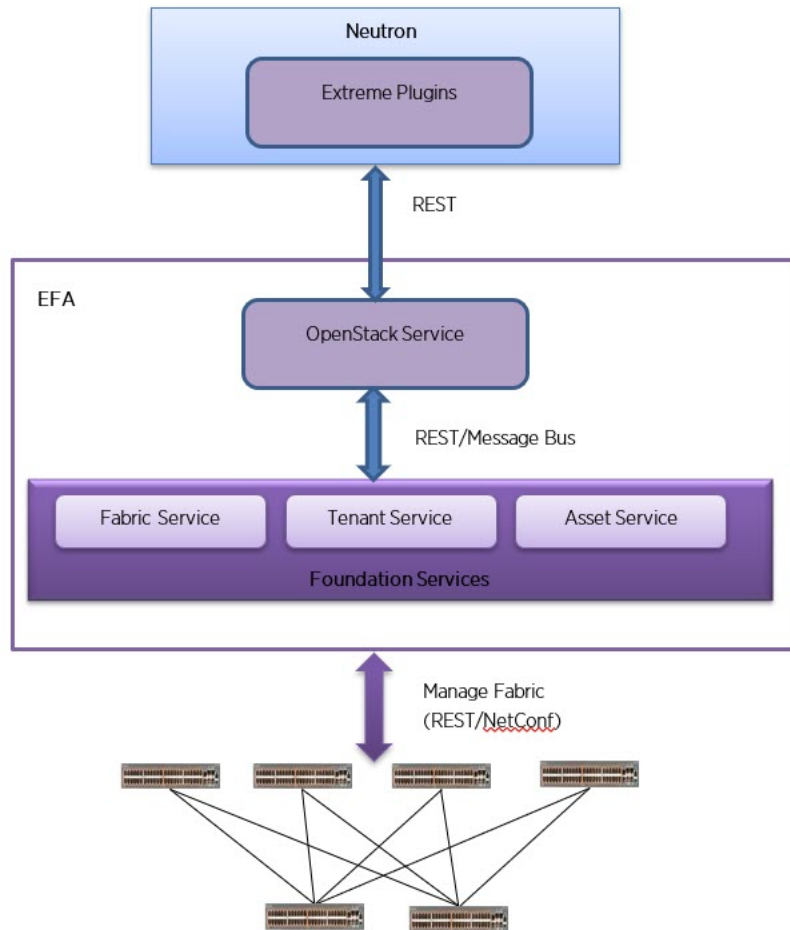
**Figure 4: Overview of EFA OpenStack Network**

## OpenStack Ecosystem Integration Management

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

Foundation Service	Description
Fabric Service	This service provides mechanisms to create: <ul style="list-style-type: none"> <li>• Non-Clos/Clos fabric</li> <li>• Clos fabrics can be 3-stage or 5-stage</li> </ul>
Tenant Service	This service provides mechanisms to create: <ul style="list-style-type: none"> <li>• Layer 2 networks</li> <li>• Layer 3 networks</li> </ul>
Asset or Inventory Service	This service provides mechanisms to manage: <ul style="list-style-type: none"> <li>• Physical and logical assets of the switches in the fabric</li> <li>• The credentials of the switches</li> </ul>

The following figure shows an overview of the 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 61.

For EFA commands and supported parameters, see [Extreme Fabric Automation Command Reference, 2.4.0](#).

## Limitations

OpenStack ecosystem integration limitations are as follows:

- Only green field deployment is supported.
- There is an operation in Neutron to disable routers. This operation is not supported by Extreme OpenStack support (L3 plugin or flavor), because SLX devices do not support the disabling of routers.

## Dependencies

OpenStack ecosystem integration dependencies are as follows:

- Non-Clos (2-node, 4-node, and 8-node)
- Clos
  - 3-stage Clos (leaf or spine nodes)
  - 5-stage Clos (leaf, spine, or super-spine nodes)
- Compute nodes (connected to every leaf node in MCT and non-MCT configurations)



# OpenStack Plugin Installation

---

[EFA OpenStack Plugin Package](#) on page 22

[Install EFA OpenStack Neutron Plugin](#) on page 22

[Upgrade or Downgrade OpenStack Neutron Plug-in](#) on page 24

[Uninstall EFA OpenStack Neutron Plug-in](#) on page 24

## EFA OpenStack Plugin Package

---

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

The EFA plugin for OpenStack is packaged as RPM files.

The EFA OpenStack plugin package supports many features. For a detailed list, see [OpenStack Ecosystem Integration Management](#) on page 19.

## System Requirements

System requirements for EFA are provided in the [Extreme Fabric Automation Administration Guide, 2.4.0](#).

## Install EFA OpenStack Neutron Plugin

---

This section provides information required to install Extreme Fabric Automation OpenStack Neutron plugin on Ubuntu.

### Before You Begin

The prerequisites for installing the EFA OpenStack Neutron plugin are as follows:

- Working knowledge of 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 plugin supports only Extreme specific OpenStack services. If other OpenStack services are required, install the respective plugins prior to EFA OpenStack Neutron plugin 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 `m12_conf_extreme.ini` file.

## Procedure

1. Install the EFA OpenStack plugin RPM package.

```
# $sudo rpm -U <RPM file>
```

2. Note the Neutron configuration file layout.

- Neutron configuration: `/etc/neutron/neutron.conf`
- ML2 plugin 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 M12 `core_plugin` in the `neutron.conf` file.

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

Do not enable the reference router plugin.

4. Enable Neutron in the `m12_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
```

5. Enable the Neutron EFA extension plugin in the `m12_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
```

6. Enable Extreme EFA mechanism drivers in `m12_conf.ini`.

```
m12_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)
```

7. Modify the system unit file to start Neutron with `m12_conf_extreme.ini`.

```
# ExecStart = /usr/local/bin/neutron-server --config-file /etc/neutron/neutron.conf --
config-file /etc/neutron/plugins/m12/m12_conf.ini --config-file /etc/neutron/
plugins/m12/m12_conf_extreme.ini
```

```
# systemctl daemon-reload
```

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

8. Restart the Neutron server.

```
systemctl restart devstack@q-agt.service
```

along with

```
devstack@q-svc.service
```

If you are installing using Open Source, use the `sudo service neutron-* restart` command.

If you are installing using DevStack, use the `sudo systemctl restart devstack@q-svc.service` command.

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

- The Neutron service was started with the `m12_conf_extreme.ini` file.
- The **efa-topology** extension is loaded using `openstack extension show efa-topology`.

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

If you are installing using Open Source stack, use the `sudo service neutron-* status` command.

## Upgrade or Downgrade OpenStack Neutron Plug-in

### Procedure

Upgrade or downgrade the EFA plug-in RPM package from the network node.

```
# $sudo rpm -U <RPM file>
```

## Uninstall EFA OpenStack Neutron Plug-in

### Procedure

Uninstall the EFA OpenStack Neutron plug-in.

```
# sudo rpm -e networking-extreme
```





# OpenStack Configuration

---

[Setup Overview](#) on page 25

[ML2 Mechanism Driver](#) on page 32

[L3 Service Plug-in Architecture](#) on page 44

[Journaling](#) on page 51

[OpenStack Utilities](#) on page 51

[Multiple VIM/VPOD Instances](#) on page 54

[Scale-in and Scale-out of Compute Nodes](#) on page 57

[Virtual Machine Migration](#) on page 58

[Add Certificate to OpenStack Controller](#) on page 60

## Setup Overview

---

The following summarizes the phases in configuring OpenStack to work with EFA. Details are provided in the sections that follow.

The instructions assume that the IP Fabric has already been set up using Fabric Service commands. For details on creating a Clos or non-Clos IP Fabric, refer to the EFA Administration Guide.

**Table 5: Phases in Configuring OpenStack for EFA**

Steps	Explanation
Create fabric	The instructions in the following sections assume that the IP Fabric has already been set up.
Set up physical topology	Shows SLX switch topology and its interconnection with compute nodes
Create tenant	Setting up the EFA tenants according to the physical topology
Configure Neutron	Configuration of Neutron to connect to EFA
Enable mechanism driver and plugins	Enabling Layer2, Layer3 plugins and drivers on Neutron
Enable journaling	Alternate Neutron configuration for setting up Layer2 and Layer3 using journaling
Map Neutron links	Link mapping according to the physical topology on Neutron

## Topology Setup in Neutron

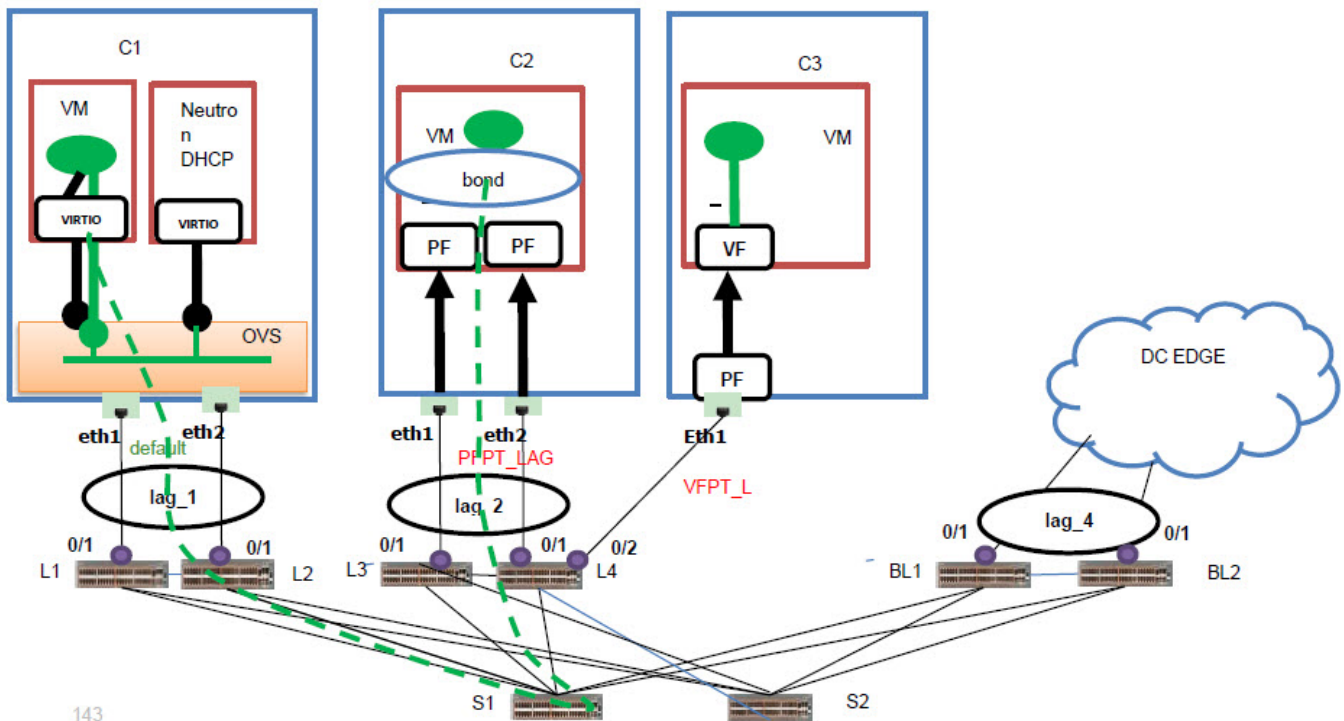
Use OpenStack CLI to manage the topology between compute nodes and SLX devices. The Physical Network Topology extension provides a Neutron CLI for the OpenStack administrator to manage links between SLX devices and Compute NICs. Neutron has an abstract for physical network called Provider Network.

The Extreme Topology plugin provides **efa-link-mapping** command to configure and set up the topology. Use the efa-link-mapping feature to scale in to and scale 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 57.



**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 `m12_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.



**Figure 6: Topology setup in Neutron**

Use the settings in the follow table to set up the physical topology.

