Extreme Fabric Automation

OpenStack Integration Guide, 2.2.0

9036701-00 Rev AA
June 2020

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/

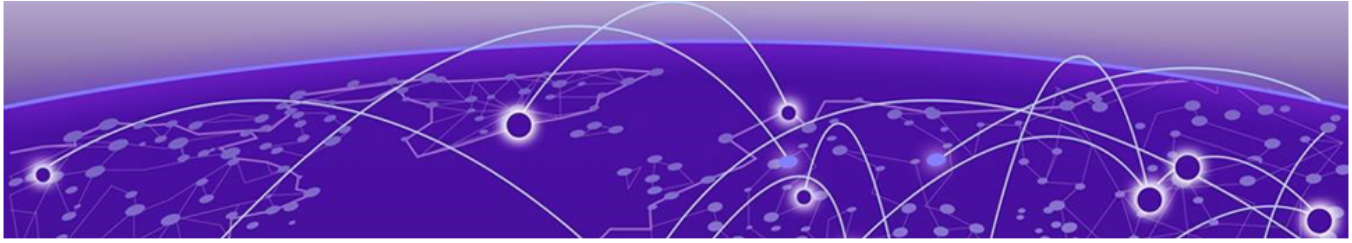


Table of Contents

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.

Table 1: Notes and warnings (continued)

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

Table 3: Command syntax

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

Documentation and Training

Find Extreme Networks product information at the following locations:

Current Product Documentation

Release Notes

Hardware and software compatibility for Extreme Networks products

Extreme Optics Compatibility

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

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

Getting Help

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

Extreme Portal

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

The Hub

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

Call GTAC

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

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

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

Subscribe to Service Notifications

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

1. Go to www.extremenetworks.com/support/service-notification-form.
2. Complete the form (all fields are required).

3. Select the products for which you would like to receive notifications.

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

4. Select **Submit**.

Providing Feedback

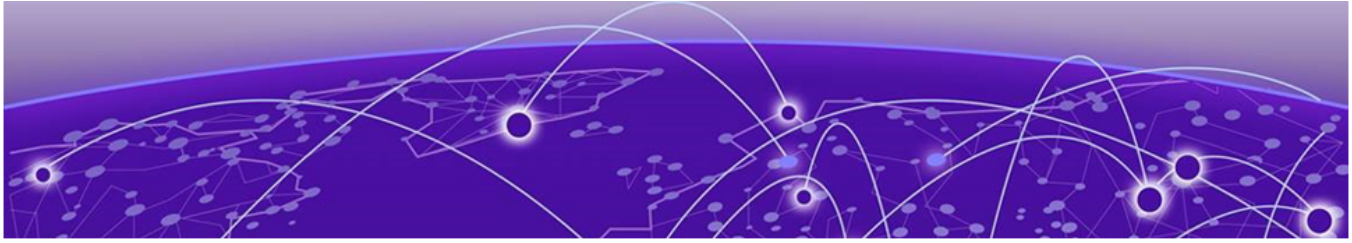
The Information Development team at Extreme Networks has made every effort to ensure the accuracy and completeness of this document. We are always striving to improve our documentation and help you work better, so we want to hear from you. We welcome all feedback, but we especially want to know about:

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

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

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

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



Introduction to OpenStack Integration

Extreme Fabric Automation Overview

Extreme Fabric Automation (EFA) is a microservices-based application that manages the life cycle of IP Fabric CLOS and Small Data Center deployments. All of the microservices support REST APIs that are detailed by OpenAPI.

EFA offers unique flexibility in supporting multiple IP Fabric topologies based on a BGP underlay with a BGP/EVPN overlay:

- Small Data Center Fabric (non-CLOS topology from a single switch pair up to four switch pairs)
- 3-stage CLOS (Leaf / Spine)
- 5-stage CLOS (Leaf / Spine / Super Spine)

Tenant Network onboarding services are supported on all the topologies, which allows you to create connectivity for devices that are connected to the fabric, such as compute (servers), storage, and any other connectivity needed such as external routers or gateways.

Life cycle management of the Fabric allows you to add or delete devices to the Fabric after Day 0. Similarly, you can add or remove Tenants as necessary. Key ecosystem integrations streamline Tenant and network provisioning by way of VMware vCenter, Microsoft System Center for Virtual Machine Management (SCVMM) and OpenStack with ML2 and L3 service plugins.

EFA Core Services and Integrations

EFA comprises several core containerized services that interact with each other and with other infrastructure services to provide the core functions of IP Fabric automation.

Service	Description
Asset Service	Provides the secure credential store and deep discovery of physical and logical assets of the managed devices, and publishes the Asset refresh and change events to other services.
Fabric Service	Helps orchestrate and visualize BGP-EVPN-based 3-stage and 5-stage IP Clos and Non-Clos fabrics.
Tenant Service	Helps manage the Tenants, Tenant Networks, and end points, fully leveraging the knowledge of Assets and the underlying fabric.
Inventory Service	Acts as an inventory of all the necessary physical and logical assets of the fabric devices. All other EFA services rely on inventory service asset data for their respective configuration automation.
System Service	Provides EFA system utilities such as support-save, backup, and restore.
Notification Service	Sends events, alerts, and task updates to external entities.
Authentication Service	Enforces a security boundary between northbound clients and downstream operations between EFA and SLX.
Authorization Service	Validates users and their credentials.

EFA also provides a microservice for each ecosystem integration. This architecture permits rapid development and integration of different ecosystem integrations. Each operates independently to externally integrate while using the same underlying services to interact with the IP Fabric.

Ecosystem	Description
VMware vCenter	The vCenter integration provides connectivity between EFA and vCenter using a REST API as documented in the VI SDK. EFA does not connect to individual ESXi servers. All integration is done through vCenter.
OpenStack	OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a data center.
Microsoft Hyper-V	The Hyper-V integration supports networking configuration for Hyper-V servers in a data center, manual and automated configuration updates when VMs move, and visibility into the VMs and networking resources that are deployed in the Hyper-V setup.

The following figure illustrates the application functionality in provisioning and discovery.

