Extreme Fabric Automation OpenStack Integration Guide

Version 2.4.2

9036928-01 Rev AA
April 2021

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

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

Preface

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

Text Conventions

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

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

Table 1: Notes and warnings

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

Table 2: Text

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

Table 3: Command syntax

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

Documentation and Training

Find Extreme Networks product information at the following locations:

- Current Product Documentation
- Release Notes
- Hardware and software compatibility for Extreme Networks products
- Extreme Optics Compatibility
- Other resources such as white papers, data sheets, and case studies

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

Getting Help

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

Extreme Portal

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

The Hub

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

Call GTAC

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

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

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

Subscribe to Product Announcements

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

1. Go to The Hub.
2. In the list of categories, expand the **Product Announcements** list.
3. Select a product for which you would like to receive notifications.
4. Select **Subscribe**.
5. To select additional products, return to the **Product Announcements** list and repeat steps 3 and 4.

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

Providing Feedback

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

- Content errors, or confusing or conflicting information.

- Improvements that would help you find relevant information in the document.
- Broken links or usability issues.

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

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

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

About this Document

What's New in this Document

The following table describes information added to this guide for the Extreme Fabric Automation 2.4.2 software release.

Table 4: Summary of changes

Feature	Description	Described in
Clearing the journal	New options for clearing journal entries by state: pending, failed, completed, processing, all	efa-journal clear on page 72
EFA health	Basic and advanced output shows journal status	efa-health show on page 70
L3 IP MTU	Support for the `ip-mtu` option at the network level	L3 IP MTU on page 48
Mechanism Driver and L3 Service Plug-in	The `extreme_efa_v2` and `extreme_l3_efa_v2` plug-ins now use the journaling mechanism. You can use the previous plug-ins to enable non-journaling.	Enable the Mechanism Driver and Plug-ins on page 28 Enable the Non-Journal Plug-in on page 29
L3 routers	Support for creating routers in distributed mode is disabled	L3 Routers on page 46 openstack router create on page 80
L3 flavor configuration	Configuration command examples are updated	L3 Flavors on page 47

Introduction to OpenStack Integration

Extreme Fabric Automation Overview

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

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

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

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

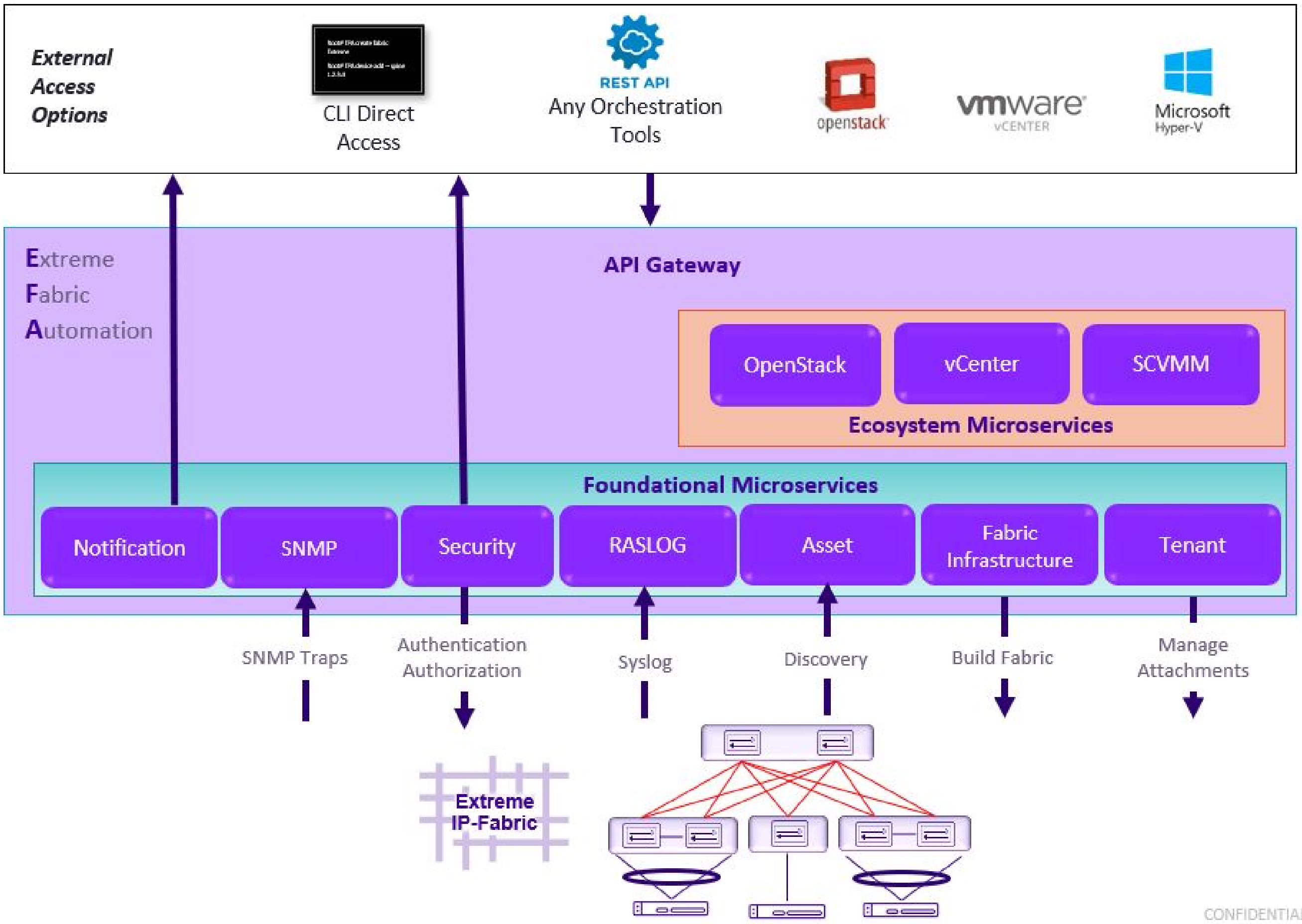


Figure 1: EFA orchestration

EFA Microservices

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

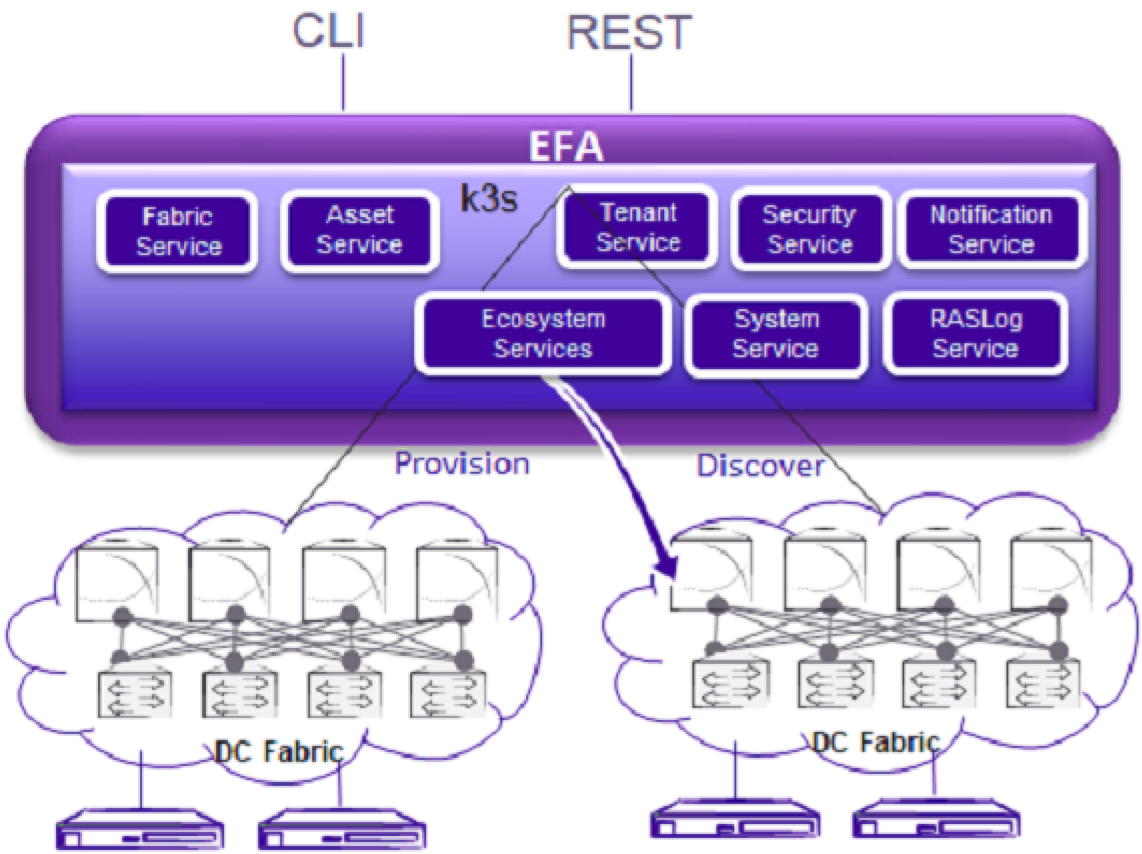


Figure 2: Microservices in the EFA architecture

Fabric Service

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

The Fabric Service features include:

- Support for small 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 RASlog Service in the Extreme Fabric Automation Administration Guide, 2.4.0.

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 EFA as SNMP Proxy in the Extreme Fabric Automation Administration Guide, 2.4.0..