Node	IP Address or Description
C1	10.24.51.115 (OpenStack115)
C2	10.24.51.116 (OpenStack116)
C3	10.24.51.117 (OpenStack117)

Node	IP Address or Description
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

## Tenant Creation

One OpenStack instance is mapped to an EFA tenant. This section describes the steps involved in setting up the tenant on EFA.

**Table 6: Steps for setting up tenant on EFA**

Steps	Description
Create a Tenant	Create an EFA tenant. The name of the tenant should be the same as the <b>region_name</b> in Neutron.
Create Lags/Port Channel	Create lag according to the physical topology. The names of the lags must be used in defining the topology mapping on Neutron.
Create Shared Tenant	Create an EFA tenant that contains the shared resources across multiple EFA tenants. The name of the tenant must be used as <b>region_shared</b> in ML2 configurations.
EFA REST TOKEN	Used as API key <b>efa_rest_token</b> in ML2 configurations for secure connection towards EFA.

The following are the EFA CLI commands used to achieve the steps above:

**Table 7: EFA CLI commands for setting up tenant on EFA**

EFA CLI Command	
<code>efa tenant create --name <b>VIM_1</b> --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]</code>	Create Tenant VIM_1
<code>efa tenant po create --name lag_1 --tenant <b>VIM_1</b> --port 10.24.14.133.[0/1], 10.24.14.134.[0/1] --speed 10Gbps --negotiation active</code>	Lag on C1
<code>efa tenant po create --name lag_2 --tenant <b>VIM_1</b> --port 10.24.14.135.[0/1], 10.24.14.136.[0/1] --speed 10Gbps --negotiation active</code>	Lag on C2

**Table 7: EFA CLI commands for setting up tenant on EFA (continued)**

EFA CLI Command	
<code>efa tenant create --name <b>SHARED_TENANT</b> --port 10.24.14.191.[0/1], 10.24.14.192.[0/1] -type <b>shared</b></code>	Shared tenant on border leaf
<code>efa tenant po create --name lag_4 --tenant <b>SHARED_TENANT</b> --port 10.24.14.133.[0/1], 10.24.14.134.[0/1] --speed 10Gbps --negotiation active</code>	Border leaf lag created on shared tenant
<code>efa auth client register --name <b>VIM_1</b> --type openstack</code>	Register <b>VIM_1</b> tenant for OpenStack access
<code>efa auth apikey generate --client-id &lt;client_id&gt;</code>	Use the Client ID generated in client register for API key generation Use the generated API Key as a the <b>efa_rest_token</b> in OpenStack

## Configure Neutron to Connect to EFA

The entities in the following script need to be configured for the OpenStack instance to connect to EFA.

### Procedure

Edit `/etc/neutron/plugins/ml2/ml2_conf_extreme.ini` on VIM-1:

```
[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
use_fqdn = false
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
```

The EFA tenant name and the `region_name` in OpenStack should be the same.

### Token Description

Token	Description
<code>efa_rest_token</code>	Authentication token used for EFA
<code>efa_cert_file</code>	The location of the CA Cert Chain for EFA
<code>efa_secure_mode</code>	True (for secure connections)
<code>efa_port</code>	Port running the HTTP service
<code>efa_host</code>	IP address of EFA service

Token	Description
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

## Enabling Mechanism Driver and Plugins

### About This Task

Make configurations on the Neutron controller for starting the Extreme EFA Mechanism Driver and L3 Service Plugin.

### Procedure

1. **ML2 Configuration:** Set the mechanism driver to `extreme_efa`, along with `openvswitch`. In the `/etc/neutron/plugins/ml2/ml2_conf.ini` file, add the following lines:

```
[ml2]
mechanism_drivers = openvswitch, extreme_efa, sriovnicswitch
```

2. **Topology Plugin Configuration:** Activate the topology plugin by adding the following lines in the `/etc/neutron/neutron.conf` file:

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

3. **L3 Service Plugin Configuration:** Activate the L3 Service plugin by adding the following lines in the `/etc/neutron/neutron.conf` file:

```
[DEFAULT]
service_plugins = extreme_l3_efa, efa_topology_plugin, trunk, segments
```



#### Note

`extreme_l3_efa` is an L3 Service Plugin and its replacement for the default L3 Service Plugin (called `router`). Only one of them should be used.

4. **L3 Flavor Configuration:** Enable L3 Flavors with Extreme by adding the relevant service providers to `neutron.conf`:

```
[service_providers]
service_provider =
L3_ROUTER_NAT:extreme:networking_extreme.l3.l3_flavor.ExtremeL3ServiceProvider:default
```

Flavors require the default L3 Service Plugin (called `router`). This means flavors can be used only when the service plugin is `router`.

```
service_plugins = router, efa_topology_plugin, trunk, segments
```

## Enable Journaling

This is an alternate Neutron Configuration for setting up Layer2 and Layer3 using Journaling.

This is an alternate Neutron Configuration for setting up Layer2 and Layer3 using Journaling. Complete the following changes in the configuration files to enable the Journaling feature.

### ML2 Configuration

The following configuration lines must be available in `/etc/neutron/plugins/ml2/ml2_conf.ini`. The mechanism driver must be set to `extreme_efa_v2` along with `openvswitch`.

```
[ml2]
mechanism_drivers = openvswitch, extreme_efa_v2,sriovnicswitch
```

### L3 Service Plugin Configuration

The following configuration lines must be available in the `/etc/neutron/neutron.conf` file for the L3 Service plugin to be activated.

```
[DEFAULT]
service_plugins = extreme_l3_efa_v2,efa_topology_plugin,trunk,segments
```



#### Note

`extreme_l3_efa_v2` is an L3 Service Plugin and is a replacement for the default L3 Service Plugin (called `router`). Use either one; not both.

### L3 Flavor Configuration

To enable L3 Flavors with Extreme, service providers must be added to `neutron.conf`, as follows. Multiple service providers flavor, that is `ExtremeL3ServiceProvider` and `ExtremeL3ServiceProviderv2` can be added to the configuration file, but only one can be the default. Router creation CLI must specify the flavor profile for whichever router is being created. For more information, see [Commands for L3 Flavor Creation](#).

```
[service_providers]
service_provider =
L3_ROUTER_NAT:extreme:networking_extreme.l3.l3_flavor_v2.ExtremeL3ServiceProvider2:default
```



#### Note

Flavors require the default L3 Service Plugin (called `router`). This means that flavors can be used only when the service plugin is `router`:

```
service_plugins =
    router,efa_topology_plugin,trunk,segments
```

### Additional Journaling Specific Configuration

These configurations effect the working of the Journaling Mechanism which are configured in `/etc/neutron/plugins/ml2/ml2_conf.ini`.

Parameter	Description	Default
<code>sync_timeout</code>	Sync thread timeout in seconds or fraction.	10
<code>retry_count</code>	Number of times to retry a row before failing.	5

Parameter	Description	Default
maintenance_interval	Journal maintenance operations interval in seconds.	300
completed_rows_retention	Time to keep completed rows (in seconds). For performance reasons it's not recommended to " change this from the default value (0) which "indicates completed rows aren't kept. This value is checked every maintenance_interval by the cleanup thread. To keep completed rows indefinitely, set the value to -1)),	0

## Neutron Link Mapping

You can configure the topology entries on Neutron according to the physical topology.

The following console script can be used for configuring the topology entries on Neutron according to the physical topology. These commands provide mapping of host, NIC, provider-network to switch, port, lag. If the lag or po is not created on EFA, then the `--po-name` should be left empty.

```
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 --switch 10.24.14.191 --
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
```

The same information can be provided as part of the startup file which is specified in `ml2_conf_extreme.ini`.

```
opensuse@Openstack114:~$ cat /home/opensuse/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
```

**Note**

The number of comma delimiters used in each line within the CSV file must be the same. This information can be provided through a `-file` option in the command.

```
efa-link-mapping add --file /home/opensuse/link.csv
```

Fields	Description
Host	Host name of the compute node (as seen in OpenStack server list). In case of a DC Gateway (border leaf), this will be a label like DC-GW1.
nic	NIC on the compute node. In case of a DC Gateway (border leaf), this is empty.
pn	Neutron physnet that is being mapped
switch	IP address of the physical switch
Port	Switch interface that is connected to the NIC. In case of a DC Gateway (border leaf), this will be the switch interface that is connected to an external DC Edge.
po-name	Port channel name that was already created using the tenant command.

Only VLAN-based Physnet entries should be added to this table.

**Note**

The same behavior can be achieved using the `openstack network efa-topology-link-map` command.

*Topology Mapping CLI Commands*

The following commands are available for showing and manipulating topology mapping:

- `openstack network efa-topology-link-map list`
- `openstack network efa-topology-link-map create`
- `openstack network efa-topology-link-map delete`

For more information, see [openstack network efa-topology-link-map list](#) on page 78 and the related commands.

## ML2 Mechanism Driver

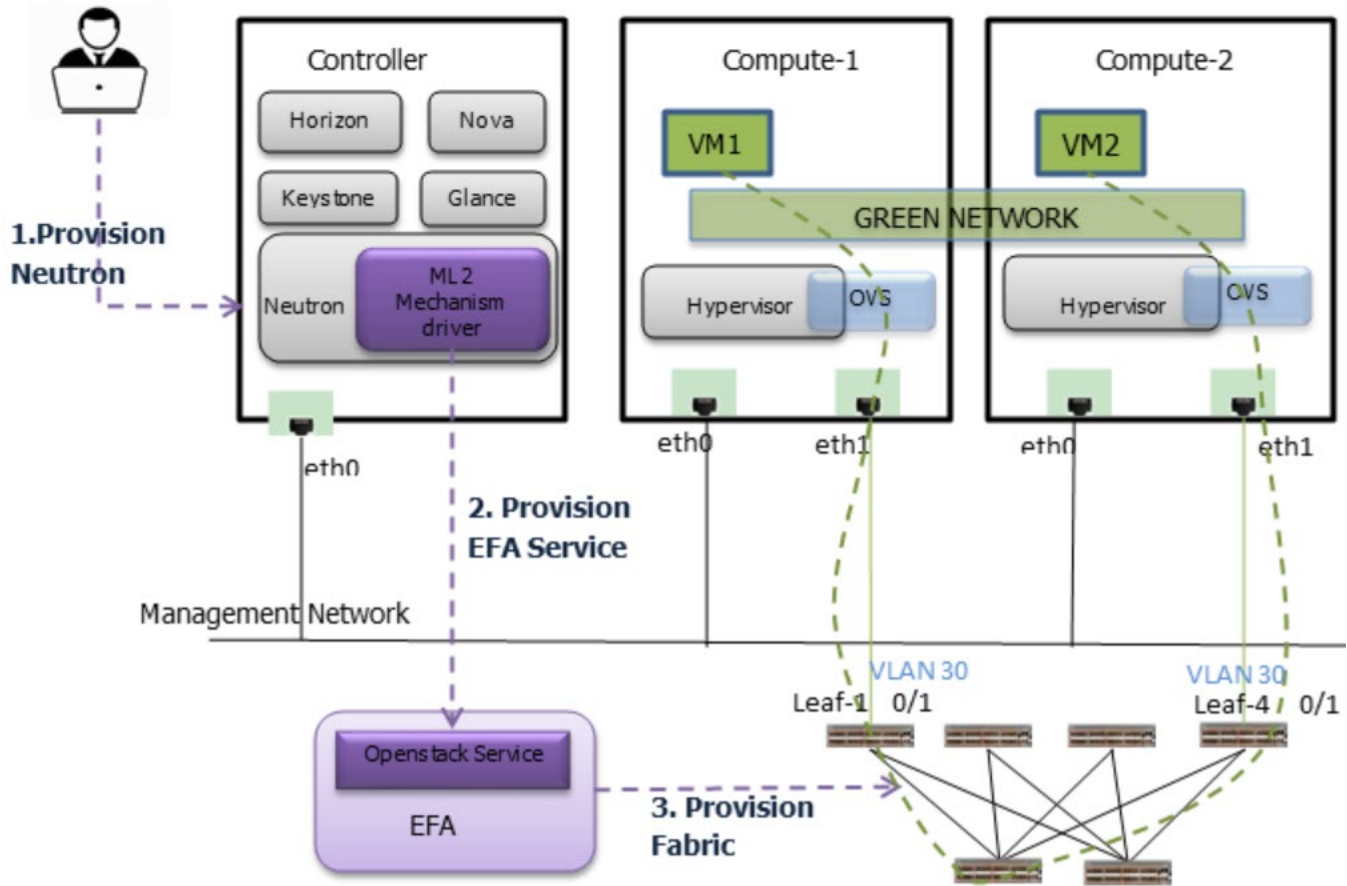
---

The Modular Layer 2 (ml2) plugin is a framework allowing OpenStack networking to simultaneously utilize the variety of layer 2 networking technologies found in complex real-world data centers.