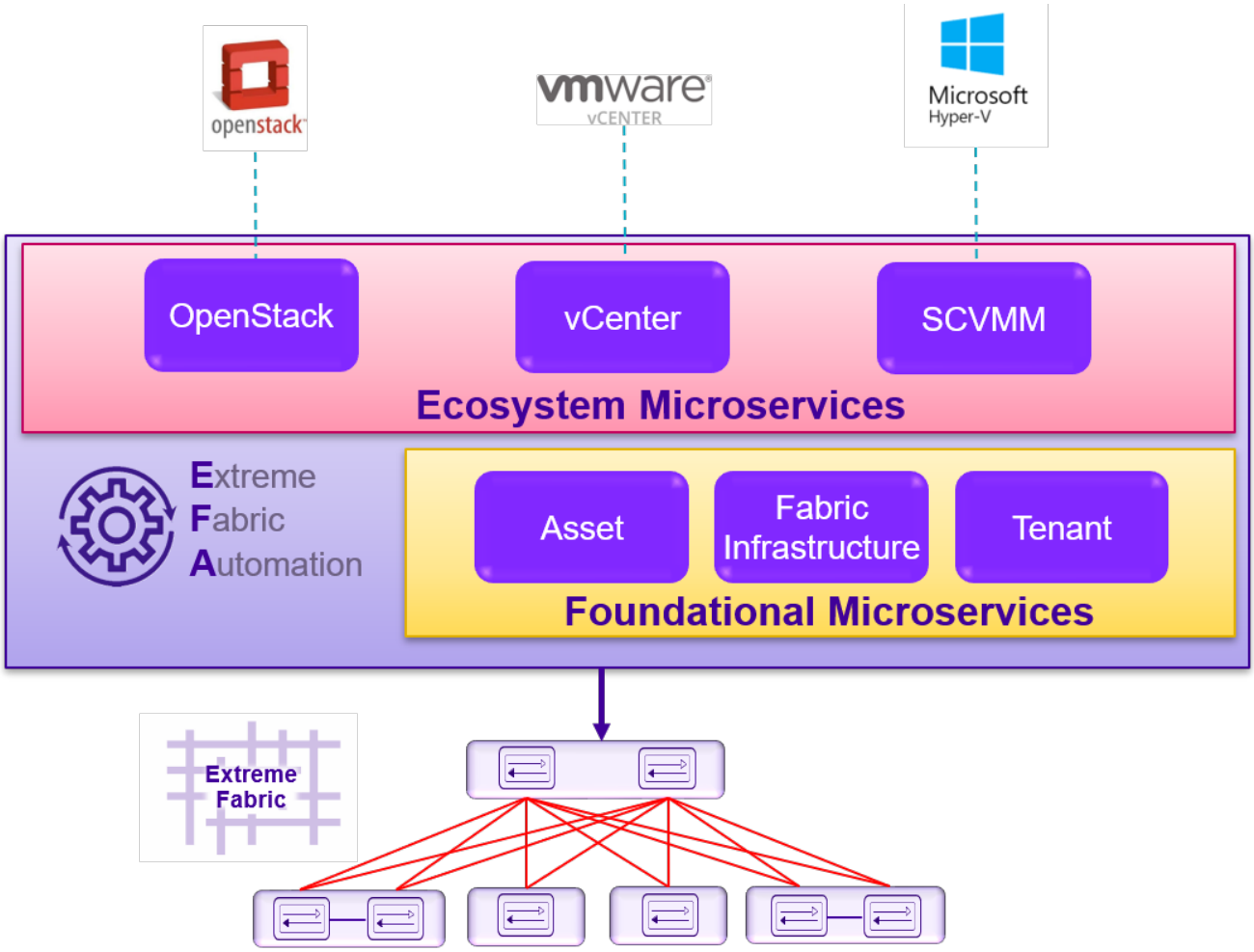


Figure 1: Extreme Fabric Automation Microservices

OpenStack Integration Overview

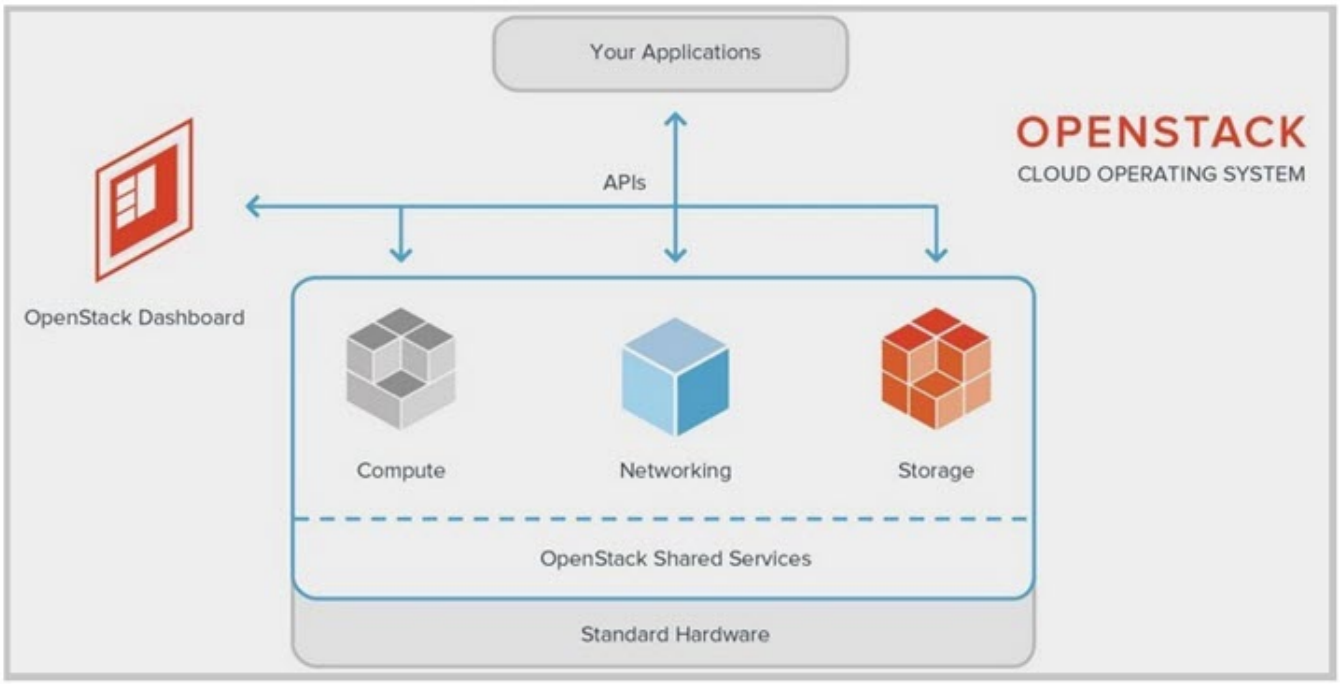
OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a Data Center. OpenStack Integration enables System administrators to manage and provision resources through APIs and web interface.

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 2: 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.

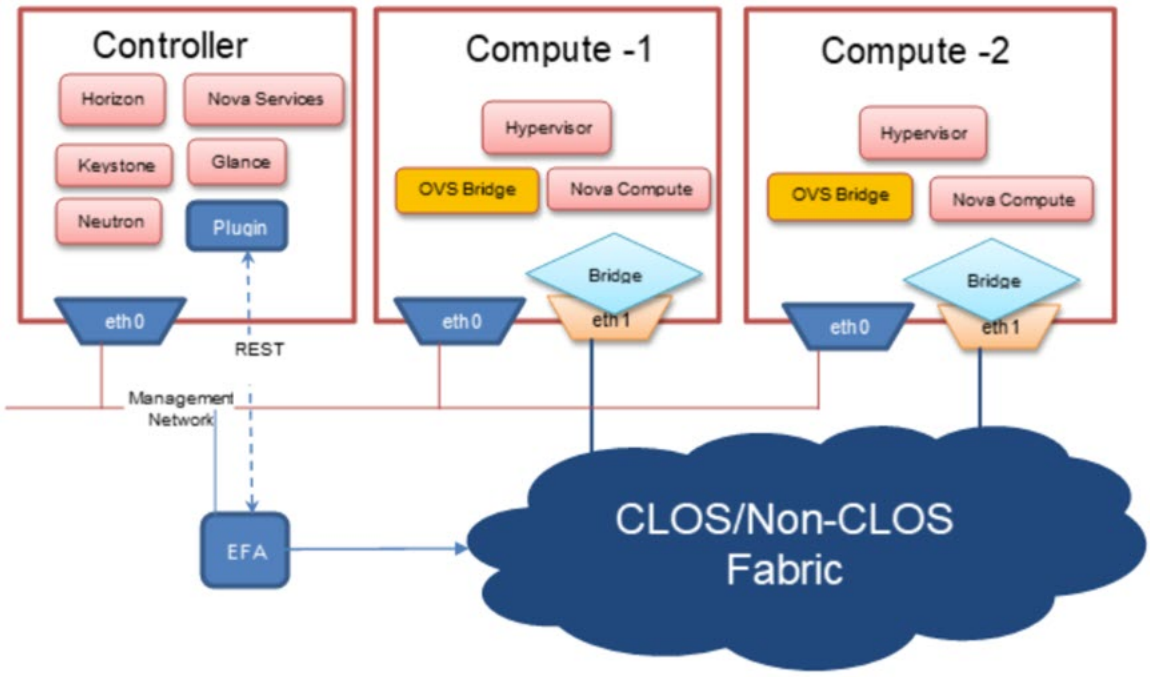


Figure 3: Overview of EFA OpenStack Network

OpenStack Ecosystem Integration Management

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

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

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

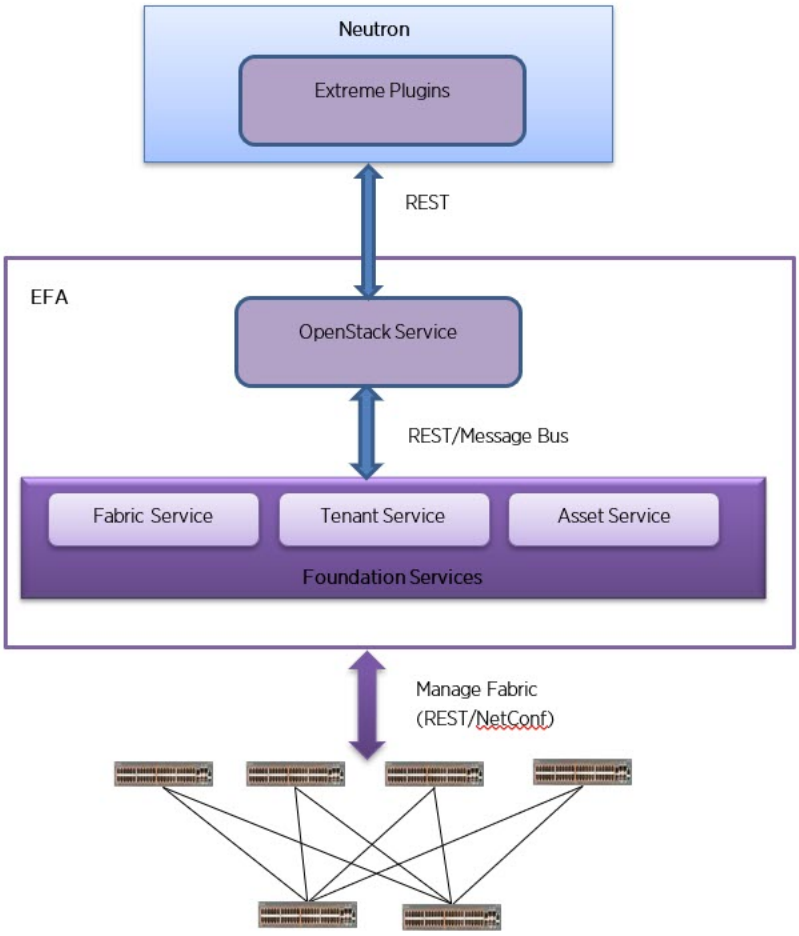


Figure 4: Overview of OpenStack ecosystem integration management

OpenStack Service Commands

OpenStack Service commands show the Neutron constructs and their provisioning status on EFA. For more information on OpenStack Service commands, refer to Appendix: OpenStack Service Command Reference on page 62.