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 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 provisoning for non-default physnets

    - DC-owner-based l2 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

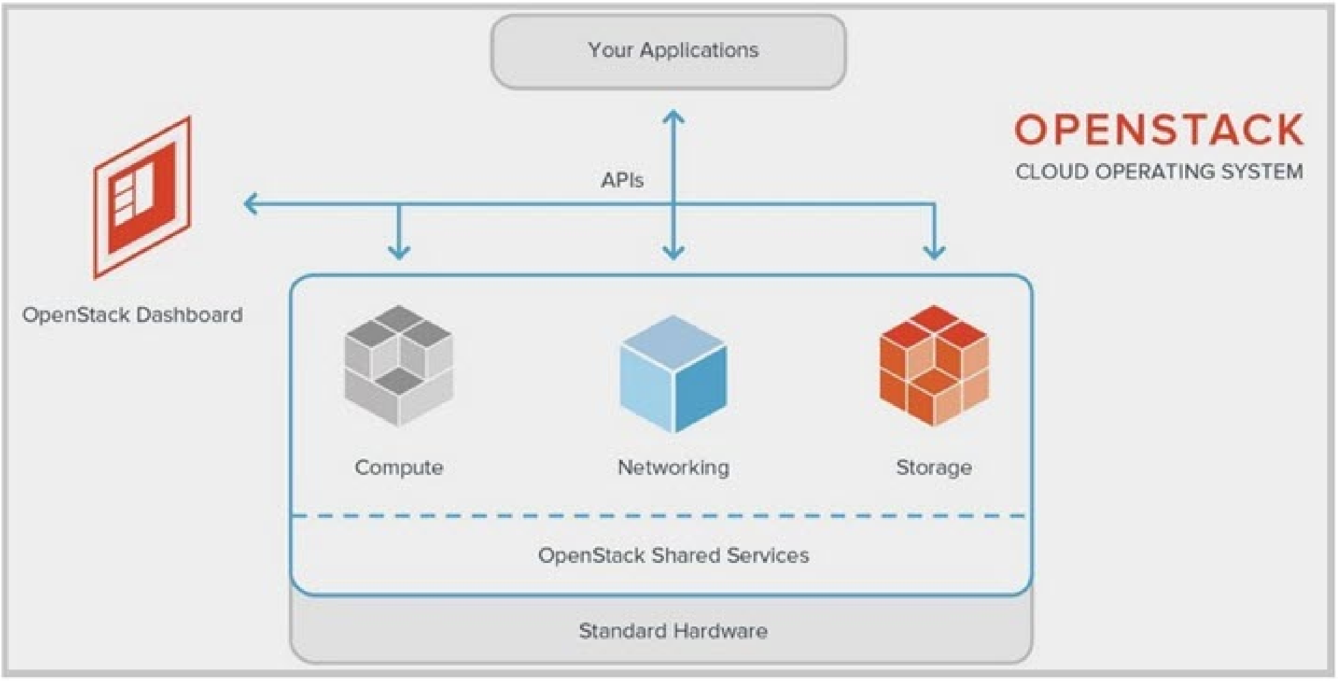
OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a Data Center. OpenStack Integration enables System administrators to manage and provision resources through APIs and web interface.

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.

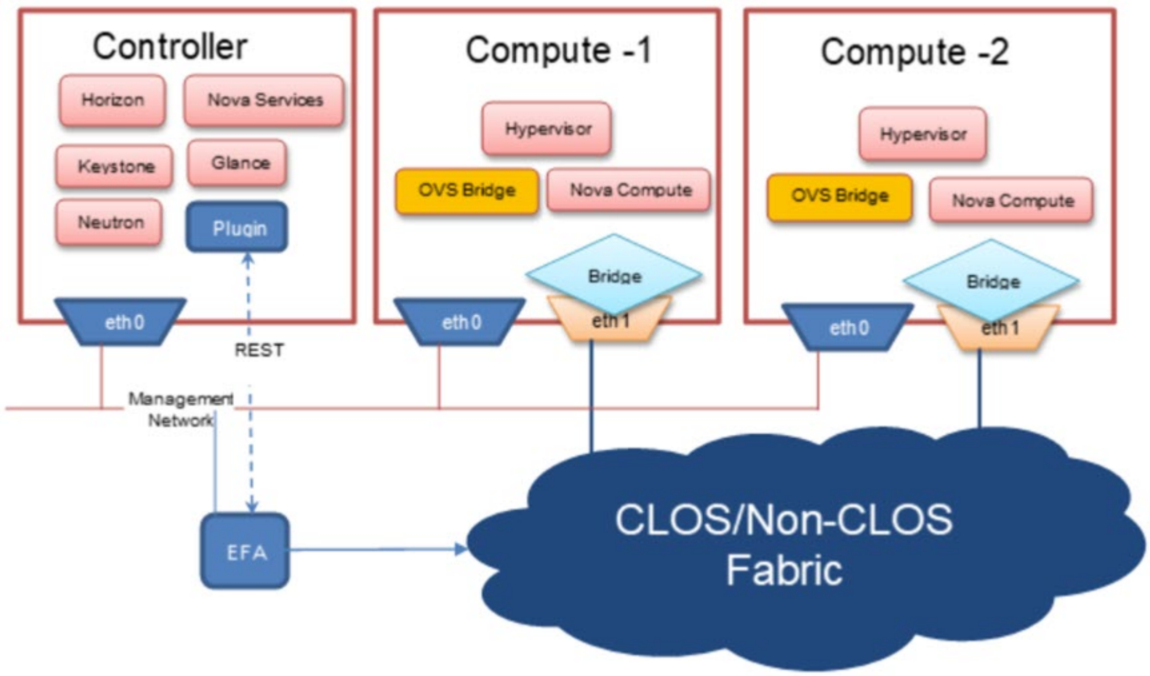


Figure 4: Overview of EFA OpenStack Network

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: • Non-Clos/Clos fabric • Clos fabrics can be 3-stage or 5-stage
Tenant Service	This service provides mechanisms to create: • Layer 2 networks • Layer 3 networks
Asset or Inventory Service	This service provides mechanisms to manage: • Physical and logical assets of the switches in the fabric • The credentials of the switches

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

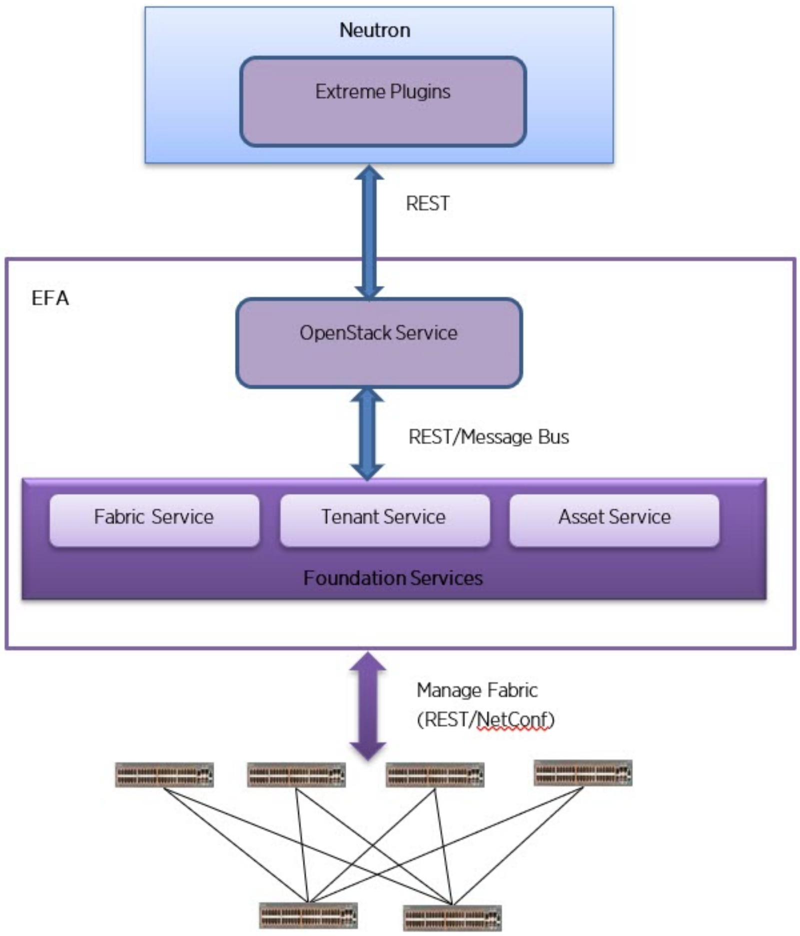


Figure 5: Overview of OpenStack ecosystem integration management

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 60.

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

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 18.

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 can be saved to .csv files for configuring them simultaneously using the file option.

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 `ml2_conf_extreme.ini` file to communicate with EFA.

```
[ml2_extreme]
efa_rest_token = <efa_api_token for VIM_1>
efa_cert_file = /root/gcla/extreme-ca-chain.crt
efa_secure_mode = True
efa_port = 443
efa_host =efa.extremenetworks.com
region_name = VIM_1
region_shared = SHARED_TENANT
#SHARED_TENANT is the name of the shared tenant created on EFA
fabric_name = CNCF
```

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

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

6. Enable Extreme EFA mechanism drivers in `ml2_conf.ini`.

```
ml2_conf.ini
[ml2]
tenant_network_types = vlan
type_drivers = vlan
mechanism_drivers = openvswitch,extreme_efa
[ml2_type_vlan]
network_vlan_ranges = physnet1:100:500 (Required vlan range)
[ovs]
bridge_mappings = physnet1:br0 (bridge used for datapath)
```