External networks are managed using ml2 Mechanism Driver. 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.

The following figure shows an overview of the ML2 mechanism driver within Neutron.



**Figure 7: Overview of ML2 Mechanism Driver within Neutron**

Extreme ML2 Mechanism driver within the Neutron component of OpenStack proxies the Neutron API calls for network management toward the OpenStack Service running in EFA.

The OpenStack Service in EFA translates the neutron network management calls to appropriate tenant API calls and provision the fabric with appropriate L2 networking constructs.

## VLAN Provisioning Based on Provider Networks

There are several scenarios in which Virtual Machine can be created on provider networks.

**Table 8: VLAN provisioning based on provider networks (physnet)**

Provider Network (Physnet)	VLAN Provisioning by ML2
default	VLAN provisioning of endpoints is done when Neutron ports are bound to a host or compute node. Example: <ul style="list-style-type: none"> <li>• The Neutron port is bound to a host when VM is launched.</li> <li>• The Neutron port is bound to a host (Neutron controller) when DHCP starts.</li> </ul>
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 deprovisioned 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

## Create a Single Segment Virtio VM Network

You can create a single segment Virtio VM network on a default physnet and extend the network to DC Edge.

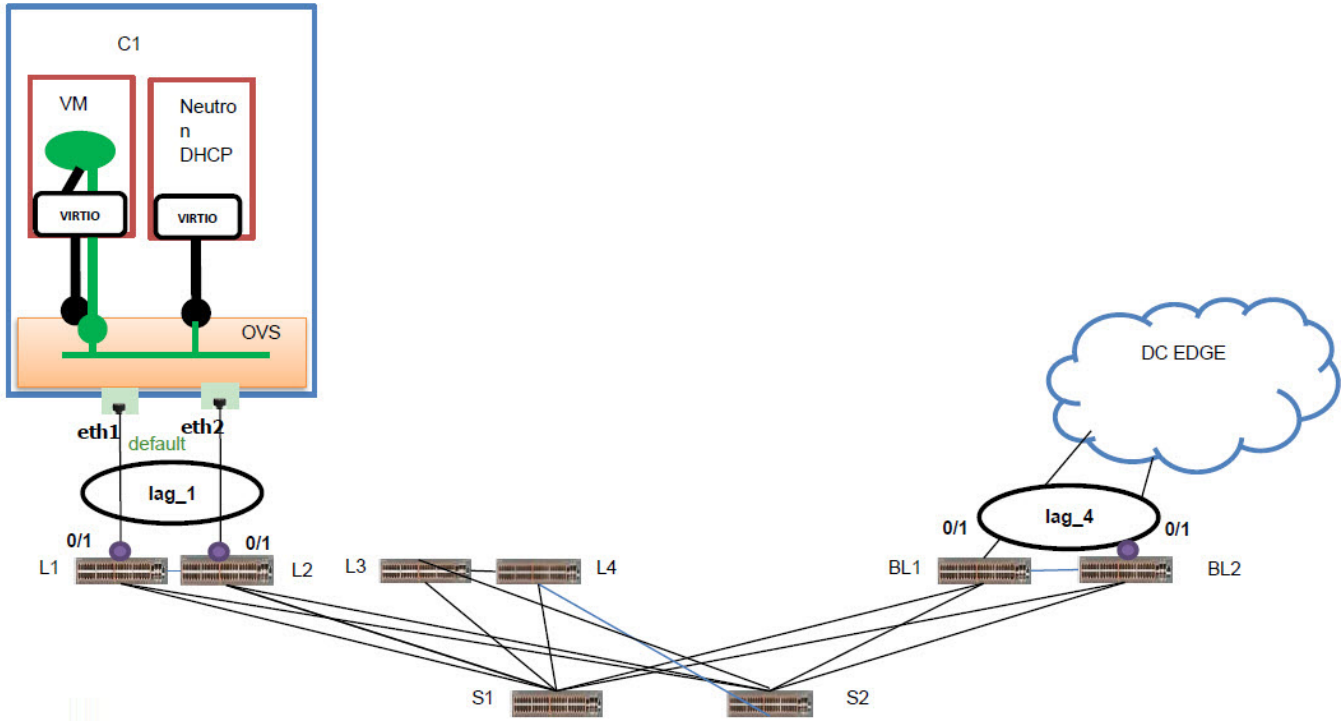


### Note

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 8: Overview of single segment Virtio VM network**

**Table 9: Commands and impacts**

Command	EFA Impact
<pre>openstack network create -- provider-network-type vlan -- provider-physical-network default --provider-segment 3320 sslnetwork1</pre>	<p>EPG Created for sslnetwork  Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d  CTAG = 3320  Note - 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d  is the neutron UUID allocated for the EPG.</p>
<pre>openstack subnet create sslsubnet1 --network sslnetwork1 --subnet- range 10.1.1.0/24 openstack subnet create sslsubnet1ipv6 --network sslnetwork1 --ip-version 6 --ipv6- address-mode=dhcpv6-stateful -- subnet-range fd00:10:0:57::1000/64</pre>	<p>DHCP EndPoints Created on EPG corresponding  to sslnetwork1. <b>VLAN Provisioned.</b>  EPG Updated  Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d  Port = lag_1 (added)</p>
<pre>openstack port create sslVirtIoTrunkPort1 --network sslnetwork1 openstack network trunk create -- parent-port sslVirtIoTrunkPort1 sslVirtIoTrunk1 openstack port create sslVirtIoSubport1 --network sslnetwork1</pre>	

**Table 9: Commands and impacts (continued)**

Command	EFA Impact
<pre>openstack network create -- provider-network-type vlan -- provider-physical-network default --provider-segment 3321 sslnetwork2</pre>	<p>EPG Created for sslnetwork2  <i>Name = 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d</i>  <i>CTAG = 3321</i></p>
<pre>openstack subnet create sslsubnet2 --network sslnetwork2 --subnet- range 11.1.1.0/24 openstack subnet create sslsubnet2ipv6 --network sslnetwork2 --ip-version 6 --ipv6- address-mode=dhcpv6-stateful -- subnet-range fd00:11:0:57::1000/64</pre>	<p>DHCP EndPoints Created on EPG corresponding to sslnetwork2. VLAN Provisioned  EPG Updated  <i>Name = 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d</i>  Port = lag_1 (added)</p>
<pre>openstack port create sslVirtIoTrunkPort2 --network sslnetwork2 openstack network trunk create -- parent-port sslVirtIoTrunkPort2 sslVirtIoTrunk2 openstack network trunk set -- subport port=sslVirtIoSubport1,segmentation -type=vlan,segmentation-id=3801 sslVirtIoTrunk2</pre>	
<pre>openstack server create --flavor m1.large --image ubuntu --port \$ (neutron port-list   grep -w 'sslVirtIoTrunkPort1'   awk '{print \$2}') sslVirtIoVM1 --availability- zone nova:Openstack116</pre>	<p>Endpoint corresponding to 'sslVirtIoTrunkPort1' added to EPG(sslnetwork1) VLAN Provisioned  EPG Updated  <i>Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d</i>  Port = lag_1 (added) - no impact already added</p>
<pre>openstack server create --flavor m1.large --image ubuntu --port \$ (neutron port-list   grep -w 'sslVirtIoTrunkPort2'   awk '{print \$2}') sslVirtIoVM2 --availability- zone nova:Openstack117</pre>	<p>Endpoint corresponding to 'sslVirtIoTrunkPort2' added to EPG(sslnetwork2) VLAN Provisioned  EPG Updated  <i>Name = 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d</i>  Port = lag_1 (added) - no impact already added  Endpoint corresponding to sslVirtIoSubport1 added to EPG(sslnetwork1) VLAN Provisioned  EPG Updated  <i>Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d</i>  Port = lag_1 (added) - no impact already added</p>
<pre>openstack port create ss2DcGwPort1 --device-owner network:dc_edge -- host <b>DCGW-1</b> --network sslnetwork1</pre>	<p>EndPoint corresponding to 'host DCGW-1' added to EPG (sslnetwork1) <b>VLAN Provisioned</b>  EPG Updated  <i>Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d</i>  Port = lag_1, <b>lag_4</b> (added)</p>
<pre>openstack port create ss2DcGwPort1 --device-owner network:dc_edge -- host <b>DCGW-1</b> --network sslnetwork2</pre>	<p>EndPoint corresponding to 'host DCGW-1' added to EPG (sslnetwork2) VLAN Provisioned  EPG Updated  <i>Name = 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d</i>  Port = lag_1, lag_4 (added)</p>

## Create a Single Segment VFPT VM Network

You can create a single segment VFPT VM network on a non-default physnet and extend the network to DC Edge.

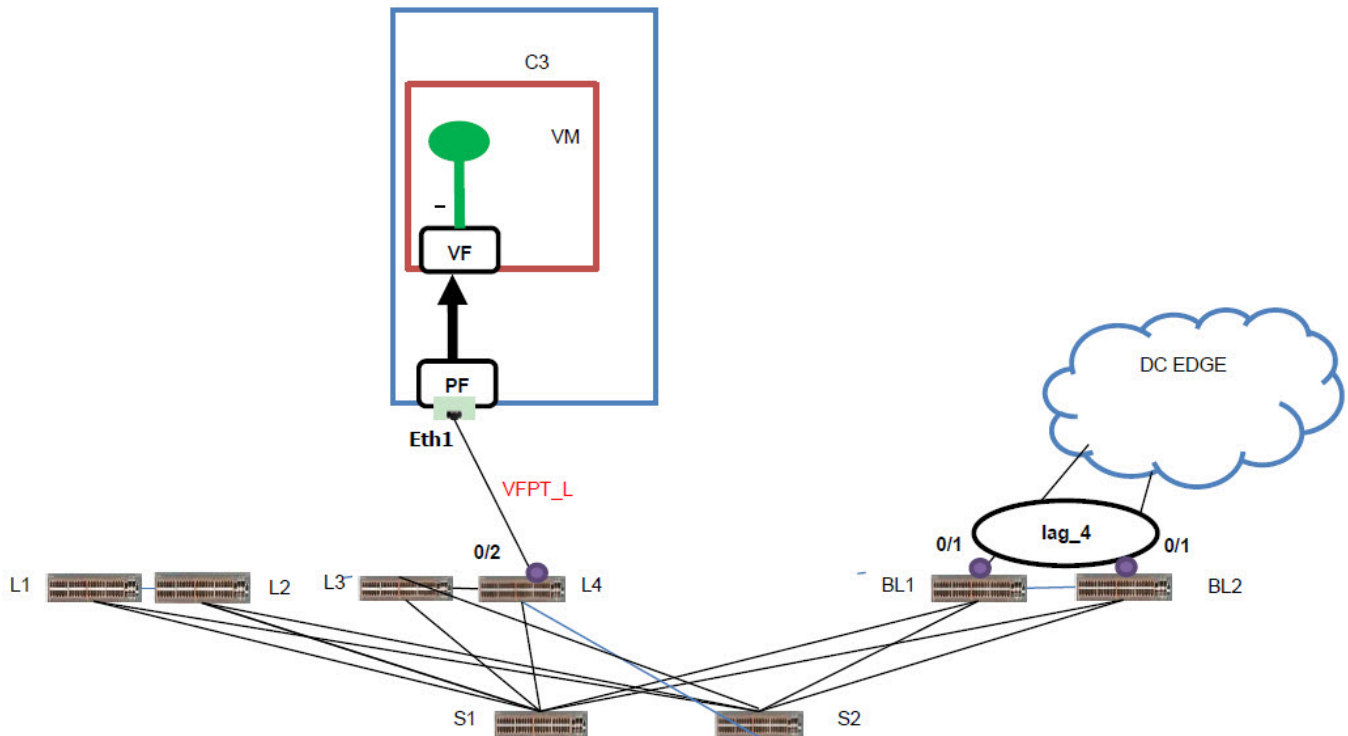


**Note**

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 a single segment VFPT VM network with DC Edge. The non-default physnet in this scenario is VFPT\_L.