Limitations

OpenStack Ecosystem Integration limitations are as follows:

- Only green field deployment is supported.
- BGP on Compute nodes setup and SR-IOV specific LAG issues require testing and special handling.

Dependencies

OpenStack Ecosystem Integration dependencies are as follows:

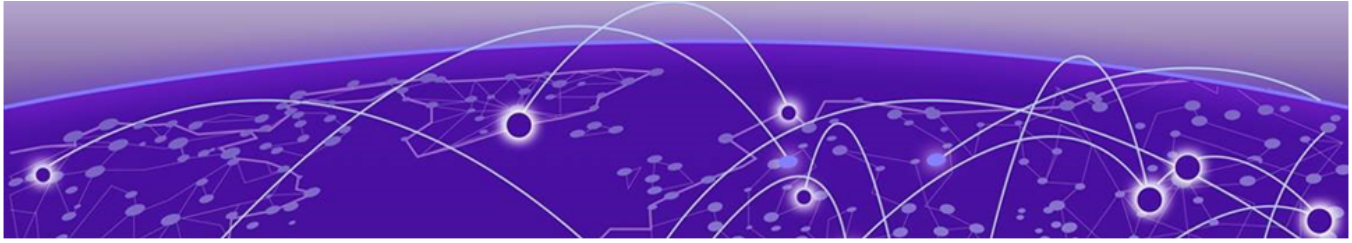
- Non-CLOS (2-Node, 4-Node, and 8-Node)

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

Supported Platforms

OpenStack Ecosystem Integration is supported on the following platforms.

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



OpenStack Plug-in Installation

EFA OpenStack Plug-in Package

OpenStack Integration requires Modular Layer 2 (ML2) Mechanism Driver and L3 Service plug-ins to interact with Extreme Fabric Automation (EFA). The Neutron drivers and plug-ins communicate with EFA over the REST interface.

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

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

- Modular Layer 2 (ML2) Mechanism Driver (Neutron)
- Topology
- Single Root I/O Virtualization (SR-IOV)
- Multi Segment
- Virtual Machine Migration (VMotion)
- OpenStack Service
- L3 Service

System Requirements

System requirements for EFA OpenStack Integration are as follows:

Node	
CPU cores	4
Storage	50 G
RAM	16 GB
OpenStack	Pike
EFA	2.1.0

Node	
Operating System	Ubuntu 16.04
Python	2.7

Prepare the Site

Before You Begin

Note the following details before preparing the site for installing the EFA OpenStack Neutron plug-in:

- EFA IP Address (Where EFA is running)
- EFA Port number (80)
- EFA auth Token (extremenetworks_user_auth_key)
- Region_name (RegionOne)
- Server interface name that is connected to fabric (eth2 or bond0)

Procedure

1. Download the EFA plug-in debian packages TAR file for OpenStack from https://extremeportal.force.com/.
2. Untar and extract the TAR file on the required network node.

```
tar xvzf-efa-neutron-plugins-2.1.0-1.tar.gz
```

3. Configure the Compute Provider Network and server connections.

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

`link.csv`

```
CCEP:
<compute-host>,<compute-nic>,<switch-ip>, <switch-port>, <lag name>
Compute1,ens2f0,<MCT Leaf1>,<0/10:1>,lag_Compute1
Compute1,ens2f1,<MCT Leaf2>,<0/10:1>,lag_Compute1
CEP/Single Homed Leaf:
Compute2,ens2f0,<MCT Leaf1>,<0/20:1>
Compute3,ens2f0,<MCT Leaf1>,<0/20:2>
```

`pn.csv`

```
<compute-host>,<openStack-provider-brigde>,<compute-nic>
Compute1,physnet1,ens2f0
Compute1,physnet1,ens2f1
Compute2,physnet1,ens2f0
Compute3,physnet1,ens2f0
```

4. Set Name resolution for Controller, Compute, and Networking nodes in `/etc/hosts`.

Note
Ensure that all OpenStack nodes have unique hostnames.

```
10.24.51.114 OpenStack114
10.24.51.115 OpenStack115
```

5. Configure the ovs bridge name in the `ovs` configuration file.

   Note
   Ensure that the name of the bridge matches with the one configured as `bridge_mappings` in `ml2_conf.ini`.

```
sudo ovs-vsctl add-br br0
```

6. Add physical data ports or bond to the ovs bridge as required.
   - Bonding: **`sudo ovs-vsctl add-bond br0 bond0 eno2 eno3 lacp=active`**
   - Single port: **`sudo ovs-vsctl add-port br0 eno2`**

What to Do Next

Install EFA OpenStack Neutron Plug-in

Install EFA OpenStack Neutron Plug-in

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

Before You Begin

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

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

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

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

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

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

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

3. Note the Neutron configuration file layout.
   - Neutron configuration: `/etc/neutron/neutron.conf`
   - ML2 plug-in configuration: `/etc/neutron/plugins/ml2/ml2_conf.ini`
   - Extreme EFA Mechanism driver or topology configuration: `/etc/neutron/plugins/ml2/ml2_conf_extreme.ini`
4. Configure the `M12 core_plugin` in the `neutron.conf` file.

   Do not enable the reference router plug-in.

   ```
   [DEFAULT]
   core_plugin=ml2
   service_plugins = extreme_l3_efa, trunk, segments, efa_topology_plugin
   ```
5. Copy the `ml2_conf_extreme.ini` file provided with TAR to the `/etc/neutron/plugins/ml2/ml2_conf_extreme.ini` file.
6. Enable Neutron in the `ml2_conf_extreme.ini` file to communicate with EFA.

   ```
   [ml2_extreme]
   efa_rest_token = extremenetworks_user_auth_key
   efa_port = 80
   efa_host = <ip-where-efa-is-running>
   region_name = RegionOne (unique name for each VIM)
   ```
7. Enable the Neutron EFA extension plug-in in the `ml2_conf_extreme.ini` file to build initial physical topology between OpenStack Compute nodes and TOR switches.

   ```
   [efa_topology]
   efa_pn_mapping_file = /home/ubuntu/pn.csv
   efa_link_mapping_file = /home/ubuntu/link.csv
   ```
8. 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)
   ```
9. 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.
10. Restart the Neutron server.

    ```
    # systemct1 restart
    ```

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

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

11. Verify if the status of the Neutron server is `Active` and confirm the following:
    - Neutron service started with the `ml2_confi_extreme.ini` file.
    - **`efa-topology`** extension is loaded using **`OpenStack extension show efa-topology`**.

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

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

Upgrade or Downgrade OpenStack Neutron Plug-in

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

   ```
   # cd efa
   ```

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

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

Uninstall EFA OpenStack Neutron Plug-in

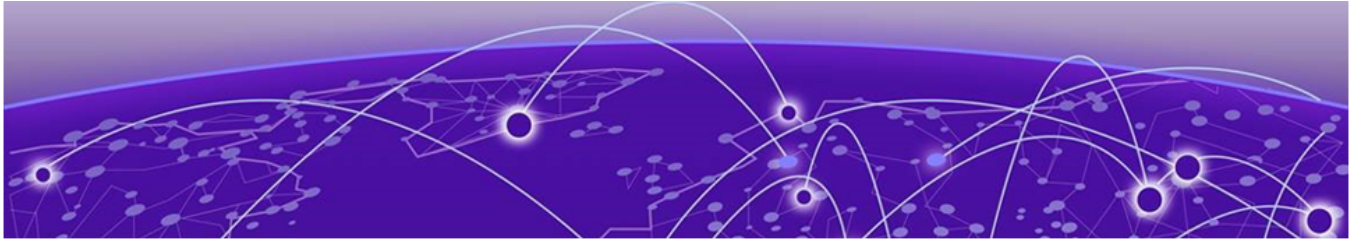
Procedure

1. Log in to the network node using SSH and go to the EFA folder.

   ```
   # cd efa
   ```

2. Uninstall the EFA OpenStack Neutron plug-in.

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



OpenStack Network Configuration

Provider Network and VM Setup

The following sections describe how to setup a provider network and launch VM instances with NICs on the provider network.

Configure IP Fabric

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Add devices to the fabric.

```
# efa fabric device add-bulk --name default --spine 10.24.80.152-153 --leaf
10.24.80.154-157 --username admin --password password
```

3. Configure the fabric.

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

4. Create the tenant.