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

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

   # systemctl daemon-reload
   ```

   On DevStack installation, modify the `/etc/systemd/system/devstack@q-svc.service` file.
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 `ml2_confi_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

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 56.

> Note
> All network and server connection settings and mappings can be saved to `.csv` files for configuring them simultaneously using the `startup` file option in the `ml2_conf_extreme.ini`.
> 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 EFA operations are updated accordingly.

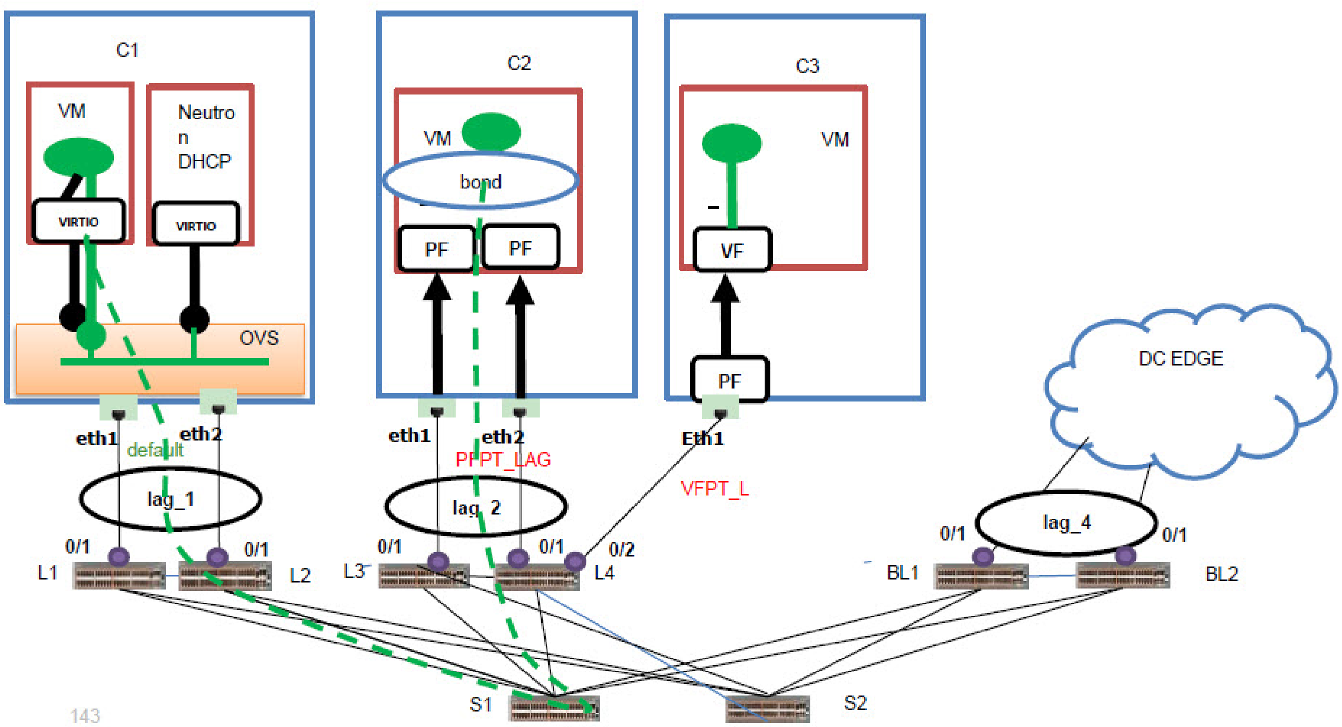


Figure 6: Topology setup in Neutron

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)

Node	IP Address or Description
C3	10.24.51.117 (OpenStack117)
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 region_name 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 region_sharedin ML2 configurations.
EFA REST TOKEN	Used as API key efa_rest_token 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	
`efa tenant create --name VIM_1 --vlan-range 2-4090 --l2-vni-range 1-5000 --port 10.24.14.133.[0/1], 10.24.14.134.[0/1],10.24.14.135.[0/1], 10.24.14.136.[0/1-2]`	Create Tenant VIM_1
`efa tenant po create --name lag_1 --tenant VIM_1 --port 10.24.14.133.[0/1], 10.24.14.134.[0/1] --speed 10Gbps --negotiation active`	Lag on C1

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

EFA CLI Command	
`efa tenant po create --name lag_2 --tenant` **`VIM_1`** `--port 10.24.14.135.[0/1], 10.24.14.136.[0/1] --speed 10Gbps --negotiation active`	Lag on C2
`efa tenant create --name` **`SHARED_TENANT`** `--port 10.24.14.191.[0/1], 10.24.14.192.[0/1] –type` **`shared`**	Shared tenant on border leaf
`efa tenant po create --name lag_4 --tenant` **`SHARED_TENANT`** `--port 10.24.14.133.[0/1], 10.24.14.134.[0/1] --speed 10Gbps --negotiation active`	Border leaf lag created on shared tenant
`efa auth client register --name` **`VIM_1`** `--type openstack`	Register VIM_1 tenant for OpenStack access
`efa auth apikey generate --client-id <client_id>`	Use the Client ID generated in client register for API key generation Use the generated API Key as a the efa_rest_token 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
`efa_rest_token`	API token available using `efa client` commands
`efa_cert_file`	The location of the CA Cert Chain for EFA

Token	Description
efa_secure_mode	True (for secure connections)
efa_port	Port running the HTTP service
efa_host	IP address of EFA service: efa.extremenetworks.com. This setting must be resolved to the EFA IP address through DNS or /etc/hosts.
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

Enable the Mechanism Driver and Plug-ins

Make configurations on the Neutron controller for starting the Extreme EFA Mechanism Driver and L3 Service Plug-in, a journal-based plug-in.

About This Task

The extreme_efa_v2 and extreme_l3_efa_v2 plug-ins use a journaling mechanism,

Procedure

1. ML2 Configuration: Set the mechanism driver to extreme_efa_v2, along with openvswitch. In the /etc/neutron/plugins/ml2/ml2_conf.ini file, add the following lines:

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

   The configurations in the following table also affect the working of the journaling mechanism, which is configured in /etc/neutron/plugins/ml2/ml2_conf.ini.

Parameter	Description	Default
sync_timeout	Sync thread timeout in seconds or fraction.	10
retry_count	Number of times to retry a row before failing.	5
maintenance_interval	Journal maintenance operations interval in seconds.	300
completed_rows_retention	Time to keep completed rows (in seconds). For performance reasons, do not change this from the default value (0) which indicates completed rows are not kept. This value is checked every maintenance_interval by the cleanup thread. To keep completed rows indefinitely, set the value to -1.	0
journal_rows_threshold	Maximum number of journal rows allowed. If the number of rows exceeds this threshold, then the efa_health tool shows the status as down.	1000

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

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

3. L3 Service Plug-in Configuration: Activate the L3 Service plug-in by adding the following lines in the `/etc/neutron/neutron.conf` file:

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

   Note

   `extreme_l3_efa_v2` is an L3 Service Plug-in and is a replacement for the default L3 Service Plug-in (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:extremev2:networking_extreme.l3.l3_flavor_v2
.ExtremeL3ServiceProviderv2:default
```

   Flavors require the default L3 Service Plug-in (called `router`). In other words, flavors can be used only when the service plug-in is `router`.

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

   For more information, see L3 Flavors on page 47.

Enable the Non-Journal Plug-in

This is an alternate Neutron configuration for setting up Layer 2 and Layer 3 without using journaling.

Complete the following changes in the configuration files to enable the non-journal plug-in.

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` along with `openvswitch`.

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

L3 Service Plug-in Configuration

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

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

Note

`extreme_l3_efa` is an L3 Service Plug-in and is a replacement for the default L3 Service Plug-in (called `router`). Only one of them should be used.

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. The router creation command must specify the flavor profile for whichever router is being created. For more information, see Commands for L3 Flavor Creation and L3 Flavors on page 47.

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

Note

Flavors require the default L3 Service Plug-in (called `router`). In other words, flavors can be used only when the service plug-in is `router`:

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

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 79 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.

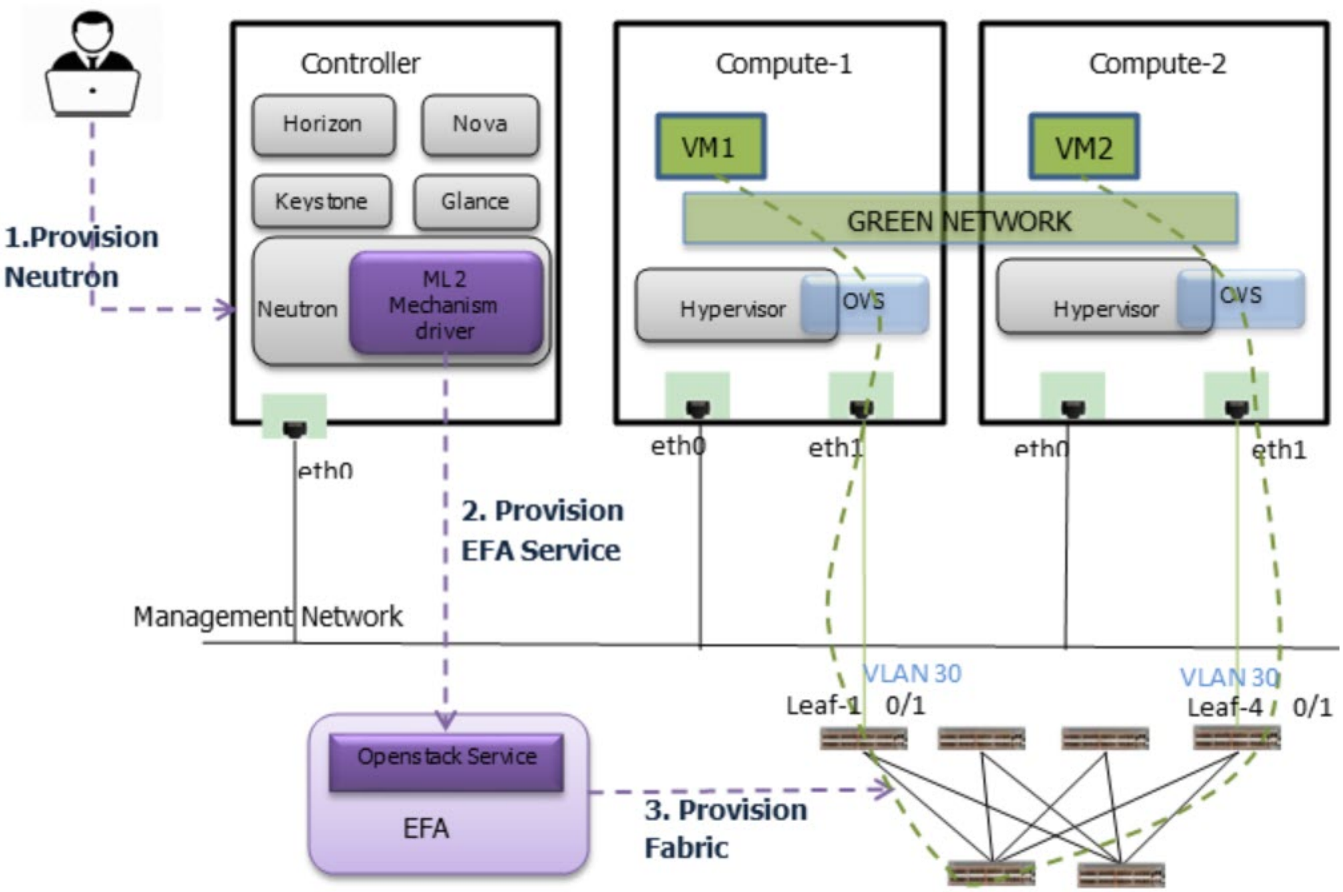


Figure 7: Overview of ML2 Mechanism Driver within Neutron

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: • The Neutron port is bound to a host when VM is launched. • The Neutron port is bound to a host (Neutron controller) when DHCP starts.
default (device-owner=dc-edge –host=<DCGW>)	VLAN provisioning for default physnet is done during port creation based on additional parameters passed during port create call.
non-default (PFPT_L, EXT1)	VLAN provisioning is done during network creation or segment creation of single segment or multiple segment.