**Figure 9: Overview of single segment VFPT VM network with DC Edge**

**Table 10: Commands and impacts**

Command	Impact on EFA
<pre>openstack network create -- provider-physical-network VFPT_L -- provider-network-type flat ss9_vfpt_flat_left</pre>	No impact as operations are on FLAT network type
<pre>openstack network create -- provider-physical-network VFPT_R -- provider-network-type flat ss9_vfpt_flat_right</pre>	

**Table 10: Commands and impacts (continued)**

Command	Impact on EFA
<pre>openstack subnet create ss8flatleftsubnet --network ss9_vfpt_flat_left --no-dhcp -- subnet-range 107.1.1.0/24</pre>	
<pre>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</pre>	
<pre>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</pre>	
<pre>openstack network create -- provider-network-type vlan -- provider-physical-network VFPT_L -- provider-segment <b>3390ss9network</b></pre>	<p>EPG Created for <b>ss9network</b>  Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d  CTAG = <b>3390</b>  <b>Note</b> - 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d  Is the neutron UUID allocated for the EPG</p>
<pre>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</pre>	No impact as -no-dhcp option is used
<pre>openstack port create ss9SriovSubPort1 --network ss9network1 --mac-address &lt;same- mac-as-ss9_port_vfpt_left1&gt; --vnic- type direct --fixed-ip subnet=ss9subnet1,ip- address=90.1.1.10 --fixed-ip subnet=ss9subnet1ipv6,ip- address=fd00:90:0:57::10</pre>	

**Table 10: Commands and impacts (continued)**

Command	Impact on EFA
<pre>openstack network trunk set -- subport port=ss9SriovSubPort1,segmentation- type=vlan,segmentation-id=3390 ss9SriovTrunkLagopenstack port set ss9SriovSubPort1 --device-owner compute:nova --host compute-0-10.domain.tld --device &lt;same-deviceid-as- ss9_port_vfpt_left1&gt; --binding- profile pci_slot=&lt;same-as-slot-of- ss9_port_vfpt_left1&gt; --binding- profile pci_vendor_info=&lt;same-as- vendor-of-ss9_port_vfpt_left1&gt; -- binding-profile physical_network=&lt;same-as-physnet- of-ss9_port_vfpt_left1&gt;</pre>	<p>EndPoint corresponding to ss9SriovSubPort1 added to EPG(ss9network1) <b>VLAN Provisioned</b> EPG Updated <i>Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d</i> <i>Port = L4[0/2] (added)</i></p>
<pre>openstack port create ss9DcGwPort --network ss9network1 --device- owner network:dc_edge --host <b>DCGW-1</b></pre>	<p>EndPoint corresponding to 'host DCGW-1' added to EPG (ss9network1) <b>VLAN Provisioned</b> EPG Updated <i>Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d</i> <i>Port = L4[0/2]. lag_4 (added)</i></p>

## Create a Single Segment PFPT VM Network

You can create a single segment PFPT VM network on a non-default physnet and extend the network to DC Edge.

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

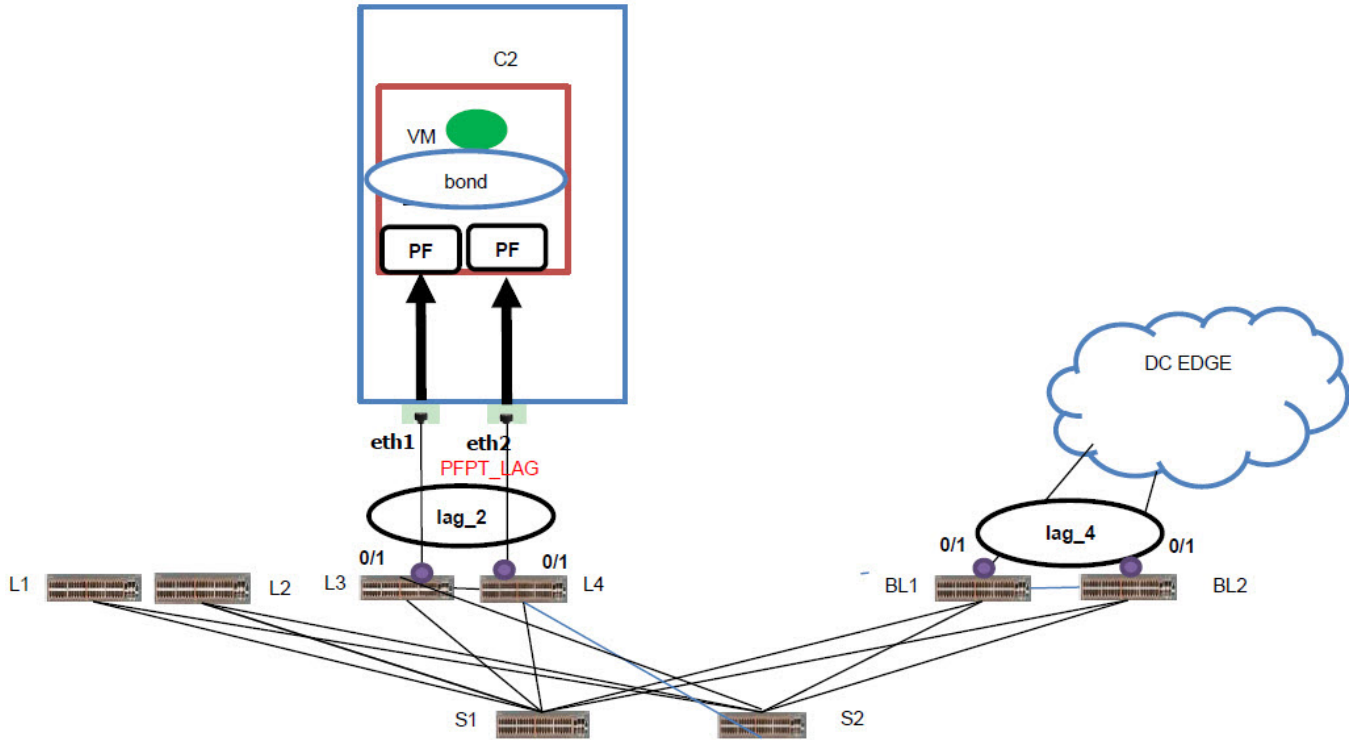


Figure 10: Overview of single segment PFPT VM network with DC Edge



**Table 11: Commands and impacts**

Command	EFA Impact
<pre> 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 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 -- subnet-range 108.1.1.0/24 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 openstack network trunk create -- parent-port ss7_port_pfpt_left ss7PFPTTrunkLag1 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 </pre>	No impact as operations are on FLAT network type
<pre> openstack network create -- provider-network-type vlan -- provider-physical-network PFPT_LAG --provider-segment <b>3370ss7network1</b> </pre>	<p>EPG Created for <b>ss7network1</b>  Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d  CTAG = <b>3370</b>  <b>Note</b> - 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d  Is the neutron UUID allocated for the EPG</p>
<pre> openstack subnet create ss7subnet1 --network ss7network1 --no-dhcp -- subnet-range 70.1.1.0/24 openstack subnet create ss7subnetlipv6 --network ss7network1 --ip-version 6 --no- dhcp --subnet-range fd00:70:0:57::1000/64 </pre>	No impact as -no-dhcp option is used.

**Table 11: Commands and impacts (continued)**

Command	EFA Impact
<pre>openstack port create ss7PFPTSubPort1 --network ss7network1 --mac-address &lt;same- mac-as-ss7_port_pfpt_left&gt; --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</pre>	
<pre>openstack network trunk set -- subport port=ss7PFPTSubPort1, segmentation- type=vlan, segmentation-id=3370</pre>	<p>Endpoint corresponding to ss7PFPTSubPort1 added to EPG(ss7network1) <b>VLAN Provisioned</b> EPG Updated Name = 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d Port = lag_2 (added)</p>
<pre>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, ip- address=fd00:70:0:57::30</pre>	<p>EndPoint corresponding to 'host DCGW-1' added to EPG (ss9network1) <b>VLAN Provisioned</b> EPG Updated Name = 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d Port = lag_2, <b>lag_4</b> (added)</p>

## Create a Multi Segment Virtio VM, PFPT, and VFPT Network

You can create default physnet and segments for PFPT\_LAG and VFPT\_L.

Perform this procedure to create default physnet and segments for PFPT\_LAG and VFPT\_L. There are Virtual Machine created on all of these segments. The network also gets extended on DC EDGE using segments.

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

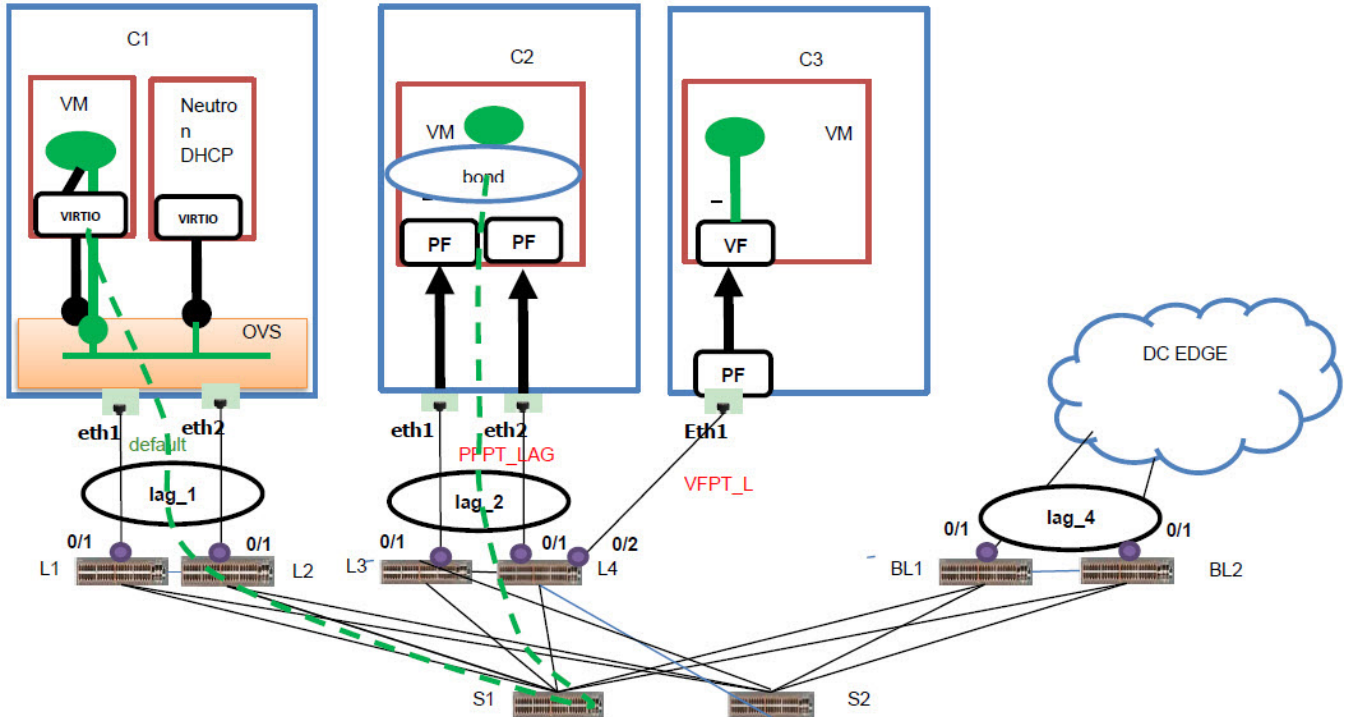


Figure 11: Overview of Virtio VM, PFPT, and VFPT segment network with DC Edge

Table 12:

Command	EFA Impact
<pre>openstack network create -- provider-network-type vlan -- provider-physical-network VFPT_L -- provider-segment 3710 ms10Network1</pre>	EPG Created for ms10Network1 Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d CTAG = 3710 <b>Note</b> - 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d Is the neutron UUID allocated for the EPG Port = L4[0/2] added
<pre>openstack network segment create -- network-type vlan --physical- network EXT1 --segment 3710 -- network ms10Network1 ms10Network1DcgwSegment</pre>	Endpoint corresponding EXT1 added to EPG(ms10Network1) <b>VLAN Provisioned</b> EPG Updated Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d Port = L4[0/2], <b>lag_4</b> (added)
<pre>openstack network segment create -- network-type vlan --physical- network PFPT_LAG --segment 3710 -- network ms10Network1 ms10Network1PFPTSegment</pre>	Endpoint corresponding EXT1 added to EPG(ms10Network1) <b>VLAN Provisioned</b> EPG Updated Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d Port = L4[0/2], <b>lag_4</b> , <b>lag_2</b> (added)
<pre>openstack network segment create -- network-type vlan --physical- network default --segment 3710 -- network ms10Network1 ms10Network1DefaultSegment</pre>	

**Table 12: (continued)**

Command	EFA Impact
<pre>openstack subnet create ms10subnet1 --network ms10Network1 --subnet- range 130.1.1.0/24 openstack subnet create ms10subnetlipv6 --network ms10Network1 --ip-version 6 --ipv6- address-mode=dhcpv6-stateful -- subnet-range fd00:0130:0:57::1000/64</pre>	
<pre>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</pre>	
<pre>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</pre>	<p>Endpoint corresponding default added to EPG(ms10Network1) <b>VLAN Provisioned</b> EPG Updated <i>Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d</i> <i>Port = L4[0/2],lag_4,<b>lag_2</b> (added)</i></p>
<pre>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</pre>	

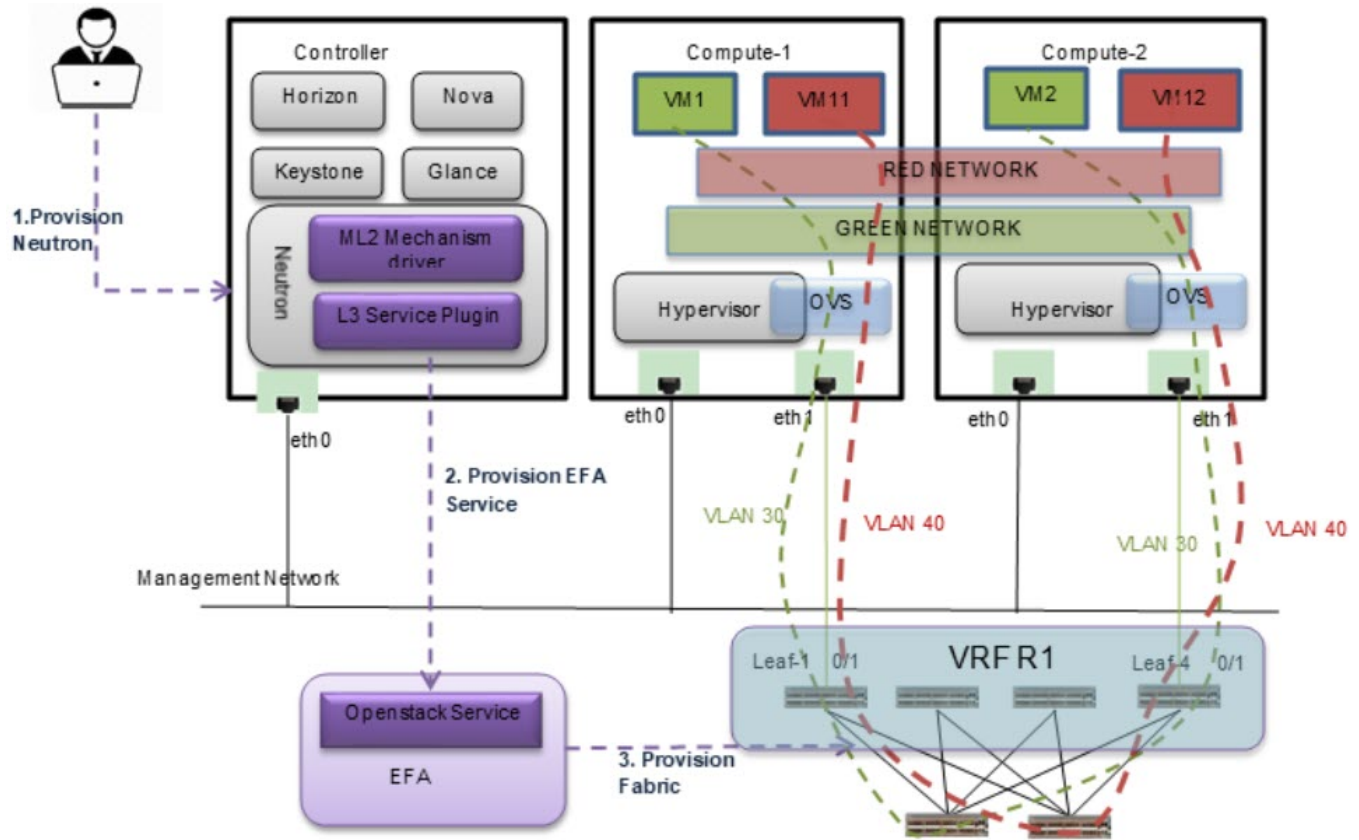
## L3 Service Plug-in Architecture

The Extreme L3 Service plug-in within the Neutron component of OpenStack proxies the Neutron API calls for network management toward the OpenStack Service running in EFA.

The OpenStack Service in EFA translates the Neutron network management calls to appropriate tenant API calls and provision the fabric with appropriate L3 networking constructs.

The L3 Service plug-in can create a distributed or centralized VRF instance on the IP fabric which corresponds to the Neutron router settings.

For distributed routers, VRF instances are created on corresponding leaf devices. For centralized routers, VRFs are created on border leaf devices only. For more details, see [Centralized Routing](#) on page 47.



**Figure 12: L3 Service plug-in**

## OpenStack Network Workflow Using L3 Service Plug-in

### About This Task

Extreme ML2 Plugin and L3 Service Plugin proxies these requests towards EFA with VLAN details about the allocated by the type driver.

Openstack Service on EFA utilizes the End Point Group (EPG) construct of Tenant Services to provision an End Point Group (EPG).

### Procedure

1. Create an OpenStack network named GREEN\_NETWORK using a Neutron CLI command.

```
openstack network create GREEN_NETWORK (assume VLAN 30 is allocated by type driver)
openstack subnet create GREEN_SUBNET_IPV4 --subnet-range 10.0.0.0/24 --network GREEN_NETWORK
```

2. Create a subnet, GREEN\_SUBNET.

```
openstack subnet create GREEN_SUBNET_IPV4 --subnet-range 10.0.0.0/24 --network GREEN_NETWORK
```

3. Create a Virtual Machine VM1 on the Compute-1 attached to GREEN\_NETWORK.

```
openstack server create --nic
    net-id=$(neutron net-list | awk '/GREEN_NETWORK/ {print $2}') --image
cirros-0.3.4-x86_64-uec --flavor m1.tiny
--availability-zone nova:Compute-1 VM1
```

4. Create a Virtual Machine VM2 on the Compute-2 attached to GREEN\_NETWORK.

```
openstack server create --nic
    net-id=$(neutron net-list | awk '/GREEN_NETWORK/ {print $2}') --image
cirros-0.3.4-x86_64-uec --flavor m1.tiny --availability-zone
nova:Compute-2 VM2
```

5. Create a second OpenStack network by the name RED\_NETWORK using Neutron CLI command.

```
openstack network create RED_NETWORK (assume
    VLAN 30 is allocated by type driver)
openstack subnet create RED_SUBNET_IPV4 --subnet-range 9.0.0.0/24 -network
RED_NETWORK
```

6. Create a Virtual Machine VM11 on the Compute-1 attached to RED\_NETWORK.

```
openstack server create --nic
    net-id=$(neutron net-list | awk '/ RED_NETWORK / {print $2}') --image
cirros-0.3.4-x86_64-uec --flavor m1.tiny
--availability-zone nova:Compute-1 VM11
```

7. Create a Virtual Machine VM12 on the Compute-2 attached to RED\_NETWORK.

```
openstack server create --nic
    net-id=$(neutron net-list | awk '/ RED_NETWORK / {print $2}') --image
cirros-0.3.4-x86_64-uec --flavor m1.tiny
--availability-zone nova:Compute-2 VM12
```

8. Create a Router R1 and add the two networking instances, GREEN\_SUBNET and RED\_SUBNET as part of the Router.

```
neutron router-create R1
neutron router-interface-add R1 GREEN_SUBNET_IPV4
neutron router-interface-add R1 RED_SUBNET_IPV6
```

## Results

Extreme ML2 Plugin and L3 Service Plugin proxies these requests towards EFA with VLAN details about the allocated by the type driver.

Openstack Service on EFA utilizes the End Point Group (EPG) construct of Tenant Services to provision an End Point Group (EPG).



#### Note

You can also create IPv6 subnet using the following command:

```
openstack subnet create --ip-version 6 --network GREEN_NETWORK --subnet-range
10:2000::/64
--gateway 10:2000::1 GREEN_SUBNET_IPV6
openstack router add subnet R1 GREEN_SUBNET_IPV6
```

Port-based addition of router interface is supported.

**Table 13: EFA impact of L3 provisioning**

VM	Neutron Network	Tenant Service(EPG)
VM1	GREEN_NETWORK(VLAN30) UUID =74cbf489-f3d9-41c7- bbb2-6cb7df33da6d	74cbf489-f3d9-41c7- bbb2-6cb7df33da6ds • Endpoint: eth 0/1 on Leaf-1
VM2	GREEN_NETWORK(VLAN30) UUID =74cbf489-f3d9-41c7- bbb2-6cb7df33da6d	• EndPoint: eth 0/1 on Leaf-2 • CTAG 30 • AnyCast 10.0.0.1 • VRF R1(UUID=99cbf489- f3d9-41c7- bbb2-6cb7df33da03)
VM11	RED_NETWORK(VLAN40) GREEN_NETWORK(VLAN30) UUID =89cbf489-f3d9-41c7- bbb2-6cb7df33da02	89cbf489-f3d9-41c7- bbb2-6cb7df33da02 • Endpoint: eth 0/1 on Leaf-1 • EndPoint: eth 0/1 on Leaf-2
VM12	RED_NETWORK(VLAN40) UUID =89cbf489-f3d9-41c7- bbb2-6cb7df33da02	• CTAG 40 • Anycast 9.0.0.1 • VRF R1(UUID=99cbf489- f3d9-41c7- bbb2-6cb7df33da03)

EPG provisioning on the fabric creates an L2 network on the fabric spanning VM1, VM2, VM11, and VM12 with necessary fabric mappings. This creates the necessary constructs to establish an end-to-end connectivity between the two sets of Virtual Machines VM1 and VM2, VM11 and VM12. The VRF configuration enables routing between the two networks.

## Centralized Routing

Routers can be configured either in centralized mode or distributed mode.

In a centralized router, the routing configurations are configured only on the border leaf pairs. In case of distributed mode, the routing configurations are configured on the corresponding leaf nodes where the endpoints reside.

During router creation, you can provide centralized mode or distributed mode as input.

- `openstack router create R1 --distributed`
- `openstack router create R2 --centralized`

The default option is centralized. Use the following command to create centralized routing:

### **openstack router create R2**

The L3 service plug-in passes this information to EFA and the EFA Tenant Service creates VRF or routing configurations on border leaves or on leaf nodes based on the configuration mode.