```
# efa tenant create --name RegionOne --port 10.24.80.154[0/1-6],10.24.80.155[0/1-6],
10.24.80.156[0/53],10.24.80.157[0/54:1]
```

5. Update the tenant.

```
# efa tenant update --name RegionOne --vlan-range 100-500 --operation vlan-update
```

Example

The following example shows the physnet and link mappings on the Compute or Controller nodes.

```
user@Compute-1:~$ sudo efa-pn-mapping list
+--------------+-----------------+---------+
|     Host     | ProviderNetwork |  Nic    |
+--------------+-----------------+---------+
| 10.24.85.174 |    sriovnet2    | ens3f1 |
| 10.24.85.173 |    sriovnet1    | ens3f0 |
| 10.24.85.173 |    sriovnet2    | ens3f1 |
| 10.24.85.174 |    sriovnet1    | ens2f0 |
```

```
+--------------+-----------------+--------+

user@Compute-1:~$ sudo efa-link-mapping list
+--------------+--------+--------------+--------+---------+
|     Host     |  Nic   |    Switch    |  Port  | Po-Name |
+--------------+--------+--------------+--------+---------+
| 10.24.85.173 | ens3f1 | 10.24.80.154 |  0/6   |         |
| 10.24.85.174 | ens2f0 | 10.24.80.157 | 0/54:1 |         |
| 10.24.85.173 | ens3f0 | 10.24.80.155 |  0/1   |         |
| 10.24.85.174 | ens3f1 | 10.24.80.156 |  0/53  |         |
+--------------+--------+--------------+--------+---------+
```

Create a Provider Network

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

   ```
   # cd efa
   ```

2. Create a provider network, `provider-vlan171` with CTAG 171.

   ```
   # openstack network create provider-vlan171 \ --provider-network-type vlan \ --
   provider-physical-network physnet1 \ --provider-segment 171 \ --share
   ```

Create a Subnet

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

   ```
   # cd efa
   ```

2. Create a subnet with Gateway and IP Range.

   ```
   # openstack subnet create subnet-provider-171 \ --network provider-vlan171 \ --subnet-
   range 10.65.217.0/24 \ --gateway 10.65.217.254
   ```

Create VM Instances on Provider Network

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

   ```
   # cd efa
   ```

2. Create a VM Instance on the Provider Network on Compute-1.

   ```
   # openstack server create \ --flavor m1.nano \ --image cirros-0.4.0-x86_64-disk \ --
   nic net-id=$(openstack network list | awk '/ provider-vlan171/ {print $2}') \ --
   availability-zone Compute-1 provider-instance-1
   ```

3. Repeat the procedure to create a VM Instance on the Provider Network on Compute-2.

Delete VM Instances on Provider Network

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

   ```
   # cd efa
   ```

2. Delete the VM instance on the Provider Network Compute-1.

```
# openstack server delete provider-instance-1
```

3. Repeat the procedure to delete VM instances on the Provider Network on Compute-2.

Delete Provider Network

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Delete the Provider Network.

```
# openstack network delete provider-vlan171
```

Virtual Machine Migration

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

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

Enable VMotion

Procedure

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

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

```
vncserver_listen = 0.0.0.0
instances_path = /var/lib/nova/instances
restorecon = /etc/hosts
```

2. Enable password-less SSH.

   The libvirt daemon that runs as root uses SSH protocol to copy the instance to the destination.

3. Configure the firewalls to allow libvirt to communicate with Compute nodes.

   The default libvirt TCP port range is 49152 to 49261 for copying memory and disk contents.

Migrate Virtual Machines

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. 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 | ...         |
+--------------------------------------+------+--------+-----------------+-------------+
```

3. 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.
4. 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                               |
| ...                  | ...                                  |
+----------------------+--------------------------------------+
```

5. 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 |
+-------+------------+-----+-----------+---------+
```

6. 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
     ```
7. 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                                |
| ...                  | ...                                  |
+----------------------+--------------------------------------+
```

8. (Optional) View the log files on the Controller and Compute nodes for more information about migration failure.
   - `nova-scheduler`
   - `nova-conductor`
   - `nova-compute`
9. (Optional) Stop the migration manually.
   ```
   # nova live-migration-abort INSTANCE_ID MIGRATION_ID
   ```
10. (Optional) Force complete the migration.
    ```
    # nova live-migration-force-complete INSTANCE_ID MIGRATION_ID
    ```

Topology Setup in Neutron

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

Extreme Topology plug-in provides **`efa-pn-mapping`** and **`efa-link-mapping`** commands to configure and setup the topology.

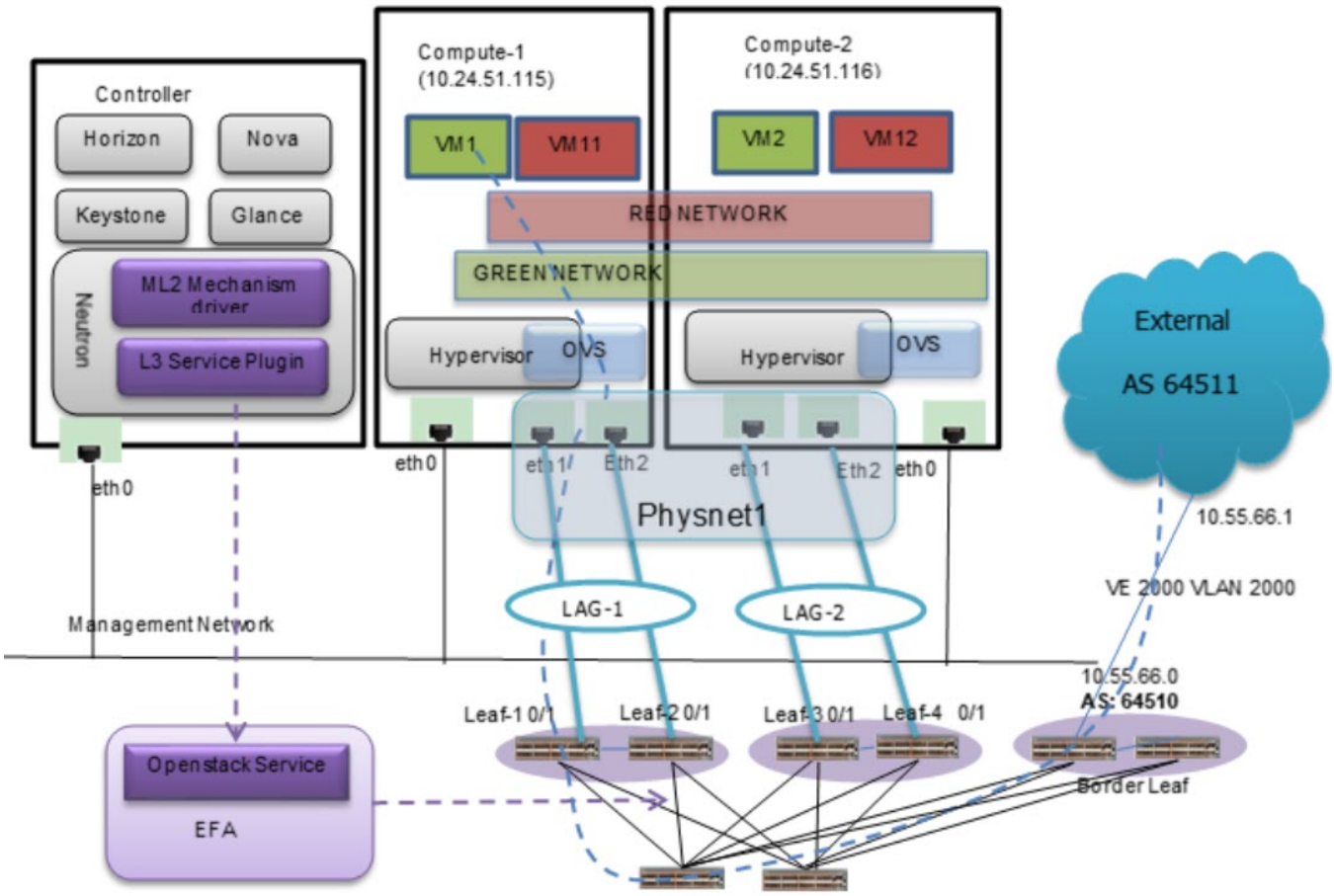


Figure 5: Topology setup in Neutron

Node	IP Address / Description
Compute-1	10.24.51.115
Compute-2	10.24.51.116
Leaf-1	10.24.14.133
Leaf-2	10.24.14.134
Leaf-3	10.24.14.135
Leaf-4	10.24.14.136
Physnet1	Provider Network

Setup Topology in Neutron

About This Task

Note

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

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Map the Provider Network to Compute NICs.