The endpoints are deprovisoned 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.

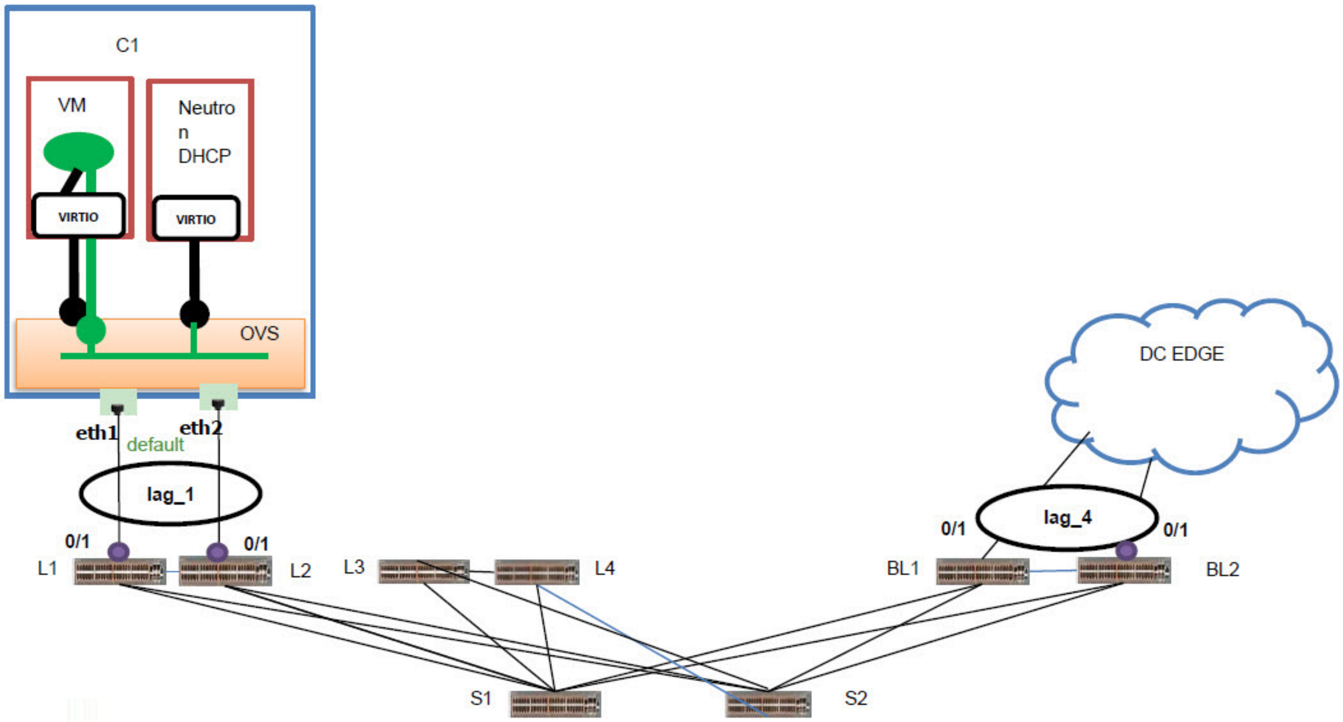


Figure 8: Overview of single segment Virtio VM network

Table 9: Commands and impacts

Command	EFA Impact
`openstack network create --provider-network-type vlan --provider-physical-network` **`default`** `--provider-segment` **`3320`** `ss1network1`	EPG Created for ss1network Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d CTAG = 3320 Note - 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d is the neutron UUID allocated for the EPG.
`openstack subnet create ss1subnet1 --network ss1network1 --subnet-range 10.1.1.0/24` `openstack subnet create ss1subnet1ipv6 --network ss1network1 --ip-version 6 --ipv6-address-mode=dhcpv6-stateful --subnet-range fd00:10:0:57::1000/64`	DHCP EndPoints Created on EPG corresponding to ss1network1. VLAN Provisioned. EPG Updated Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d Port = lag_1 (added)

`openstack port create ss1VirtIoTrunkPort1 --network ss1network1`
`openstack network trunk create --parent-port ss1VirtIoTrunkPort1 ss1VirtIoTrunk1`
`openstack port create ss1VirtIoSubport1 --network ss1network1`	

Table 9: Commands and impacts (continued)

Command	EFA Impact
`openstack network create --provider-network-type vlan --provider-physical-network default --provider-segment 3321 ss1network2`	EPG Created for ss1network2 *Name = 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d* *CTAG = 3321*
`openstack subnet create ss1subnet2 --network ss1network2 --subnet-range 11.1.1.0/24` `openstack subnet create ss1subnet2ipv6 --network ss1network2 --ip-version 6 --ipv6-address-mode=dhcpv6-stateful --subnet-range fd00:11:0:57::1000/64`	DHCP EndPoints Created on EPG corresponding to ss1network2. VLAN Provisioned EPG Updated Name = 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d Port = lag_1 (added)

`openstack port create ss1VirtIoTrunkPort2 --network ss1network2`
`openstack network trunk create --parent-port ss1VirtIoTrunkPort2 ss1VirtIoTrunk2`
`openstack network trunk set --subport port=ss1VirtIoSubport1,segmentation-type=vlan,segmentation-id=3801 ss1VirtIoTrunk2`	

`openstack server create --flavor m1.large --image ubuntu --port $(neutron port-list | grep -w 'ss1VirtIoTrunkPort1' | awk '{print $2}') ss1VirtIoVM1 --availability-zone nova:Openstack116`	Endpoint corresponding to 'ss1VirtIoTrunkPort1' added to EPG(ss1nework1) VLAN Provisioned EPG Updated Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d Port = lag_1 (added) - no impact already added
`openstack server create --flavor m1.large --image ubuntu --port $(neutron port-list | grep -w 'ss1VirtIoTrunkPort2' | awk '{print $2}') ss1VirtIoVM2 --availability-zone nova:Openstack117`	Endpoint corresponding to 'ss1VirtIoTrunkPort2' added to EPG(ss1nework2) VLAN Provisioned EPG Updated Name = 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d Port = lag_1 (added) - no impact already added Endpoint corresponding to ss1VirtIoSubport1 added to EPG(ss1nework1) VLAN Provisioned EPG Updated Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d Port = lag_1 (added) - no impact already added
`openstack port create ss2DcGwPort1 --device-owner network:dc_edge --host` **`DCGW-1`** `--network ss1network1`	EndPoint corresponding to 'host DCGW-1' added to EPG (ss1network1) VLAN Provisioned EPG Updated *Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d* *Port = lag_1, lag_4 (added)*
`openstack port create ss2DcGwPort1 --device-owner network:dc_edge --host` **`DCGW-1`** `--network ss1network2`	EndPoint corresponding to 'host DCGW-1' added to EPG (ss1network2) VLAN Provisioned EPG Updated Name = 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d Port = lag_1, lag_4 (added)

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.

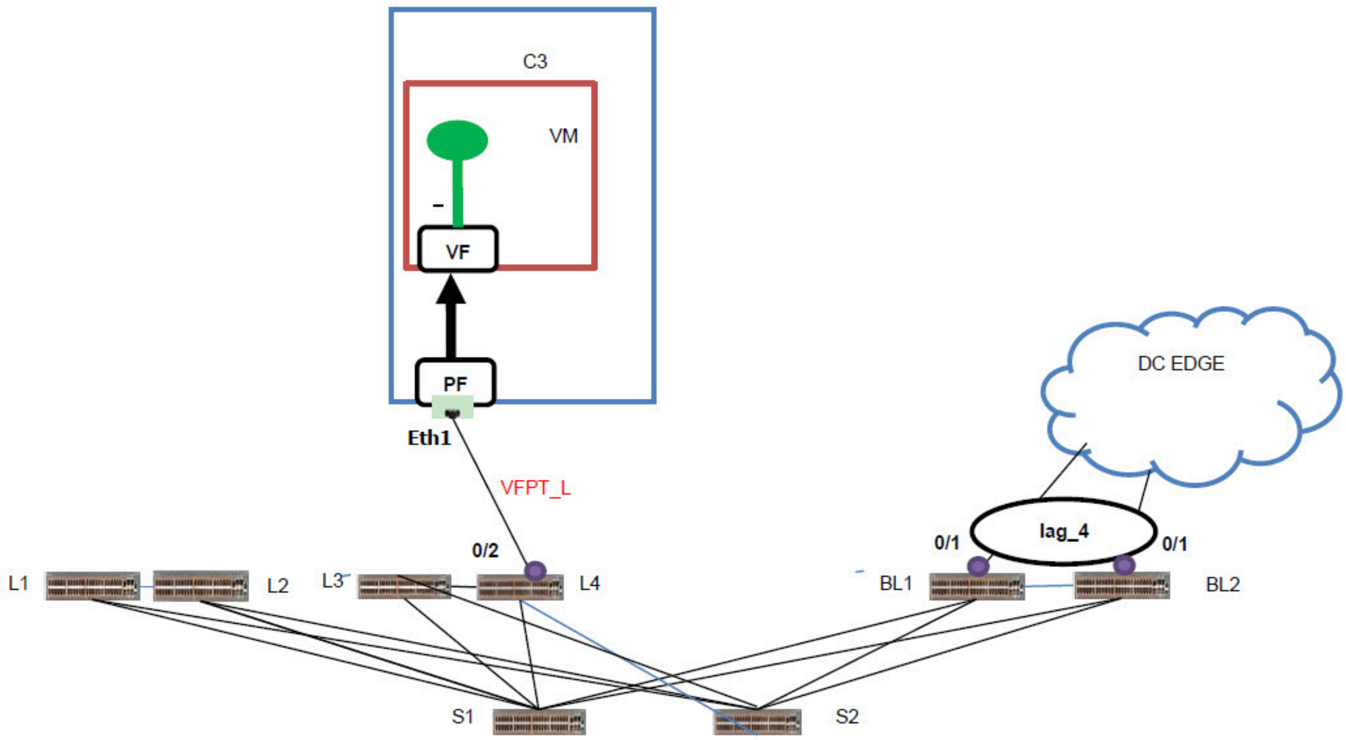


Figure 9: Overview of single segment VFPT VM network with DC Edge

Table 10: Commands and impacts

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

Table 10: Commands and impacts (continued)