#### Note

For the version 2.4 release, the OpenStack integration works with only a single pair of border leaf devices. A centralized routing instance is created on the single pair of border leaf devices. You must ensure that only a single border leaf pair is added as part of the fabric creation.

## L3 Flavors

L3 flavors allow multiple backends in a single Neutron deployment.

L3 flavors allow multiple backends in a single Neutron deployment, each with its own logical network topology, completely separated from each other. It allows incremental migrations from one backend to the other.

Command	Description
<code>openstack network flavor profile create --driver networking_extreme.l3.l3_flavor.ExtremeL3ServiceProvider</code>	Creates Flavor Profile <flavorprofileid>
<code>openstack network flavor create --service-type L3_ROUTER_NAT l3_extreme</code>	Creates a network flavor
<code>openstack network flavor add profile l3_extreme &lt;flavorprofileid&gt;</code>	Adds the Profile to the Flavor using the <flavorprofileid>
<code>neutron router-create router1 --flavor l3_extreme</code>	Creates a Neutron router using the create flavor



#### Note

For Journaling support in flavors create a flavor profile using **openstack network flavor profile create --driver networking\_extreme.l3.l3\_flavor\_v2.ExtremeL3ServiceProviderv2**.

Replace `l3_extreme` in the table above with `l3_extreme_v2` and follow the same steps for the journaling flavor.

For changes required in the Neutron configuration file, see [Configure Neutron to Connect to EFA](#) on page 28 and [Enable Journaling](#) on page 29.

When you create a router with a specific flavor the flavor framework looks up the service profiles for that flavor and creates the router using the associated drivers. Once a driver is selected for a router, the router or driver association is stored in the database, so any future operations on the router can look up the driver without going through the flavor framework.



A centralized or distributed router can be specified on router creation along with the flavor. By default, a centralized router is created.

- Distributed Router: `neutron router-create router1 --flavor l3_extreme - distributed='True'`
- Centralized Router: `neutron router-create router1 --flavor l3_extreme - distributed=False`

## L3 IPV6 MTU

Neutron provides multiple ways to specify MTU for networks.

You can provide a global value in `neutron.conf` using `global_physnet_mtu`. It applies to all networks.

```
/etc/neutron/neutron.conf
[DEFAULT]
global_physnet_mtu = <mtu_value>
```

You can provide a specific value when creating the network using the `--mtu` option. This value takes higher precedence and overrides the global value.

```
openstack network create -mtu <mtu_value> GREEN_NETWORK
```

The MTU value for each network is captured by Extreme's plugins and passed to EFA. EFA then enforces the value on the EPG as part of its `ipv6-nd-mtu` option. This value is per ctag and is enforced on the VE interface corresponding to the ctag. EFA configures the MTU value on the specific VE interface on the switch.

The value is applied to the SLX switch when router-interface is associated with the network in Neutron - either when the router is added to a subnet that is associated with the network or when a port on the network is added to a router.

The following is a sample SLX configuration.

```
Rack1-Device1# show running-config interface Ve 11
vrf forwarding ten1vrfl
ip anycast-address 10.10.11.1/24
ipv6 anycast-address fd00:10:0:57::1000/64
ipv6 nd mtu 2100
no shutdown !
```

## L3 IPV6 ND/RA

When creating subnets in Neutron, you can provide values to specify IPv6 Router Advertisement mode and IPv6 address mode.

Values to specify IPv6 Router Advertisement mode and IPv6 address mode can be one of: `dhcpv6-stateful`, `dhcpv6-stateless` or `slaac` (stateless address autoconfiguration).

The subnet settings take effect when it is added to a router.

Example:

```
openstack subnet create --ip-version 6 --subnet-range fd00:10:0:57::/64
--gateway fd00:10:0:57::1000 --network net1 subnet1 --ipv6-address-mode dhcpv6-stateful
--ipv6-ra-mode dhcpv6-stateful
```

```
openstack router subnet add Router1 subnet1
```

Based on the value passed in `ipv6-ra-mode` and `ipv6-address-mode`, the A, M, O bits are set in the IPv6 Router Advertisements sent from the external router.

- A – autoconfig flag
- M – managed-config-flag
- O – Other-config-flag

The following table explains the router advertisement flag settings and expected guest instance behavior.

**Table 14: Router advertisement flag settings and expected guest instance behavior**

S number	ipv6 ra mode	ipv6 address mode	A, M, O bits set in Router Advertisements sent from Extreme Router	Guest instance behavior
1	slaac	slaac	1, 0, 0	Guest instance obtains IPv6 address from non-OpenStack router using SLAAC.
2	dhcpv6-stateful	dhcpv6-stateful	0, 1, 1	Guest instance obtains IPv6 address from dnsmasq using DHCPv6 stateful and optional information from dnsmasq using DHCPv6.
3	dhcpv6-stateless	dhcpv6-stateless	1, 0, 1	Guest instance obtains IPv6 address from non-OpenStack router using SLAAC and optional information from dnsmasq using DHCPv6.



**Note**

Other combinations for `ipv6-ra-mode` and `ipv6-address-mode` in the above command are considered invalid.

Sample SLX configuration:

```
Leaf4# show running-config interface ve
interface Ve 1330
vrf forwarding vrf_2052
ipv6 anycast-address fd00:10:0:57::1000/64
ipv6 nd managed-config-flag
```

```
ipv6 nd other-config-flag
ipv6 nd prefix fd00:10:0:57::/64 2592000 604800 no-autoconfig
no shutdown
!
```

- When the M bit is set to 1, `managed-config-flag` is set on the VE interface
- When the O bit is set to 1, `other-config-flag` is set on the VE interface.
- When the A bit is set to 0, `no-autoconfig` is added to the `ipv6 nd prefix` command on the VE interface.

## Journaling

The V2 driver is set to tackle problems encountered in the V1 driver while maintaining feature parity. The major design concept of the V2 driver is journaling. Instead of passing the calls directly to the EFA, they get registered in the journal table which keeps a sort of queue of the various operations that occurred on Neutron and must be mirrored to the controller.

The journal is processed mainly by a journaling thread which runs periodically and checks if the journal table has any entries in need of processing. Additionally, the thread is triggered in the postcommit hook of the operation (where applicable).

For example, in case of create network again, after it gets stored in the Neutron DB by the ML2 plugin, the EFA mechanism driver stores a “journal entry” representing that operation and triggers the journaling thread to take care of the entry.

The journal entry is recorded in the pre-commit phase (whenever applicable) so that in case of a commit failure the journal entry gets aborted along with the original operation, and there’s nothing extra needed.

Refer to [https://docs.openstack.org/networking-odl/latest/contributor/drivers\\_architecture.html](https://docs.openstack.org/networking-odl/latest/contributor/drivers_architecture.html).



### Note

Journaling is available to ML2 and Layer3.

## Journaling Console CLI Commands

The **efa-journal** console command is available to view journal entries and to reset or clear failed journal entries.

Full details of the commands **efa-journal list**, **efa-journal reset** and **efa-journal clear** commands are provided in the [EFA OpenStack Service Command Reference](#) on page 61.

## OpenStack Utilities

The EFA Sync utility enables you to sync all entries from Neutron to EFA.

The EFA Health utility enables you to verify that OpenStack is connected with EFA.

## Sync EFA with Neutron

The OpenStack integration enables you to sync all entries from Neutron to EFA.

### About This Task

When EFA goes out of sync with Neutron configuration, execute **efa-sync** to sync all entries from Neutron to EFA. For example, if there is a network and associated ports that are present on Neutron but are not present on EFA, running **efa-sync** creates the entities on EFA. Similarly, if there are stale entries on EFA, they are cleaned up.

You can display summary or detailed information about entities that are out of sync between Neutron and EFA configurations.

### Procedure

1. Sync EFA with Neutron.

```
$ efa-sync execute
Starting Sync
Syncing Networks..
  Add Network to EFA Id: 9e63bc57-568f-488e-9214-21db3fd3cd12 (Succeeded)
Syncing Ports..
  Add Port to EFA Id: 9e63bc57-568f-488e-9214-21db3fd3cd12-10.25.225.11-port-channel-
lag_1 (Succeeded)
  Add Port to EFA Id: 9e63bc57-568f-488e-9214-21db3fd3cd12-10.25.225.46-port-channel-
lag_1 (Succeeded)
Completed Sync
```

2. Display summary information about entities that are out of sync between Neutron and EFA configurations.

```
$ efa-sync check-summary
05-November-2020 11:04:15 : Starting Check
Summary:
  Neutron Networks to be added to EFA : 1
  Neutron Networks to be deleted from EFA : 0
  Neutron Ports to be added to EFA : 2
  Neutron Ports to be deleted from EFA : 0
  Total # of resources to be resynced : 3
05-November-2020 11:04:16 : Completed Check
```

3. Display detailed information about those entities.

```
$ efa-sync check-detail
05-November-2020 11:04:22 : Starting Check
  Neutron Networks to be added to EFA :
    Id: 9e63bc57-568f-488e-9214-21db3fd3cd12
  Neutron Ports to be added from EFA :
    Id: 9e63bc57-568f-488e-9214-21db3fd3cd12-10.25.225.11-port-channel-
lag_1
    Id: 9e63bc57-568f-488e-9214-21db3fd3cd12-10.25.225.46-port-channel-lag_1
05-November-2020 11:04:22 : Completed Check
```

## Verify EFA Health

The EFA Health utility is provided as part of the installation package.

The health utility enables you to verify connectivity to EFA, database write operations, and the status of the EFA Kubernetes cluster.



### Tip

This feature creates a VRF for testing the health status. So the tenant `vrf-count` option will account for this additional VRF.

Run `efa-health show` to show a summary health status.

```
$ efa-health show
EFA Host      : efa.extremenetworks.com
EFA Health    : UP
Database Health : UP
Primary Node  : tpvm
Cluster Status : 14/14 Pods Running
```

Run `efa-health show --advanced` to show a detailed health status.

```
$ efa-health show --advanced
EFA Host      : efa.extremenetworks.com
EFA Health    : UP
Database Health : DOWN (Unable to connect to Tenant Service)
Master Node   : tpvm
  rabbitmq-0: Running
  efa-api-docs-5886cb6fb4-w4qsg      : Running
  gosnmp-service-56dc6c46c8-dzrlh    : Running
  gosystem-service-67767c4796-xncrn  : Running
  goopenstack-service-84fd5784f5-k88hw : Running
  goauth-service-56f8665858-vdvrz    : Running
  gorbac-service-75c6f5f976-5pbr5    : Running
  gonotification-service-79b4989ff6-dxw dv : Running
  goinventory-service-75847f48d9-4l9vr : Running
  gofabric-service-77fddfc474-dqw4d   : Running
  goraslog-service-5b8c9c979b-ztt71   : Running
  gohyperv-service-785d457475-bftvv   : Running
  gotenant-service-64dd55b96f-f9wqd   : Running
  govcenter-service-5f6c64b5b-h58fg   : Running
```

Sample output of advanced `efa-health show` when the `gotenant` pod is down, indicating an issue with creating routers.

```
$ efa-health show --advanced
EFA Host      : efa.extremenetworks.com
EFA Health    : UP
Database Health : DOWN (Unable to connect to Tenant Service)
Master Node   : tpvm

Cluster Pods Status (13/15 Pods Running):
  rabbitmq-0      : Running
  efa-api-docs-5886cb6fb4-r52mq      : Running
  gosnmp-service-56dc6c46c8-6zgrk    : Running
  gosystem-service-67767c4796-4tr2v  : Running
  goopenstack-service-6dcdb44d95-7xgcb : Running
  gorbac-service-7b4d67d9d8-wvdvn    : Running
  goauth-service-85ff7bcc7d-ckf7m    : Running
  gofabric-service-68f9965ccd-w4fjm   : Running
  goinventory-service-6c5d5846d9-fbq2n : Running
  gonotification-service-79cdd65d76-2m2j f : Running
  goraslog-service-5f4b5ff756-trflk   : Running
  govcenter-service-78f8454f84-jg42h  : Running
```

```
gohyperv-service-7d776d4f67-tgwt6      : Running
gotenant-service-584c47b9df-kzxtm      : Pending
gotenant-service-584c47b9df-cjbwr      : Terminating
```

## Multiple VIM/VPOD Instances

---

Each VIM/VPOD instance or Open Stack 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.



```

+-----+
|      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 ---

```



## Scale-in and Scale-out of Compute Nodes

**Table 15: Scale-in of compute nodes**

Scale-in	EFA Impact
Link removal from default physical network	VLAN provisioning for endpoints is done when Neutron ports are unbound to a host or compute. Hence, removing a link which belongs to the default physical network does not have any impact, only the link-mapping table is updated.
Link removal from non-default physical networks	For non-default physical networks, VLAN provisioning is done at the network level. Hence, the links that are removed from non-default networks are automatically removed from the existing networks. On EFA, the interface is removed from the corresponding EPGs. If there are any errors during EFA unbinding, the operation succeeds. This behavior is consistent with the delete port operation.
Links bound to port-channels on non-default physical networks	Removing a link which maps to a 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 16: Scale-out of compute nodes**

Scale-out	EFA Impact
Link addition to default physical network	VLAN provisioning of endpoints is done when Neutron ports are bound to a host or compute. Hence, adding a link which belongs to the default physical network does not have any impact, only the 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 16: 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 requires no further changes for EFA.
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 shut down 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 | ml.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 |
+-----+-----+-----+-----+-----+-----+
| 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	<b>HostC</b>
...	...