```
# efa-pn-mapping add --host 10.24.51.115 --nic eth1 --provider-network physnet1
# efa-pn-mapping add --host 10.24.51.115 --nic eth2 --provider-network physnet1
# efa-pn-mapping add --host 10.24.51.116 --nic eth1 --provider-network physnet1
# efa-pn-mapping add --host 10.24.51.116 --nic eth2 --provider-network physnet1

ubuntu@Openstack114:~$ cat /home/ubuntu/pn.csv
10.24.51.115,physnet1,eth1
10.24.51.115,physnet1,eth2
10.24.51.116,physnet1,eth1
10.24.51.116,physnet1,eth2

ubuntu@Openstack114:~$ efa-pn-mapping list
+--------------+-----------------+------+
|     Host     | ProviderNetwork | Nic  |
+--------------+-----------------+------+
| 10.24.51.115 |     physnet1    | eth1 |
| 10.24.51.115 |     physnet1    | eth2 |
| 10.24.51.116 |     physnet1    | eth1 |
| 10.24.51.116 |     physnet1    | eth2 |
+--------------+-----------------+------+
```

3. Map Compute NIC with SLX Interfaces with LAG details.

```
# efa-link-mapping add --host 10.24.51.115 --nic eth1 --switch 10.24.14.133 --port 0/1
--po-name lag_1
# efa-link-mapping add --host 10.24.51.115 --nic eth2 --switch 10.24.14.134 --port 0/1
--po-name lag_1
# efa-link-mapping add --host 10.24.51.116 --nic eth1 --switch 10.24.14.135 --port 0/1
--po-name lag_2
# efa-link-mapping add --host 10.24.51.116 --nic eth2 --switch 10.24.14.136 --port 0/1
--po-name lag_2

ubuntu@Openstack114:~$ cat /home/ubuntu/link.csv
10.24.51.115,eth1,10.24.14.133,0/5,lag_1
10.24.51.115,eth2,10.24.14.134,0/6,lag_1
10.24.51.116,eth1,10.24.14.135,0/5,lag_2
10.24.51.116,eth2,10.24.14.136,0/6,lag_2

ubuntu@Openstack114:~$ efa-link-mapping list
+--------------+------+--------------+------+---------+
|     Host     | Nic  |    Switch    | Port | Po-Name |
+--------------+------+--------------+------+---------+
| 10.24.51.115 | eno2 | 10.24.14.133 | 0/1  | lag_1   |
| 10.24.51.115 | eno3 | 10.24.14.134 | 0/1  | lag_1   |
| 10.24.51.116 | eno2 | 10.24.14.135 | 0/1  | lag_2   |
| 10.24.51.116 | eno3 | 10.24.14.136 | 0/1  | lag_2   |
+--------------+------+--------------+------+---------+
```

Add Provider Network Mapping

About This Task

Note

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

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Add Provider Network mapping.

```
# sudo efa-pn-mapping  add -H compute-hostname/ip-address -p physical-network-alias-name -n NIC

# sudo efa-pn-mapping add -H 10.24.51.114 -p physnet1 -n eth1
```

Example

The following example shows the CSV file format for including all the Neutron provider network and mapping information.

```
ps.csv

<compute-host>,<openStack-provider-brigde>,<compute-nic>
Compute1,physnet1,ens2f0
Compute1,physnet1,ens2f1
Compute2,physnet1,ens2f0
Compute3,physnet1,ens2f0
```

The **sudo efa-pn-mapping add -f pn.csv** command reads the CSV file and maps the provider network to NICs.

```
# sudo efa-pn-mapping add -H compute-hostname/ip-address -p physical-network-alias-name -n NIC

# sudo efa-pn-mapping add -H 10.24.51.114 -p physnet1 -n eth1
```

Dump Provider Network Mapping

The **efa-pn-mapping dump** command enables you to dump all the host NIC and Provider Network mapping to a file for future use.

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Dump provider network mapping to a file.

```
# efa-pn-mapping dump -F <file-name>
```

Example

```
# sudo efa-pn-mapping dump -F pn.csv
```

Verify Provider Network Mapping

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Verify Provider Network mapping.

```
# efa-pn-mapping list
```

Example

```
# sudo efa-pn-mapping list

+--------------+------+--------------+------+---------+
|     Host     | Nic  |    Switch    | Port | Po-Name |
+--------------+------+--------------+------+---------+
| 10.24.51.114 | eth1 | 10.24.12.133 | 0/1  |         |
| 10.24.51.115 | eth1 | 10.24.12.134 | 0/4  |         |
+--------------+------+--------------+------+---------+
```

Remove Provider Network Mapping

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

   ```
   # cd efa
   ```

2. Remove host NICs from Provider Network mapping.
   - Remove all the mappings.

     ```
     # sudo efa-pn-mapping remove
     ```

   - Remove all the mappings for the selected host.

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

   - Remove the selected mapping.

     ```
     # sudo efa-pn-mapping remove -H compute1 -p physnet1 -n eth0
     ```

Add Link Mapping

About This Task

Note

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

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

   ```
   # cd efa
   ```

2. Add link mapping.
   - If bonding is enabled on Compute node.

     ```
     # Sudo efa-link-mapping add -H compute-hostname/ip-address -n NIC -s switch-ip -p
     port-name -P portchannel-name
     ```

   - If bonding is not enabled on Compute node.

     ```
     # Sudo efa-link-mapping  add -H compute-hostname/ip-address -n NIC -s switch-ip -p
     port-name
     ```

Example

The following example shows the `CSV` file format for including all the host NIC and switch port mapping information.

```
Link.csv

CCEP:
<compute-host>,<compute-nic>,<switch-ip>, <switch-port>, <lag name>
```

```
Compute1,ens2f0,<MCT Leaf1>,<0/10:1>,lag_Compute1
Compute1,ens2f1,<MCT Leaf2>,<0/10:1>,lag_Compute1

CEP/Single Homed Leaf:
Compute2,ens2f0,<MCT Leaf1>,<0/20:1>
Compute3,ens2f0,<MCT Leaf1>,<0/20:2>
```

The **sudo efa-link-mapping add -f link.csv** command reads the CSV file and maps NICs and switch ports.

```
# sudo efa-pn-mapping add -H 10.24.51.114  -n eth1 -s 10.24.14.134 -p 0/1 5

# sudo efa-pn-mapping add -H 10.24.51.115  -n eth1 -s 10.24.14.134 -p 0/2 -P lag_115

# sudo efa-pn-mapping add -H 10.24.51.115  -n eth2 -s 10.24.14.133 -p 0/2 -P lag_115
```

Dump Link Mapping

The **efa-link-mapping dump** command enables you to dump all the host NIC and switch port mapping to a file for future use.

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

   ```
   # cd efa
   ```

2. Dump link mapping.

   ```
   # efa-link-mapping dump -F <file-name>
   ```

Example

```
# sudo efa-link-mapping dump -F link.csv
```

Verify Link Mapping

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

   ```
   # cd efa
   ```

2. Verify link mapping.

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

Example

```
# sudo efa-link-mapping list

+--------------+------+--------------+------+---------+
|     Host     | Nic  |    Switch    | Port | Po-Name |
+--------------+------+--------------+------+---------+
| 10.24.51.115 | eno2 | 10.24.14.133 | 0/6  | lag_115 |
| 10.24.51.115 | eno3 | 10.24.14.134 | 0/5  | lag_115 |
| 10.24.51.116 | eno2 | 10.24.14.133 | 0/16 | lag_116 |
| 10.24.51.116 | eno3 | 10.24.14.134 | 0/15 | lag_116 |
| 10.24.51.114 | eno2 | 10.24.14.133 | 0/5  | lag_114 |
| 10.24.51.114 | eno3 | 10.24.14.134 | 0/6  | lag_114 |
```

Remove Link Mapping

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Remove link mappings.
   - Remove all the mappings.

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

   - Remove all the mappings for the selected host.

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

   - Remove the selected mapping.

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

Tenant or Project Network

The following sections describe how to setup a tenant or project network and launch VM instances with NICs on the provider network.

Create a Tenant Network

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Create a Tenant Network.

Note
VLAN is auto-allocated.

```
# openstack network create tenant-network
```

Create a Tenant Subnet

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Create a subnet with Gateway and IP Range.

```
# openstack subnet create tenant-subnet \ --network tenant-network \ --subnet-range
20.65.217.0/24 \ --gateway 20.65.217.254
```

Create a VM Instance on Tenant Network

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Create a VM Instance on the Tenant Network on Compute-1.

```
# openstack server create \ --flavor m1.nano \ --image cirros-0.4.0-x86_64-disk \ --
nic net-id=$(openstack network list | awk '/ tenant-network/ {print $2}') \ --
availability-zone Compute-1 tenant-instance-1
```

3. Repeat the procedure to create VM Instances on the Tenant Network on Compute-2.

Delete VM Instances on Tenant Network

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Delete the VM Instance on the Tenant Network on Compute-1.