Command	Impact on EFA
openstack subnet create ss8flatleftsubnet --network ss9_vfpt_flat_left --no-dhcp -- subnet-range 107.1.1.0/24	
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	
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	
openstack network create -- provider-network-type vlan -- provider-physical-network VFPT_L -- provider-segment **3390ss9network**	EPG Created for ss9network *Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d* *CTAG =* 3390 Note - *74cbf489-f3d9-41c7-bbb2-6cb7df33da6d* *Is the neutron UUID allocated for the EPG*
openstack subnet create ss9subnet1 --network ss9network1 --no-dhcp -- subnet-range 90.1.1.0/24 openstack subnet create ss9subnet1ipv6 --network ss8network1 --ip-version 6 --no- dhcp --subnet-range fd00:90:0:57::1000/64	No impact as –no-dhcp option is used
openstack port create ss9SriovSubPort1 --network ss9network1 --mac-address <same- mac-as-ss9_port_vfpt_left1> --vnic- type direct --fixed-ip subnet=ss9subnet1,ip- address=90.1.1.10 --fixed-ip subnet=ss9subnet1ipv6,ip- address=fd00:90:0:57::10	

Table 10: Commands and impacts (continued)

Command	Impact on EFA
`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 <same-deviceid-as-ss9_port_vfpt_left1> --binding-profile pci_slot=<same-as-slot-of-ss9_port_vfpt_left1> --binding-profile pci_vendor_info=<same-as-vendor-of-ss9_port_vfpt_left1> --binding-profile physical_network=<same-as-physnet-of-ss9_port_vfpt_left1>`	Endpoint corresponding to ss9SriovSubPort1 added to EPG(ss9network1) VLAN Provisioned EPG Updated *Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d* *Port = L4[0/2] (added)*
`openstack port create ss9DcGwPort --network ss9network1 --device-owner network:dc_edge --host` **`DCGW-1`**	EndPoint corresponding to 'host DCGW-1' added to EPG (ss9network1) VLAN Provisioned EPG Updated *Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d* *Port = L4[0/2]. lag_4 (added)*

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.

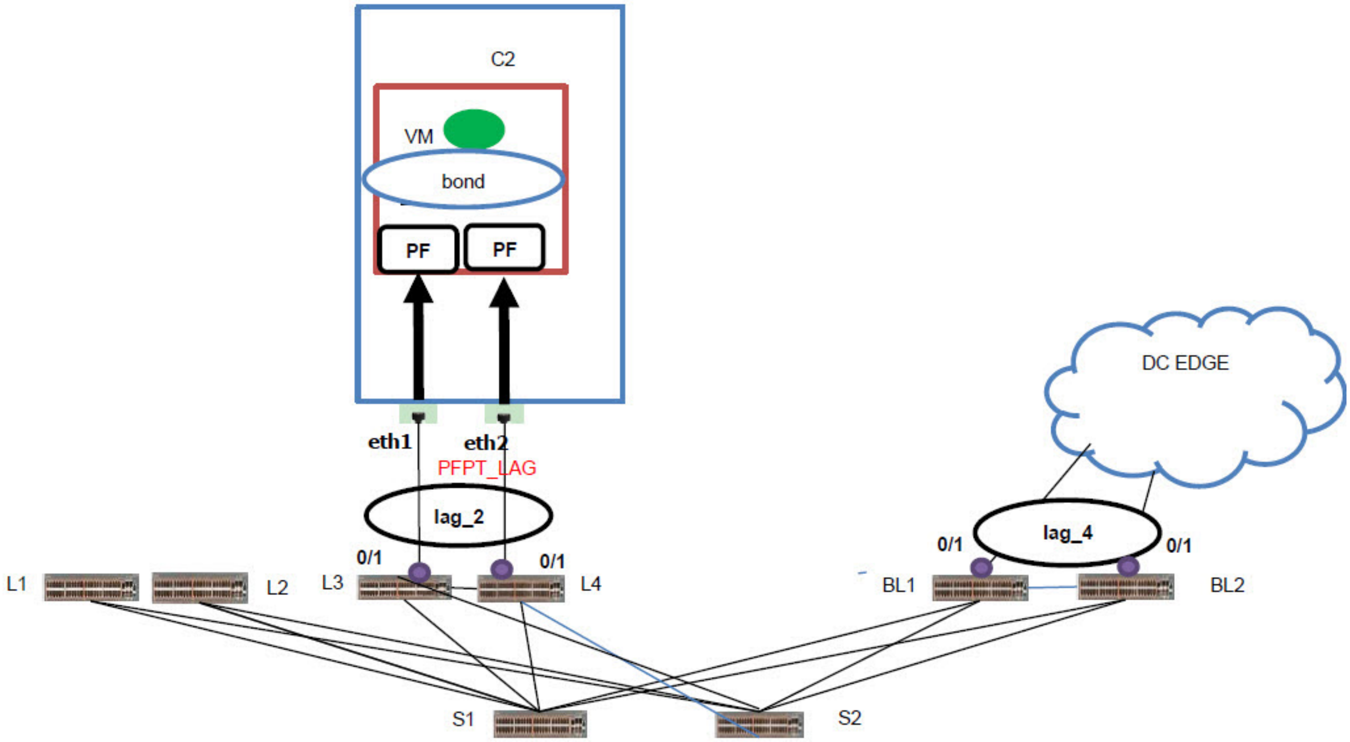


Figure 10: Overview of single segment PFPT VM network with DC Edge

Table 11: Commands and impacts

Command	EFA Impact

`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`
	No impact as operations are on FLAT network type

`openstack network create --provider-network-type vlan --provider-physical-network PFPT_LAG --provider-segment` **`3370ss7network1`**	EPG Created for ss7network1 *Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d* *CTAG =* 3370 Note *- 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d Is the neutron UUID allocated for the EPG*
`openstack subnet create ss7subnet1 --network ss7network1 --no-dhcp --subnet-range 70.1.1.0/24` `openstack subnet create ss7subnet1ipv6 --network ss7network1 --ip-version 6 --no-dhcp --subnet-range fd00:70:0:57::1000/64`	No impact as –no-dhcp option is used.

Table 11: Commands and impacts (continued)

Command	EFA Impact
`openstack port create ss7PFPTSubPort1 --network ss7network1 --mac-address <same-mac-as-ss7_port_pfpt_left> --vnic-type direct-physical --fixed-ip subnet=ss8subnet1,ip-address=70.1.1.10 --fixed-ip subnet=ss7subnet1ipv6,ip-address=fd00:70:0:57::10`	
`openstack network trunk set --subport port=ss7PFPTSubPort1,segmentation-type=vlan,segmentation-id=3370`	Endpoint corresponding to ss7PFPTSubPort1 added to EPG(ss7network1) VLAN Provisioned EPG Updated *Name = 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d* *Port = lag_2 (added)*
`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`	EndPoint corresponding to 'host DCGW-1' added to EPG (ss9network1) VLAN Provisioned EPG Updated *Name = 84cbf489-f3d9-41c7-bbb2-6cb7df33da6d* *Port = lag_2 ,lag_4 (added)*

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.

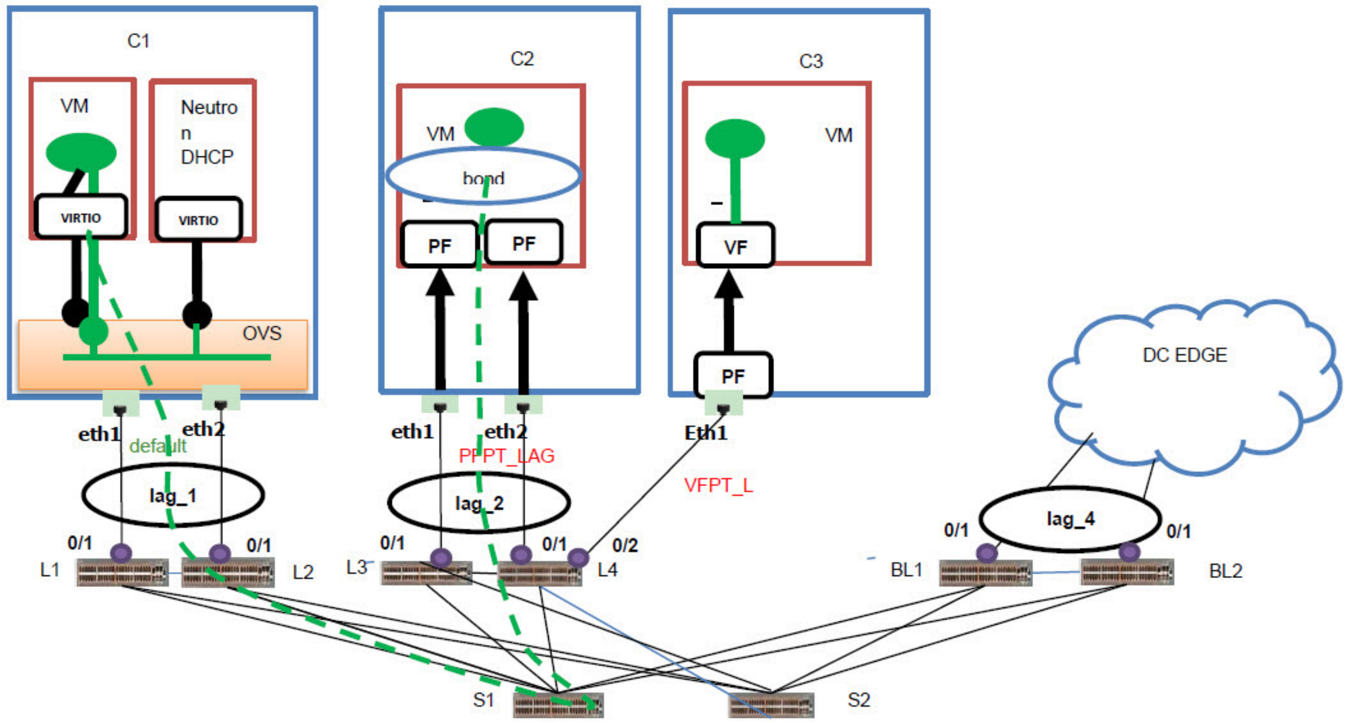


Figure 11: Overview of Virtio VM, PFPT, and VFPT segment network with DC Edge

Table 12:

Command	EFA Impact
`openstack network create --provider-network-type vlan --provider-physical-network VFPT_L --provider-segment 3710 ms10Network1`	EPG Created for ms10Network1 *Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d* *CTAG =* 3710 *Note - 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d* *Is the neutron UUID allocated for the EPG* *Port = L4[0/2] added*
`openstack network segment create --network-type vlan --physical-network EXT1 --segment 3710 --network ms10Network1 ms10Network1DcgwSegment`	Endpoint corresponding EXT1 added to EPG(ms10Network1) VLAN Provisioned EPG Updated *Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d* *Port = L4[0/2] , lag_4 (added)*
`openstack network segment create --network-type vlan --physical-network PFPT_LAG --segment 3710 --network ms10Network1 ms10NetworkPFPTSegment`	Endpoint corresponding EXT1 added to EPG(ms10Network1) VLAN Provisioned EPG Updated *Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d* *Port = L4[0/2] , lag_4 , lag_2 (added)*
`openstack network segment create --network-type vlan --physical-network default --segment 3710 --network ms10Network1 ms10Network1DefaultSegment`	

Table 12: (continued)

Command	EFA Impact
`openstack subnet create ms10subnet1 --network ms10Network1 --subnet-range 130.1.1.0/24` `openstack subnet create ms10subnet1ipv6 --network ms10Network1 --ip-version 6 --ipv6-address-mode=dhcpv6-stateful --subnet-range fd00:0130:0:57::1000/64`	

`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`	

`openstack server create --flavor myhuge --image ubuntu --port $(neutron port-list | grep -w 'ms10VirtIoPort1' | awk '{print $2}') ms10VirtIoVM1 --availability-zone nova:compute-0-5.domain.tld`	Endpoint corresponding default added to EPG(ms10Network1) VLAN Provisioned EPG Updated *Name = 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d* *Port = L4[0/2],lag_4,*lag_2 *(added)*
`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`	

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 L3 Routers on page 46.

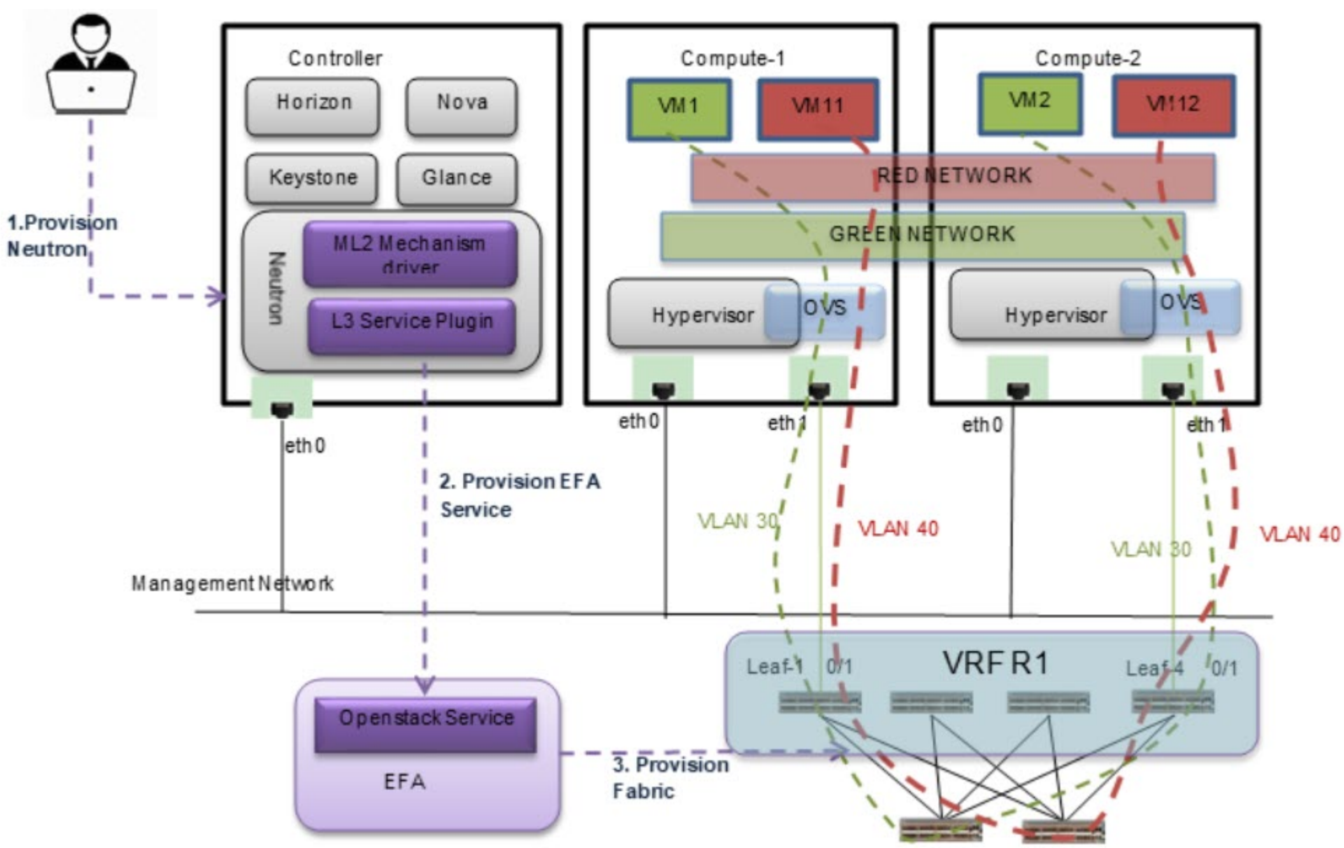


Figure 12: L3 Service plug-in

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_SUBNETas 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 provisoning

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

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.

L3 Routers

L3 routers can be configured in centralized mode only.

In centralized mode, routing is configured only on the border leaf pairs. In contrast, for distributed mode, routing is configured on the corresponding leaf nodes where the endpoints reside.

> Note
>
> The Extreme L3 service plug-in does not support routers configured in distributed mode, although the feature was supported in a previous release. The functionality to configure distributed mode is disabled.

When you create a router, the default mode is centralized. You can create a router with or without the **--centralized** option and achieve the same result. For example, each of these commands creates a router called R2 in centralized mode:

- `openstack router create R2`
- `openstack router create R2 --centralized`

The L3 service plug-in passes this information to EFA. Then, the EFA Tenant Service configures VRF or routing on border leaf devices.

Note
The OpenStack integration works with only one pair of border leaf devices. The centralized routing instance is created on this pair. Ensure that you add only one border leaf pair during fabric creation.

L3 Flavors

L3 flavors allow multiple backends in one Neutron deployment.

L3 flavors allow multiple backends in one Neutron deployment, each with its own logical network topology, completely separate from each other. This architecture allows incremental migrations from one backend to another.

Create a flavor profile <flavorprofileid>

```
openstack network flavor profile create --driver
networking_extreme.l3.l3_flavor_v2.ExtremeL3ServiceProviderv2
```

Create a network flavor

```
openstack network flavor create --service-type L3_ROUTER_NAT
l3_extreme_v2
```

Add a profile to the flavor

```
openstack network flavor add profile l3_extreme_v2 <flavorprofileid>
```

Create a Neutron router using the create flavor

```
neutron router-create router1 --flavor l3_extreme_v2
```

Create a flavor profile with non-journaling support

```
openstack network flavor profile create --driver
networking_extreme.l3.l3_flavor.ExtremeL3ServiceProvider
```

For changes required in the Neutron configuration file, see Configure Neutron to Connect to EFA on page 27 and Enable the Non-Journal Plug-in 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.

You can use the following command to create a centralized router along with the flavor: `neutron router-create router1 --flavor l3_extreme_v2 –distributed=False`.

Note
- The **`openstack router create`** command does not support the creation of routers using flavors.
- The functionality to configure distributed mode is disabled. For more information, see L3 Routers on page 46.

L3 IP MTU

Neutron provides multiple ways to specify the maximum transmission unit (MTU) for networks.

You can provide a global value in the `neutron.conf` file using the `global_physnet_mtu` value. This value applies to all networks. For example:

```
/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. For example:

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

The MTU value for each network is captured by Extreme plug-ins and passed to EFA. EFA then enforces the value on the endpoint group as part of its `ipv6-nd-mtu` and `ip-mtu` options. This value is per Ctag and is enforced on the VE interface corresponding to the Ctag. EFA configures the `ipv6-nd-mtu` and `ip-mtu` values on the specific VE interface on the device.

The value is applied to the SLX device when the 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.

```
(device)# show running-config interface Ve 3

interface Ve 3
 vrf forwarding 274e9461081849f8867b77a112f6c510
 ip anycast-address 61.1.1.1/24
 ip mtu 9000
 ipv6 anycast-address fd00:61:0:57::1/64
 ipv6 nd managed-config-flag
 ipv6 nd other-config-flag
 ipv6 nd mtu 9000
 ipv6 nd prefix fd00:61:0:57::/64 2592000 604800 no-autoconfig
 no shutdown
```

Note
SLX supports IP MTU values in the range from 1280 through 9194 bytes. Creation of the router-interface fails if you try to configure an MTU value outside of this range.

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 journal maintains a list of operations that occurred on Neutron and should be mirrored to the controller.

The journal is processed mainly by a journaling thread that runs periodically and checks whether the journal table has any entries that need to be processed. The thread is triggered in the post-commit phase of the operation (when applicable.

For example, after the ML2 plug-in stores the 'create network again' process in the Neutron database, 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 (when applicable) so that when a commit failure occurs, the journal entry and the original operation are deleted.

For more information, see https://docs.openstack.org/networking-odl/latest/contributor/drivers_architecture.html.

> Note
> Journaling is available for the ML2 plug-in and Layer 3 plug-in.

Several commands are available for journaling. For more information, see EFA OpenStack Service Command Reference on page 60.

- **`efa-journal list`**
- **`efa-journal reset`**
- **`efa-journal clear`**

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 show`** command is provided as part of the installation package.

You can use the command to verify the following: connectivity with EFA, database write operations, OpenStack journal thread status, 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 accounts for this additional VRF.

In the command output, the EFA Health entry represents EFA reachability and EFA database health.

In the output, the EFA High Availability Status entry shows DOWN even if only one node is DOWN. However, even with one node down, EFA Health remains UP because EFA is still operational.

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