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
```

## 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:~# 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:~#
```

2. Restart Neutron to apply the token and certificate.

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



# EFA OpenStack Service Command Reference

---

- [efa openstack debug](#) on page 62
- [efa openstack execution](#) on page 64
- [efa openstack network show](#) on page 65
- [efa openstack network-interface show](#) on page 66
- [efa openstack router show](#) on page 67
- [efa openstack router-interface show](#) on page 68
- [efa openstack subnet show](#) on page 69
- [efa openstack sync start](#) on page 70
- [efa-health show](#) on page 71
- [efa-journal](#) on page 72
- [efa-sync execute](#) on page 74
- [openstack network efa-topology-link-map create](#) on page 76
- [openstack network efa-topology-link-map delete](#) on page 77
- [openstack network efa-topology-link-map list](#) on page 78

## 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
```

Deletes the network and its summary information.

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

Deletes the network interface and its summary information.

```
efa openstack debug router delete --router-id
```

Deletes the router and its summary information.

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

Deletes the router interface and its summary information.

```
efa openstack debug tenant cleanup --name tenant name
```

Deletes a network.

### 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 network commands

**network-interface**

Specifies network interface commands

**tenant**

Specifies tenant commands

**router**

Specifies router commands

**router-interface**

Specifies router interface commands

**neutron-id** *string*

Neutron ID of the network

**router-id** *string*

Router ID

**subnet-id** *string*

Subnet ID

**name** *string*

Tenant name

**help** *string*

Displays help on the command

## 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 is 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 is 10.

#### **show**

Displays the list of executions.

#### **--status**

Filters the executions based on the status (failed, succeeded, all). Default is all.



## efa openstack network show

---

Displays OpenStack network information.

### Syntax

```
efa openstack network show
```

### Examples

This example shows typical results.

```
efa openstack network show
+-----+-----+-----+-----+
|           Neutron ID           | Tenant | MTU | CTAG |
+-----+-----+-----+-----+
| 9237b75d-6fa9-404b-8fab-979b7d384c40 | RegionOne | 2100 | 3 |
+-----+-----+-----+-----+
| 23313c80-0b91-455a-8dce-91e6cb7424a7 | RegionOne | 2100 | 2 |
+-----+-----+-----+-----+
```

## efa openstack network-interface show

Displays OpenStack network-interface information.

### Syntax

```
efa openstack network-interface show
```

### Examples

This example shows typical results.

```
# 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
```

### Examples

This example shows typical results.

```
efa openstack router show
+-----+-----+-----+
+-----+
|           Router ID           |           VRF Name           | Tenant |
Routing type |
+-----+-----+-----+
| bc482e46-8896-442c-a29d-2dd76abca2ae | bc482e468896442ca29d2dd76abca2ae | RegionOne |
centralized |
+-----+-----+-----+
| 99ec4fd8-fbc4-4117-a529-5e9a40049906 | 99ec4fd8fbc44117a5295e9a40049906 | RegionOne |
distributed |
+-----+-----+-----+
+-----+
```

## efa openstack router-interface show

---

Displays OpenStack router-interface information.

### Syntax

```
efa openstack router-interface show
```

### Examples

This example shows typical results.

```
# 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
```

### Examples

This example shows typical results.

```
efa openstack subnet show
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          Subnet ID          |          Network ID          |
| CIDR      | Gateway IP | IPv6RAMode | IPv6AddressMode |
+-----+-----+-----+-----+
| 869dbbcd-2c80-440a-a42f-1e85cab60b98 | 9237b75d-6fa9-404b-8fab-979b7d384c40 |
| 110:2000::/64 | 110:2000::1 | dhcpv6-stateful | |
+-----+-----+-----+-----+
| 063eec5a-dbb3-480b-b1d0-1d7fc4715b67 | 23313c80-0b91-455a-8dce-91e6cb7424a7 |
| 10:2000::/64 | 10:2000::1 | | |
+-----+-----+-----+-----+
| ea233dda-b692-4a46-adc8-ea58ac07f4d2 | 23313c80-0b91-455a-8dce-91e6cb7424a7 |
| 8.8.8.0/24 | 8.8.8.1 | | |
+-----+-----+-----+-----+
| db91323d-fcdd-4450-87b2-57bb7e033c0c | 9237b75d-6fa9-404b-8fab-979b7d384c40 |
| 18.8.8.0/24 | 18.8.8.1 | | |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

## efa openstack sync start

---

Syncs entries such as networks, network interfaces, routers, and router interfaces from OpenStack to a tenant if they are out of sync.

### Syntax

```
efa openstack sync start [ --tenant tenant name | --help ]
```



#### Note

There is no output for this command.

### Parameters

#### **--tenant**

The name of the tenant

#### **--help**

Provides help for execution

## efa-health show

Verifies connectivity with EFA, database write operations, and the status of the EFA Kubernetes cluster.

### Syntax

```
efa-health show [--advanced | --help ]
```

### Parameters

#### **--advanced**

Lists advanced status.

#### **--help**

Provides help for execution.

### Examples

This example shows EFA status as UP.

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

The following is an example of the output from a basic efa-health show execution:

```
$ efa-health show
EFA Host      : efa.extremenetworks.com
EFA Health    : UP
Database Health : UP
Primary Node   : tpvm
Cluster Status : 14/14 Pods Running
```

The following is an example of advanced efa-health show:

```
$ efa-health show --advanced
EFA Host      : efa.extremenetworks.com
EFA Health    : UP
Database Health : UP
Primary Node   : tpvm
Cluster Status (14/14 Pods Running):
rabbitmq-0: Running
efa-api-docs-5886cb6fb4-w4qsg      : Running
gosnmp-service-56dc6c46c8-dzrlh    : Running
gosystem-service-67767c4796-xncrn  : Running
goopenstack-service-84fd5784f5-k88hw : Running
goauth-service-56f8665858-vdvrz    : Running
gorbac-service-75c6f5f976-5pbr5    : Running
gonotification-service-79b4989ff6-dxwdv : Running
goinventory-service-75847f48d9-419vr : Running
gofabric-service-77fddfc474-dqw4d  : Running
goraslog-service-5b8c9c979b-ztt71  : Running
gohyperv-service-785d457475-bftvv   : Running
gotenant-service-64dd55b96f-f9wqd   : Running
govcenter-service-5f6c64b5b-h58fg   : Running
```

The following is an example of advanced efa-health show when the gotenant pod is down, indicating an issue with creating routers:

```
$ efa-health show --advanced
EFA Host      : efa.extremenetworks.com
EFA Health    : UP
Database Health : DOWN (Unable to connect to Tenant Service)
Primary Node  : tpvm
Cluster Pods Status (13/15 Pods Running):
rabbitmq-0   : Running
efa-api-docs-5886cb6fb4-r52mq   : Running
gosnmp-service-56dc6c46c8-6zgrk : Running
gosystem-service-67767c4796-4tr2v : Running
goopenstack-service-6dcdb44d95-7xgcb : Running
gorbac-service-7b4d67d9d8-wvdvn : Running
goauth-service-85ff7bcc7d-ckf7m : Running
gofabric-service-68f9965ccd-w4fjm : Running
goinventory-service-6c5d5846d9-fbq2n : Running
gonotification-service-79cdd65d76-2m2jf : Running
goraslog-service-5f4b5ff756-trflk : Running
govcenter-service-78f8454f84-jg42h : Running
gohyperv-service-7d776d4f67-tgwt6 : Running
gotenant-service-584c47b9df-kzxtm : Pending
gotenant-service-584c47b9df-cjbwr : Terminating
```

## efa-journal

Performs actions on journal entries.

### **efa-journal list**

Lists all the journal entries that are present in the Extreme journaling DB and their statuses.

### **efa-journal reset** [--ids | --help ]

Resets failed entries retry count and retries the operations again. The option works for all failed entries or selected failed entries.

### **efa-journal clear** [--ids | --help ]

Removes failed entries from journaling. The option works for all failed entries or selected failed entries.

#### **--ids** *Journal UUID*

List of failed IDs to reset or clear. If not provided, the command resets or clears all failed entries.

#### **--help**

Prints help for the command.

This example lists all the journal entries and their statuses as present in the extreme journaling DB.

```
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| Id |          Journal UUID          | Object Type |          Object          |
| UUID | | Operation | state | Retry Count | |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| 301 | 3ebf2416-366b-4c11-9a2e-8c058c219555 | network | 108755af-c7fd-4590- | |
| ac00-5359f76b7ba4 | create | processing | 0 | |
| 302 | 78884b55-9dfb-4dbb-98ef-1916a5f932f8 | network | 78b7b7e8-ea64-420e- |
| afd8-5349485afe22 | create | processing | 0 | |
```



```

| 303 | ef3b4637-a899-49e0-9f7d-28dda671685f | network | 477714ac-83d1-4ce5-
a56d-88d8bc4b9536 | create | processing | 0 |
| 304 | 7d662b57-07b3-4f71-b084-ab89604dc097 | network | fc2549af-6eec-47a5-
bc78-1831f362b6b3 | create | pending | 0 |
| 305 | 9c1a8131-4232-4f9b-ac5e-b694c5a1168c | network | 9e64c55e-8895-491d-91ed-
cfc81cd5586a | create | pending | 0 |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+

```

The following example resets a specific failed entry:

```
$ sudo efa-journal reset 848ca333-9e78-46e4-8f54-096453347d55 5dc0bf8b-674c-4c9f-944e-
dc048bd184f0
```

The following example clears a specific failed entry:

```
$ sudo efa-journal clear 5dc0bf8b-674c-4c9f-944e-dc048bd184f0
848ca333-9e78-46e4-8f54-096453347d55
```

## efa-sync execute

Syncs EFA with Neutron. Displays summary or detailed information about entities that are out of sync between Neutron and EFA configurations.

The command also logs console output into efa-sync-console.log.

The efa-sync tool uses Neutron APIs and keystone authentication. The authentication parameters are selected from the neutron.conf file. Before running efa-sync, make sure that all the parameters under the [keystone-authtoken] section are set to the correct values.