```
# openstack server delete tenant-instance-1
```

3. Repeat the procedure to delete VM Instances on the Tenant Network on Compute-2.

Delete Tenant Network

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Delete the Tenant Network.

```
# openstack network delete tenant-network
```

Network Trunking

Network trunking allows multiple networks to connect to an instance using a single virtual NIC (vNIC).

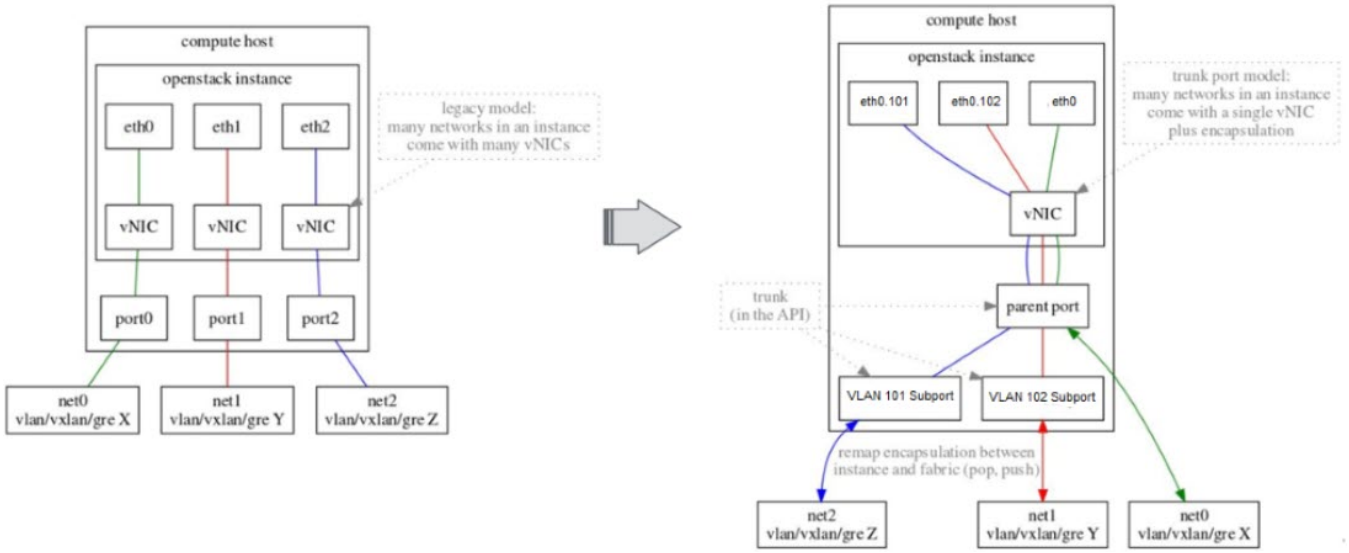


Figure 6: Overview of network trunking

Create a Network for Parent Trunk

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Create a network for the parent trunk.

```
# openstack network create net0 \ --provider-network-type vlan \ --provider-physical-network physnet1 \ --provider-segment 100
```

3. Create a subnet.

```
# openstack subnet create subnet0 \ --network net0 \ --subnet-range 20.0.4.0/24
```

4. Create a port.

```
# openstack port create port0 \ --network net0
```

Create a Network for Subnet

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Create a network for subnet 1.

```
# openstack network create net1 \ --provider-network-type vlan \ --provider-physical-network physnet1 \ --provider-segment 101

# openstack subnet create subnet1 \ --network net1 \ --subnet-range 20.0.5.0/24
parent_mac="$( openstack port show port0 | awk '/ mac_address / { print $4 }' )"

# openstack port create port1 \ --network net1 --mac-address  "$parent_mac”
```

3. Repeat the procedure to create a network for subnet 2.

Create a Trunk Port

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Create a trunk port.

```
# openstack network trunk create trunk0 \ --parent-port port0
```

Add Subports to Trunk

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Add subports to a trunk.

```
# openstack network trunk set trunk0 \ --subport port=port1,segmentation-type=inherit

# openstack network trunk set trunk0 \ --subport port=port2,segmentation-type=inherit

# openstack network subport list --trunk trunk0 -f value -c 'Segmentation ID' 101 102
```

Launch VM Instances on Compute Node

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Download the server cloud image file.

```
# wget https://cloud-images.ubuntu.com/bionic/current/bionic-server-cloudimg-amd64.img
```

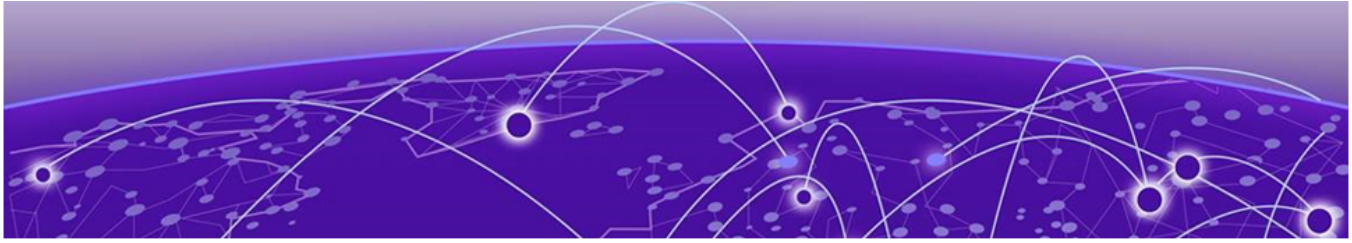
3. Launch the VM instance.

```
# sudo virt-customize -a bionic-server-cloudimg-amd64.img  --root-password
password:password

# openstack image create --disk-format qcow2 --public --file bionic-server-cloudimg-
amd64.img  vlan-capable-image

# The only vNIC in your instance corresponds to the parent port, so boot your instance
with the parent port given.
# Do not add child ports as NICs to 'nova boot / openstack server create'.

# openstack server create --flavor ds512M --image vlan-capable-image --nic port-
id=port0 --availability-zone Compute-1  trunk-instance-1

On the VM
# sudo ip link add link eth0 name eth0.101 type vlan id 101
# sudo ifconfig eth0.101 up
# sudo dhclient eth0.101
```

4. Launch the second VM instance.

```
# openstack port create port0_2 \ --network net0 parent_mac_2="$( openstack port show
port0_2 | awk '/ mac_address / { print $4 }' )"

# openstack port create port1_2 \ --network net1   --mac-address  "$parent_mac_2"

# openstack port create port2_2 \ --network net2 --mac-address  "$parent_mac_2"

# openstack network trunk create trunk0_2 \ --parent-port port0_2

# openstack network trunk set trunk0_2 \ --subport port=port1_2,segmentation-
type=inherit

# openstack network trunk set trunk0_2 \ --subport port=port2_2,segmentation-
type=inherit

# openstack server create --flavor ds512M --image vlan-capable-image --nic port-
id=port0_2  --availability-zone Compute-2  trunk-instance-2

On the VM
# sudo ip link add link eth0 name eth0.101 type vlan id 101
# sudo ifconfig eth0.101 up
# sudo dhclient eth0.101
```