```
$ efa-health show

EFA Host                         : efa.extremenetworks.com
EFA Health                       : UP
EFA Health Failure Reason        :
EFA High Availability Status     : UP
EFA Primary                      : UP
EFA Secondary                    : UP

EFA Neutron Journal Size within Threshold    : NO
EFA Neutron Journal Size Threshold Setting   : 5
EFA Neutron Current Journal Size             : 7
EFA Neutron Journal Maintenance Status       : UP
```

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

```
$ efa-health show --advanced

EFA Host                         : efa.extremenetworks.com
EFA Health                       : UP
EFA Health Failure Reason        :
EFA High Availability Status     : UP
EFA Primary                      : UP
EFA Secondary                    : UP

EFA Neutron Journal Size within Threshold   : NO
EFA Neutron Journal Size Threshold Setting  : 5
EFA Neutron Current Journal Size            : 7
EFA Neutron Journal Maintenance Status      : UP

EFA Cluster Status               : 18/28 Pods Running

  rabbitmq-0                                       : Running
  efa-api-docs-67b8c76ddb-2hg7s                    : Running
  gosnmp-service-546d76b6f9-7wb6t                  : Running
  goopenstack-service-5b687f9f8c-srlxx             : Running
  gosystem-service-76bbbc6d-jzkng                 : Running
  goauth-service-75c88f4986-vqppm                  : Running
  goraslog-service-7467fb7759-jm28m                : Running
  gotenant-service-5f8bc9f458-zbz2l                : Running
  gohyperv-service-b546d647f-ws7t8                 : Running
  gorbac-service-b546dbdd5-v9dml                   : Running
  gofabric-service-65b6f7d4cf-rfq6f                : Running
  gonotification-service-5fb74fc959-2x9lt          : Running
  goinventory-service-79d6545c69-s66tx             : Running
  govcenter-service-7c6cb944dd-2d8lt               : Running
  goinventory-service-79d6545c69-gxtn6             : Init:0/2
  efa-api-docs-67b8c76ddb-n8zfp                    : Running
  gosnmp-service-546d76b6f9-fv8tz                  : Init:0/1
  goraslog-service-7467fb7759-rvppq                : Running
  gorbac-service-b546dbdd5-s9qxh                   : Init:1/2
  gosystem-service-76bbbc6d-x8x8g                  : Running
  govcenter-service-7c6cb944dd-rx7gd               : Init:0/2
  gohyperv-service-b546d647f-zjq97                 : Init:0/2
  goauth-service-75c88f4986-9fbn7                  : Init:0/2
  gofabric-service-65b6f7d4cf-q7qns                : Init:0/2
  gonotification-service-5fb74fc959-rr724          : Init:0/2
  gotenant-service-5f8bc9f458-n7gm8                : Init:0/3
  goopenstack-service-5b687f9f8c-bfssf             : Init:0/2
  rabbitmq-1                                       : Running
```

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.

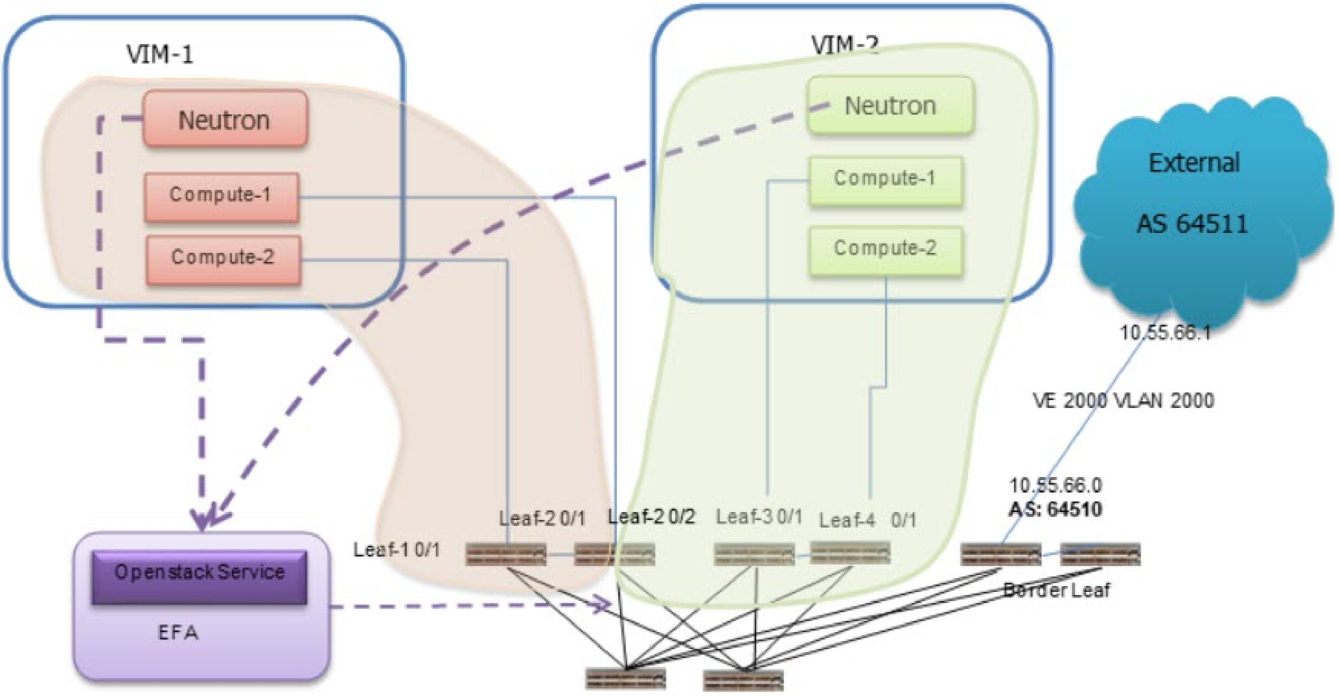


Figure 13: Multiple VIM or VPOD instances

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

The following example shows multiple VIM/VPOD instances.

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

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

# efa tenant show
+----------------+------------+------------+------------+-----------+-----------
```

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

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               | m1.tiny                              |
| id                   | d1df1b5a-70c4-4fed-98b7-423362f2c47c |
| name                 | vm1                                  |
| status               | ACTIVE                               |
| ...                  | ...                                  |
+----------------------+--------------------------------------+
```

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

```
# openstack compute service list

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

# openstack host show HostC

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

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

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

Add Certificate to OpenStack Controller

Procedure

1. Copy certificate from EFA to OpenStack Controller.
   ```
   administrator@osController-55:$ su
   Password:
   root@osController-55:# cd /usr/local/share/ca-certificates
   root@osController-55:#
   root@osController-55:/usr/local/share/ca-certificates# scp root@10.21.88.130://usr/
   local/share/ca-certificates/extreme-ca-chain.crt .
   root@10.21.88.130's password:
   extreme-ca-chain.crt
   100% 4488     4.4KB/s   00:00
   root@osController-55:/usr/local/share/ca-certificates#
   ```
2. Restart Neutron to apply the token and certificate.
   ```
   # sudo service neutron-* restart
   ```

EFA OpenStack Service Command Reference

efa openstack debug

Displays OpenStack debug information.

Syntax

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

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

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 the following: connectivity with EFA, database write operations, OpenStack journal thread status, and the status of the EFA Kubernetes cluster.

Syntax

efa-health show [**--advanced**]

Parameters

--advanced

Provides detailed status about EFA pods and the cluster. Omit this parameter to return basic status information.

Usage Guidelines

In the output, the EFA Health entry represents EFA reachability and EFA database health.

In the output, the EFA High Availability Status entry shows DOWN even if only one node is DOWN. However, even with one node down, EFA Health remains UP because EFA is still operational.

Examples

This example shows basic output for the command.

```
$ efa-health show

EFA Host                        : efa.extremenetworks.com
EFA Health                      : UP
EFA Health Failure Reason       :
EFA High Availability Status    : UP
EFA Primary                     : UP
EFA Secondary                   : UP

EFA Neutron Journal Size within Threshold   : NO
EFA Neutron Journal Size Threshold Setting  : 5
EFA Neutron Current Journal Size            : 7
EFA Neutron Journal Maintenance Status      : UP
```

This example shows advanced output for the command.

```
$ efa-health show --advanced

EFA Host                        : efa.extremenetworks.com
EFA Health                      : UP
EFA Health Failure Reason       :
EFA High Availability Status    : UP
EFA Primary                     : UP
EFA Secondary                   : UP

EFA Neutron Journal Size within Threshold   : NO
EFA Neutron Journal Size Threshold Setting  : 5
EFA Neutron Current Journal Size            : 7
EFA Neutron Journal Maintenance Status      : UP

EFA Cluster Status              : 18/28 Pods Running
```

```
rabbitmq-0                                          : Running
efa-api-docs-67b8c76ddb-2hg7s                       : Running
gosnmp-service-546d76b6f9-7wb6t                     : Running
goopenstack-service-5b687f9f8c-srlxx                : Running
gosystem-service-76bbbc6d-jzkng                     : Running
goauth-service-75c88f4986-vqppm                     : Running
goraslog-service-7467fb7759-jm28m                   : Running
gotenant-service-5f8bc9f458-zbz2l                   : Running
gohyperv-service-b546d647f-ws7t8                    : Running
gorbac-service-b546dbdd5-v9dml                      : Running
gofabric-service-65b6f7d4cf-rfq6f                   : Running
gonotification-service-5fb74fc959-2x9lt             : Running
goinventory-service-79d6545c69-s66tx                : Running
govcenter-service-7c6cb944dd-2d8lt                  : Running
goinventory-service-79d6545c69-gxtn6                : Init:0/2
efa-api-docs-67b8c76ddb-n8zfp                       : Running
gosnmp-service-546d76b6f9-fv8tz                     : Init:0/1
goraslog-service-7467fb7759-rvppq                   : Running
gorbac-service-b546dbdd5-s9qxh                      : Init:1/2
gosystem-service-76bbbc6d-x8x8g                     : Running
govcenter-service-7c6cb944dd-rx7gd                  : Init:0/2
gohyperv-service-b546d647f-zjq97                    : Init:0/2
goauth-service-75c88f4986-9fbn7                     : Init:0/2
gofabric-service-65b6f7d4cf-q7qns                   : Init:0/2
gonotification-service-5fb74fc959-rr724             : Init:0/2
gotenant-service-5f8bc9f458-n7gm8                   : Init:0/3
goopenstack-service-5b687f9f8c-bfssf                : Init:0/2
rabbitmq-1                                          : Running
```

efa-journal clear

Removes all or specified entries from the journal.