### Note

In case of default OpenStack settings in neutron.conf, the following parameters need to be edited to 'default' -- lowercase 'd' under [keystone\_authtoken]:

```
project_domain_name = default
user_domain_name = default
```

## Syntax

**efa-sync execute**

**efa-sync check-summary**

**efa-sync check-detail**

This example syncs EFA with Neutron.

```
$ efa-sync execute
Starting Sync
Syncing Networks..
  Add Network to EFA Id: 9e63bc57-568f-488e-9214-21db3fd3cd12
  (Succeeded)
Syncing Ports..
  Add Port to EFA Id:
  9e63bc57-568f-488e-9214-21db3fd3cd12-10.25.225.11-port-channel-lag_1
  (Succeeded)
  Add Port to EFA Id:
  9e63bc57-568f-488e-9214-21db3fd3cd12-10.25.225.46-port-channel-lag_1
  (Succeeded)
Completed Sync
```

This example displays summary information about entities that are out of sync between Neutron and EFA configurations.

```
$ efa-sync check-summary
05-November-2020 11:04:15 : Starting Check
Summary:
  Neutron Networks to be added to EFA : 1
  Neutron Networks to be deleted from EFA : 0
  Neutron Ports to be added to EFA : 2
  Neutron Ports to be deleted from EFA : 0
  Total # of resources to be resynced : 3
05-November-2020 11:04:16 : Completed Check
```

This example displays detailed information about entities that are out of sync between Neutron and EFA configurations.

```
$ efa-sync check-detail
05-November-2020 11:04:22 : Starting Check
  Neutron Networks to be added to EFA :
    Id: 9e63bc57-568f-488e-9214-21db3fd3cd12
  Neutron Ports to be added from EFA :
    Id: 9e63bc57-568f-488e-9214-21db3fd3cd12-10.25.225.11-port-channel-
lag_1
    Id: 9e63bc57-568f-488e-9214-21db3fd3cd12-10.25.225.46-port-channel-lag_1
05-November-2020 11:04:22 : Completed Check
```

## openstack network efa-topology-link-map create

---

Creates a topology link

### Syntax

```
openstack network efa-topology-link-map create [--host HOST | --nic NIC |  
--provider-network PROVIDER_NETWORK | --port PORT | --po-name PO_NAME  
| --help ]
```

### Parameters

**host**

Host name of the compute node which connects to the switch

**nic**

Physical NIC on the compute host which connects to the switch

**provider-network**

Provider network name on compute host. For example, physnet1 (default).

**port**

Switch port to which the physical NIC on the compute host is connected

**po-name**

Switch PO name where physical ports are aggregated to NIC where compute host is connected.

**help**

Displays help for the command

### Examples

```
# 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
```

## openstack network efa-topology-link-map delete

---

Deletes a topology link

### Syntax

```
openstack network efa-topology-link-map delete [--help ]
```

```
openstack network efa-topology-link-map delete [-h] <link> [<link> ...]
```

Specify the UUID of the link to delete.

### Examples

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

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

Lists topology links

### Syntax

```
openstack network efa-topology-link-map list [--help ]
```

### Examples

The following example configures....

```
# openstack network efa-topology-link-map list
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
| ID                                     | Host           | Nic | Provider Network |
Switch      | Port | Po Name |                                     |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
| 08fa6b52-8c25-4193-8e4b-00ce3d81f392 | DCGW1          |     | external_nw      |
10.20.246.8 | 0/14 | lag_4   |                                     |
| 0977490d-f07a-4b2c-b683-8359bde19fcc | niyer-devstack | eth3 | physnet3         |
10.20.246.8 | 0/15 |         |                                     |
| 14486fd7-1bd7-4ba0-a159-b2e76f775fdd | niyer-devstack | eth4 | default          |
10.20.246.8 | 0/16 |         |                                     |
| 4ce69af9-03ce-49d7-a0dc-0b5c64e25ded | niyer-devstack | eth1 | physnet11       |
10.20.246.8 | 0/13 | lag_1   |                                     |
| c3879465-ecdd-492b-918e-981c104005ba | DCGW1          |     | external_nw      |
10.20.246.7 | 0/14 | lag_4   |                                     |
| c9663e8d-f91c-44ff-b4a1-fb97e603bda2 | niyer-devstack | eth0 | physnet11       |
10.20.246.7 | 0/13 | lag_1   |                                     |
| daf3c58a-9b24-4069-afe8-94d2a20004c9 | niyer-devstack | eth2 | physnet2         |
10.20.246.7 | 0/15 |         |                                     |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```



# Appendix: Neutron REST Endpoints

[Neutron REST Endpoints](#) on page 79

## 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 17: 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 NIC is connected
po_name	Body	String	LAG as created on EFA for the NICs

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

Deletes a topology link from its associated resources

**Table 18: Request parameters**

Name	IN	Type	Description
id	Path	UUID	Topology ID

POST

/v2.0/efa\_topologies

Create a link on the efa\_topology object

**Table 19: Request 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 NIC is connected
po_name	Body	String	LAG as created on EFA for the NICs

```
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](#) on page 81

[Multi-Segment Network](#) on page 84

## SR-IOV Network

---

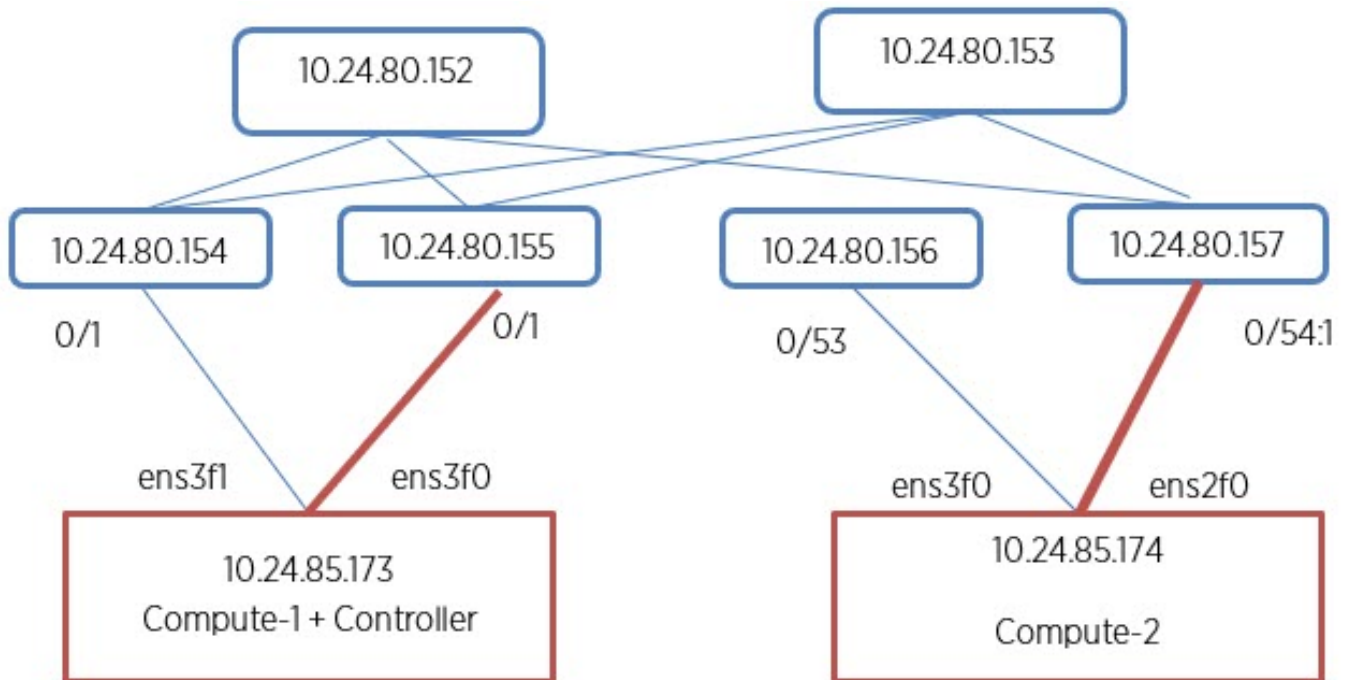
Single Root I/O Virtualization (SR-IOV) allows a single physical NIC to appear as multiple physical NICs. All NICs in the VM must be tagged to the appropriate VLAN.

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 14: SR-IOV network (VF passthrough)**

## Create PCI Passthrough Allowed List

### Procedure

1. Configure the PCI passthrough allowed list 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"}, {"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 `m12_conf.ini` file on each controller.

```
[m12]
tenant_network_types = vlan
type_drivers = vlan
mechanism_drivers = openvswitch, sriovnicswitch, extreme_efa
[m12_type_vlan]
network_vlan_ranges = physnet1:2:500
```

2. Ensure that `sriovnet` is configured for the selected network type in the `m12_conf.ini` file on each controller.

```
[m12_type_vlan]
network_vlan_ranges = sriovnet1:100:500,sriovnet2:100:500
```

3. Restart the Neutron server.

## 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 endpoints 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.

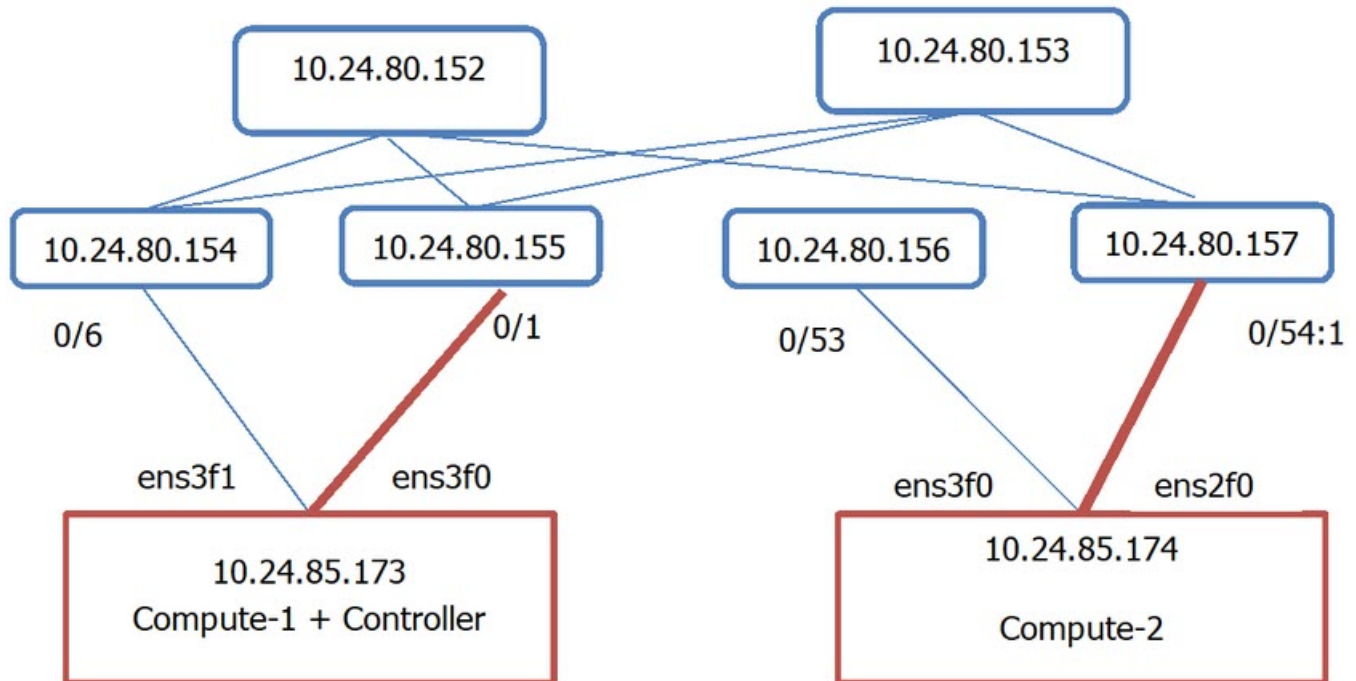
```
# openstack server delete sriov-instance-1
# openstack server delete sriov-instance-2
# openstack port delete sriov-port2
# openstack port delete sriov-port
# openstack subnet delete sriov-subnet
# openstack network delete sriov-net
```

## 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 15: 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 endpoints are provisioned for SR-IOV PF-PT and DC GW after segment creation.

## Rename Multi-Segment Network

The 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
```