```
# sudo ip link add link eth0 name eth0.102 type vlan id 102
# sudo ifconfig eth0.102 up
# sudo dhclient eth0.102
```



Neutron and L3 Service Configuration

ML2 Mechanism Driver

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

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

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

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

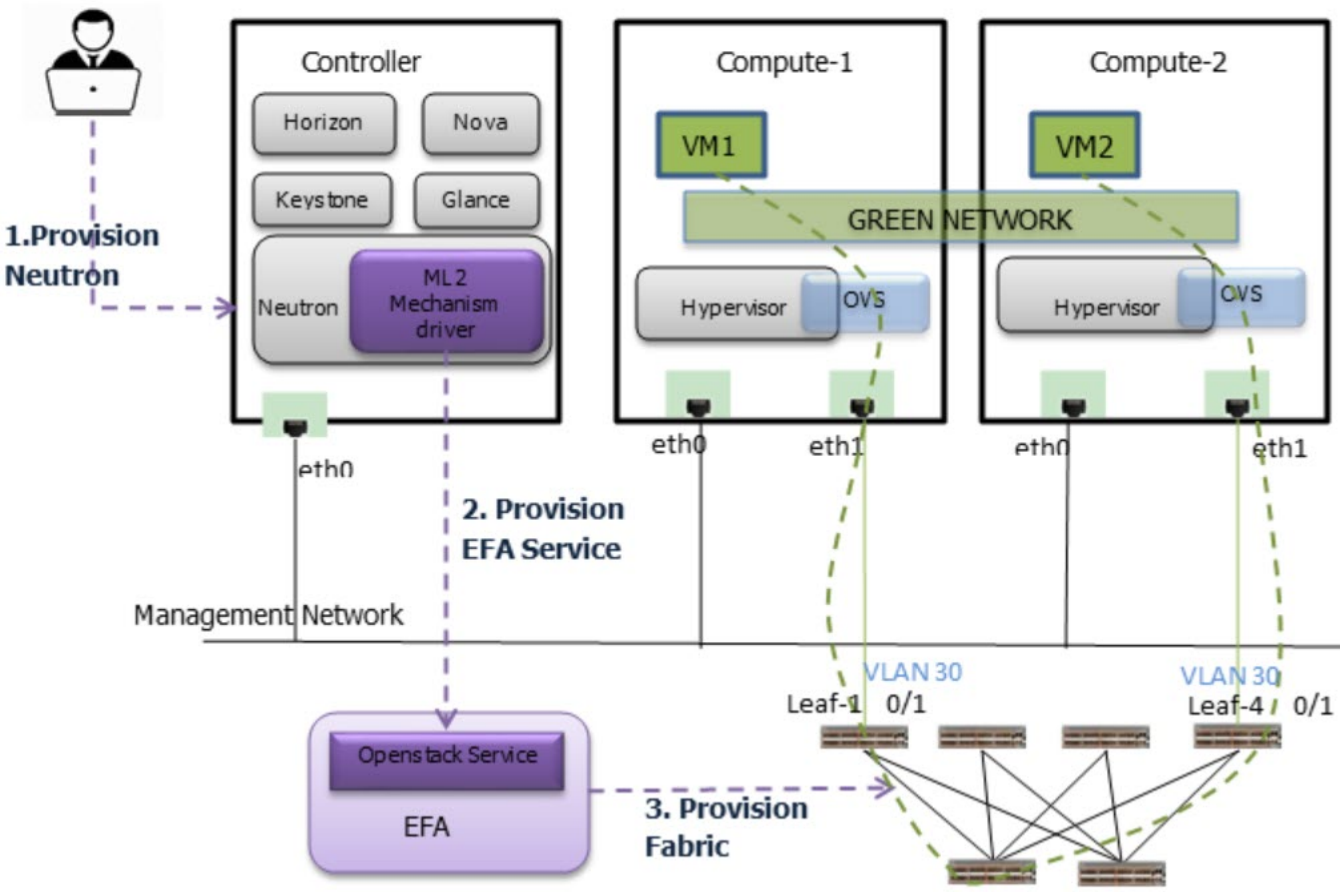


Figure 7: Overview of ML2 Mechanism Driver within Neutron

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

Table 4: ML2 tenant mapping

VM	Neutron Network	Tenant Service (EPG)
VM1	GREEN_NETWORK (VLAN30) UUID =74cbf489-f3d9-41c7-bbb2-6cb7df33da6d	74cbf489-f3d9-41c7-bbb2-6cb7df33da6d • Endpoint: eth 0/1 on Leaf-1 • Endpoint: eth 0/1 on Leaf-4 • CTAG 30
VM2	GREEN_NETWORK (VLAN30) UUID =74cbf489-f3d9-41c7-bbb2-6cb7df33da6d	
Naming convention for EPG creation is `<Neutron Network UUID>`.		

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

The following table shows an example of the configuration on the leaf nodes.

Table 5: ML2 switch configuration

EPG	Switch Configuration	Comments

74cbf489-f3d9-41c7-bbb2-6cb7df33da6d (GREEN_NETWORK)
• Endpoint: eth 0/1 on Leaf-1
• EndPoint: eth 0/1 on Leaf-2
• CTAG 30
	`vlan 30`
	`description L2 Tenant vlan`
	`!`
	`evpn default`
	`vlan add 30`
	`!`
		VLAN 30 configured on Leaf-1 and Leaf-4
		VLAN 30 added to evpn
		
		Note: VLAN to VNI mapping is set to auto as part of fabric setup.

	`interface ethernet 0/1`
	`switchport`
	`switchport trunk allowed vlan add 30`
	`no shutdown`
	`!`
		Interface configurations on both the switches, Leaf-1 and Leaf-4.

Configure Neutron

Procedure

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

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

2. Configure the IP Address and Authentication Token in `/etc/neutron/plugins/ml2/ml2_conf_extreme_efa.ini`.

```
[ml2_extreme]
efa_rest_token = extremenetworks_user_auth_key
efa_port = 80
efa_host = 10.24.51.170
region_name = RegionOne
[efa_topology]
efa_pn_mapping_file = /home/ubuntu/pn.csv
efa_link_mapping_file = /home/ubuntu/link.csv
```

Token Description

Token	Description
`efa_rest_token`	Authentication token used for EFA
`efa_port`	Port running the HTTP service
`efa_host`	IP address of EFA service
`region_name`	Used as the Tenant name by EFA

Token	Description
efa_pn_mapping_file	Mapping of Physical Network to Ethernet NICs
efa_link_mapping_file	Mapping of Physical NICs to Extreme Switch Ports

Create an OpenStack Network Using ML2 Plug-in

About This Task

Extreme ML2 plug-in forwards network and VM creation requests to EFA along with details about VLAN allocated by the Type Driver. OpenStack Service on EFA utilizes the End Point Group (EPG) construct of Tenant Services to provision an EPG.

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Create an OpenStack network, GREEN_NETWORK.

```
# openstack network create GREEN_NETWORK
```

3. Create a Virtual Machine, VM1 on 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. Repeat the procedure to create a Virtual Machine, VM2 on Compute-2 attached to GREEN_NETWORK.

Display Endpoints Provisioned on Switch

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. View endpoints provisioned on the switch.

```
# efa tenant epg show –tenant <tenant-name>
```

Example

```
root@admin1-08:~# efa tenant epg show --tenant RegionOne

Name          : 74cbf489-f3d9-41c7-bbb2-6cb7df33da6d
Description   :
Ports         : 10.24.80.115[eth 0/1], 10.24.80.116[eth 0/1]
POs           :
Port Property : switchport mode     : trunk
              : native-vlan         : 0
              : native-vlan-tagging : false
NW Policy     : ctag-range          : 30
              : vrf                 :
              : l3-vni              :
Network Property
+------+--------+---------------+------------+----------+-------------+-------------+
| Ctag | L2-Vni |  Anycast-ip   | Gateway-ip | Tag-type |  Dev-state  |  App-state  |
+------+--------+---------------+------------+----------+-------------+-------------+
```