Syntax

efa-journal clear [**pending | failed | completed | processing | all**]

Parameters

pending
: Clears all entries in Pending state.

failed
: Clears all entries in Failed state.

completed
: Clears all entries in Completed state.

processing
: Clears all entries in Processing state.

all
: Clears all entries.

Usage Guidelines

Important
Stop the Neutron service before clearing entries using the **all**, **pending**, or **processing** parameters. Otherwise, the processing thread can repopulate the entries in the journal.

Examples

The following example clears all entries in Failed state.

```
$ efa-journal clear failed
```

The following example clears all entries. Note the warning to stop the Neutron service.

```
$ efa-journal clear all
Warning!!! Clearing states pending/processing/all should be executed after shutting
down neutron service. Otherwise the entries can be re-populated by the processing thread.
```

efa-journal list

Lists all journal entries and their status.

efa-journal list

This example lists all journal entries and their statuses.

```
+-----+--------------------------------------+-------------+-----------------------------
|  Id |             Journal UUID             | Object Type |             Object
```

```
UUID              |
+-----+--------------------------------------+-------------
+--------------------------------------+
| 301 | 3ebf2416-366b-4c11-9a2e-8c058c219555 |   network   | 108755af-c7fd-4590-
ac00-5359f76b7ba4 |
| 302 | 78884b55-9dfb-4dbb-98ef-1916a5f932f8 |   network   | 78b7b7e8-ea64-420e-
afd8-5349485afe22 |
| 303 | ef3b4637-a899-49e0-9f7d-28dda671685f |   network   | 477714ac-83d1-4ce5-
a56d-88d8bc4b9536 |
| 304 | 7d662b57-07b3-4f71-b084-ab89604dc097 |   network   | fc2549af-6eec-47a5-
bc78-1831f362b6b3 |
| 305 | 9c1a8131-4232-4f9b-ac5e-b694c5a1168c |   network   | 9e64c55e-8895-491d-91ed-
cfc81cd5586a |
+-----+--------------------------------------+-------------
+--------------------------------------

| Operation |   state    | Retry Count |
+-----------+------------+-------------+
|   create  | processing |      0      |
|  create   | processing |      0      |
|   create  | processing |      0      |
|   create  |  pending   |      0      |
|   create  |  pending   |      0      |
+-----------+------------+-------------+
```

efa-journal reset

Resets the retry count for all or specified failed entries and retries the operation.

Syntax

efa-journal reset[*UUID*]

Parameters

UUID

Specifies the UUID of a failed journal entry that you want to reset.

Examples

The following example resets a specific failed entry.

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

The following example resets all failed entries.

```
$ efa-journal reset
```

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

openstack router create

Creates a router in centralized mode.

Syntax

openstack router create [*router-name* **--centralized**]

Parameters

router-name **--centralized**

Creates a router in centralized mode.

Usage Guidelines

In centralized mode, routing is configured only on the border leaf pairs.

When you create a router, the default mode is centralized. You can create a router with or without the **--centralized** option and achieve the same result.

Note
EFA does not support routers configured in distributed mode, although the feature was supported in a previous release. The functionality to configure distributed mode is disabled.

Each of these examples creates 1 centralized router called R2.

```
$ openstack router create R2

$ openstack router create R2 --centralized
```

Appendix: Neutron REST Endpoints

Neutron REST Endpoints

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

EFA Topology Neutron Extension

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

Normal response codes: 200
```

Table 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 toology link from its asscociate 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

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

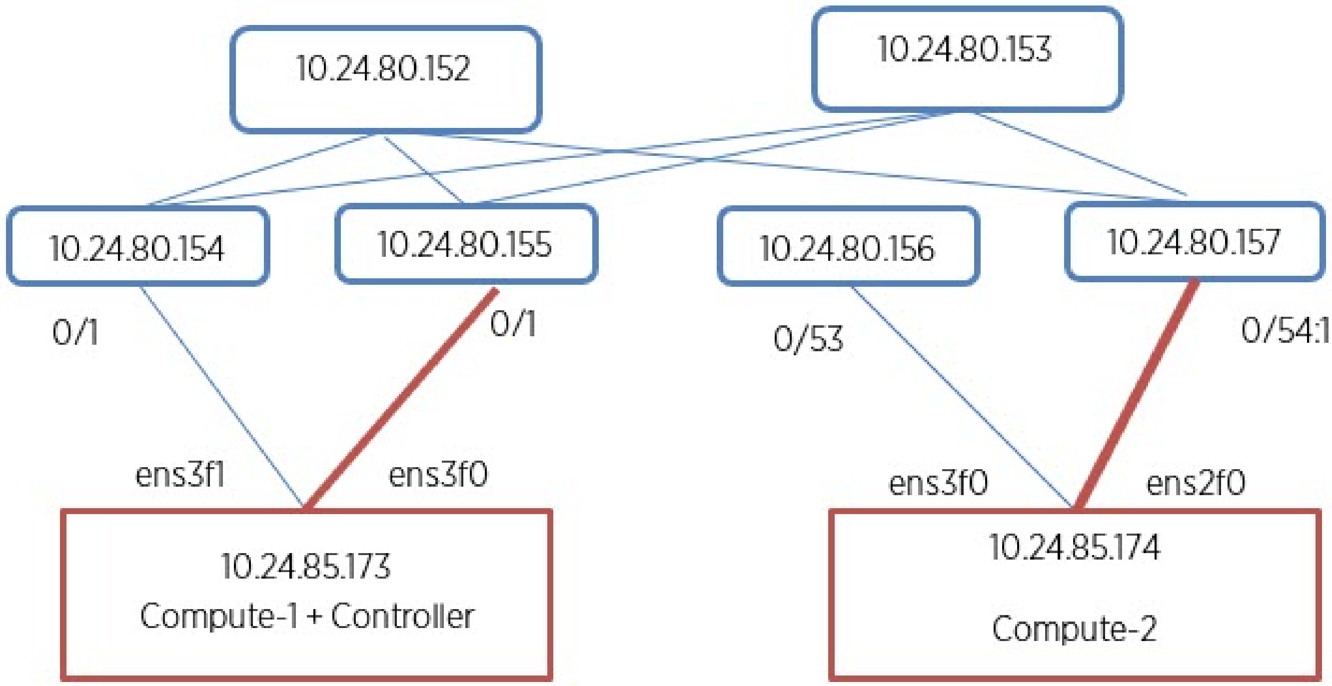


Figure 14: SR-IOV network (VF passthrough)

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","device_type": "type-PF"},{ "devname": "ens3f0", "physical_network":
"sriovnet1"}]
```

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

Configure SR-IOV Agent

Procedure

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

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

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

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

3. Repeat the procedure for Compute node 2.

Configure Nova Scheduler

Procedure

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

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

2. Restart the Nova Scheduler.

Configure Mechanism Drivers for SR-IOV

Procedure

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

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

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

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

3. Restart the Neutron 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

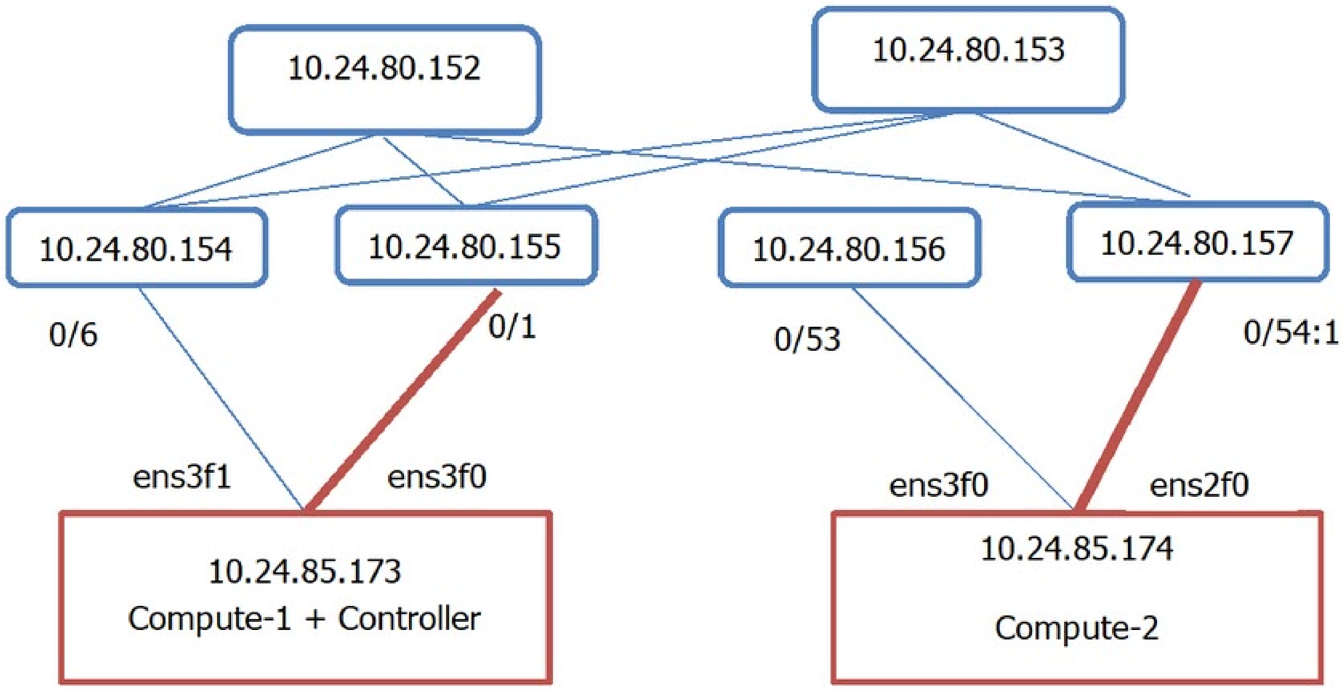


Figure 15: Overview of multi-segment network

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
```