```
| 30   | 30     |               |            |          | provisioned | cfg-in-sync |
+------+--------+---------------+------------+----------+-------------+-------------+
```

Mechanism Driver for SR-IOV

ML2 Mechanism Driver for Single Root I/O Virtualization (SR-IOV) capable NIC based switching is supported. ML2 mechanism driver for EFA works when SR-IOV feature is used on the OpenStack.

The following figure shows an overview of the mechanism driver for SR-IOV.

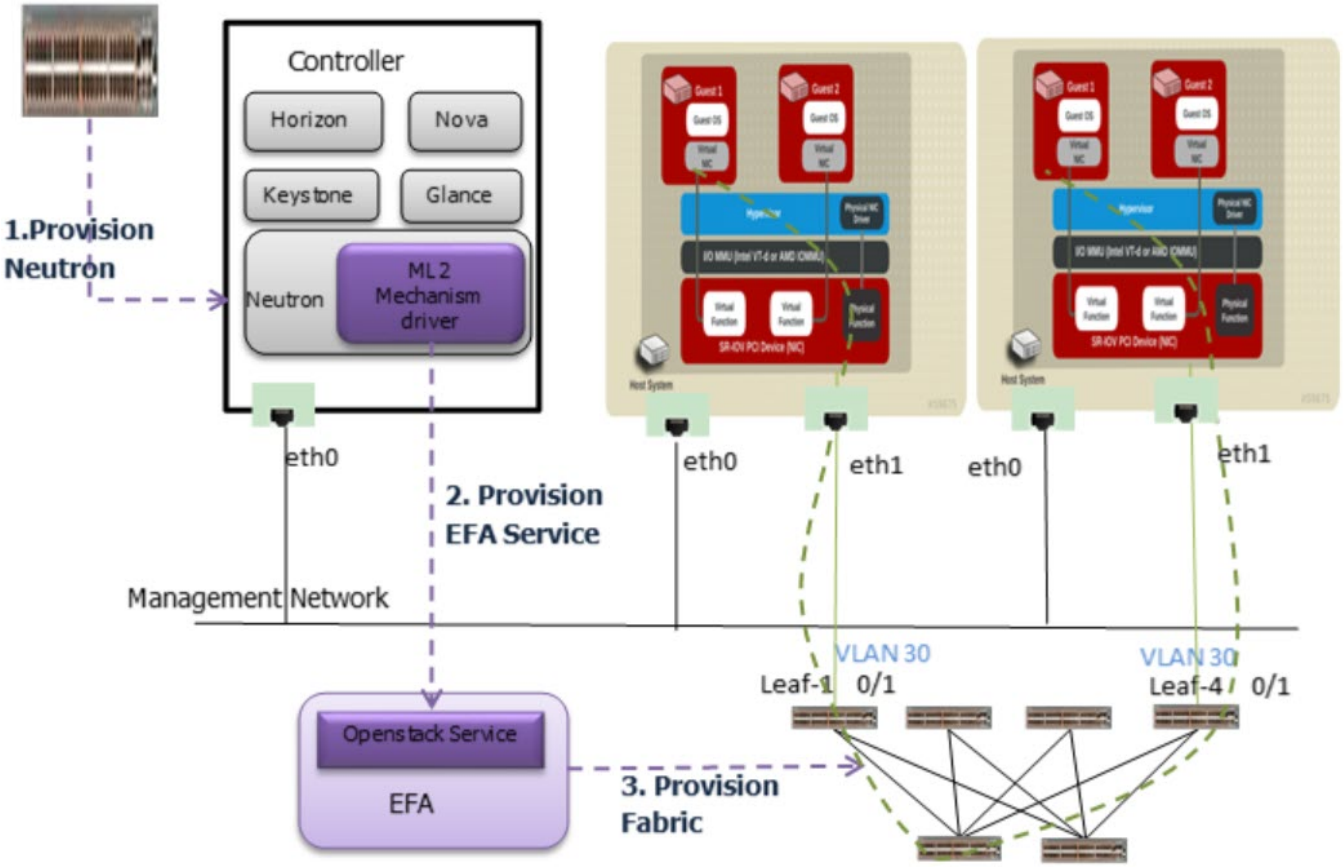


Figure 8: Mechanism Driver for SR-IOV

For more information on SR-IOV, refer to SR-IOV and Multi Segment Support on page 55.

L3 Service Plug-in

Extreme L3 Service Plug-in within Neutron initiates Neutron API calls for network management. OpenStack Service in EFA translates the Neutron network management calls to appropriate tenant API calls and provisions the fabric with appropriate L3 networking constructs.

L3 Service plug-in creates a distributed VRF instance on the IP fabric that corresponds to the Neutron router. VRF instances are created on the Leaf and Border Leaf devices.

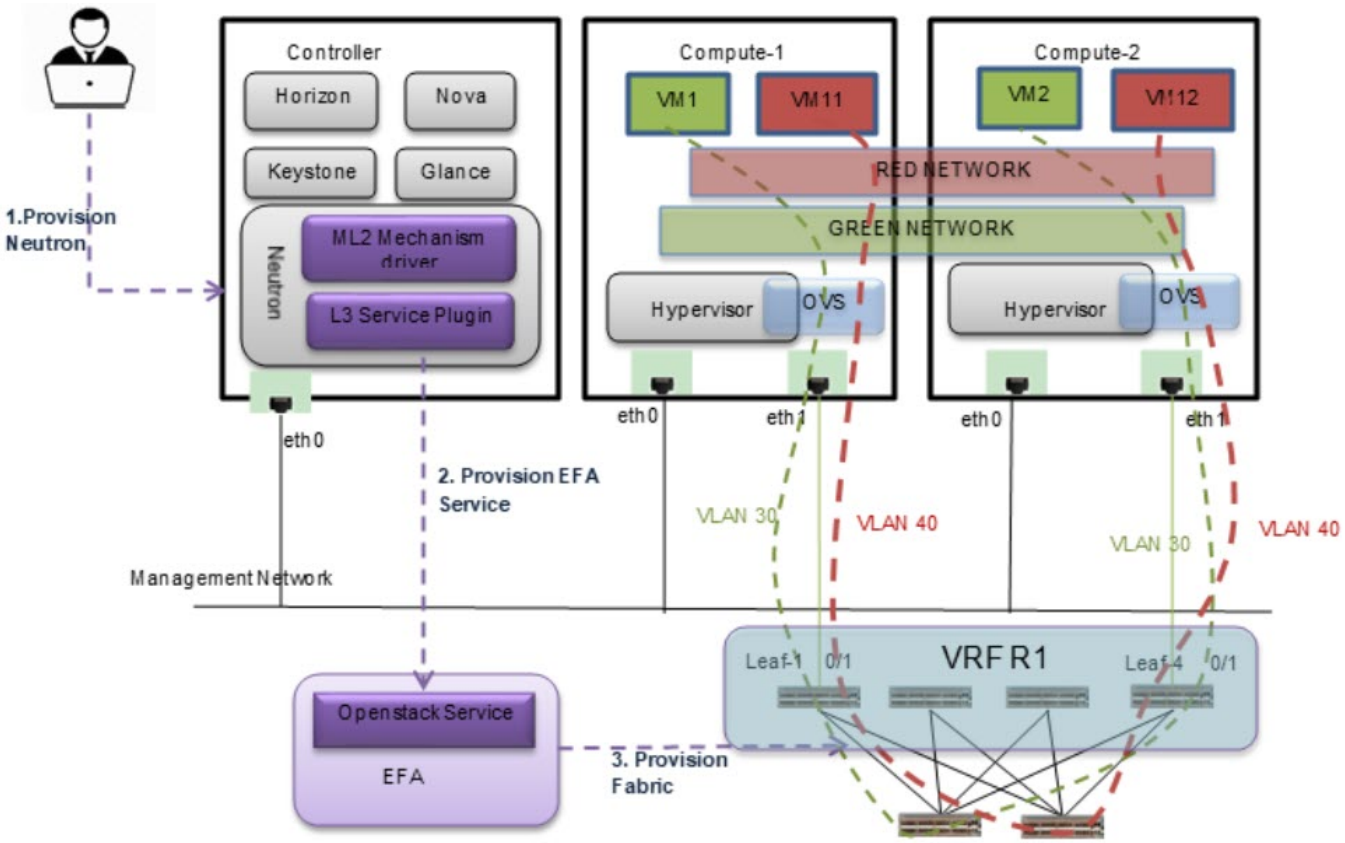


Figure 9: L3 Service plug-in

Configure L3 Service Plug-in

Procedure

Configure `extreme_l3_efa` in the `neutron.conf` file.

Note
Do not run `extreme_l3_efa` along with router service plug-in.

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

Create an OpenStack Network Using L3 Service Plug-in

About This Task

Extreme ML2 plug-in and L3 Service plug-in forward network and VM creation requests to EFA along with details about VLAN allocated by the Type Driver. OpenStack Service on EFA utilizes the End Point Group (EPG) construct of Tenant Services to provision an EPG.

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Create an OpenStack network, GREEN_NETWORK.

```
# openstack network create GREEN_NETWORK
```

3. Create a subnet, GREEN_SUBNET.

```
# openstack subnet create GREEN_SUBNET --network GREEN_NETWORK --subnet-range
10.0.0.0/24 \ --gateway 10.0.0.1
```

4. Create a Virtual Machine, VM1 on 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-1VM1
```

5. Create a Virtual Machine, VM2 on 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-2VM2
```

6. Create a second OpenStack network, RED_NETWORK.

```
# openstack network create RED_NETWORK openstack subnet create RED_SUBNET --network
RED_NETWORK --subnet-range 9.0.0.0/24 \ --gateway 9.0.0.1
```

7. Create a Virtual Machine, VM11 on 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-1VM11
```

8. Create a Virtual Machine, VM12 on 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-2VM12
```

9. Create a router, R1 and add both the two networking instances, GREEN_SUBNET and RED_SUBNET as part of the Router.

```
# neutron router-create R1
# neutron router-interface-add R1 GREEN_SUBNET
# neutron router-interface-add R1 RED_SUBNET
```

L3 Service Tenant Mapping

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

Table 6: L3 Service tenant mapping

VM	Neutron Network	Tenant Service (EPG)
VM1	GREEN_NETWORK (VLAN30) UUID =74cbf489-f3d9-41c7-bbb2-6cb7df33da6d	89cbf489-f3d9-41c7-bbb2-6cb7df33da02 • 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) UUID =74cbf489-f3d9-41c7-bbb2-6cb7df33da6d	

Table 6: L3 Service tenant mapping (continued)

VM	Neutron Network	Tenant Service (EPG)
VM11	RED_NETWORK (VLAN40) GREEN_NETWORK (VLAN30) UUID =89cbf489-f3d9-41c7-bbb2-6cb7df33da02	89cbf489-f3d9-41c7-bbb2-6cb7df33da02 • Endpoint: eth 0/1 on Leaf-1 • EndPoint: eth 0/1 on Leaf-2 • CTAG 40 • Anycast 9.0.0.1 • VRF R1 (UUID=99cbf489-f3d9-41c7-bbb2-6cb7df33da03)
VM12	RED_NETWORK (VLAN40) UUID =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 DB1, DB2, APP1, and APP2. The VRF configuration enables routing between the two networks.

L3 Service Switch Configuration

The following table shows an example of the switch configuration on the leaf nodes.

Table 7: L3 Service switch configuration

EPG	Switch Configuration	Comments

89cbf489-f3d9-41c7-bbb2-6cb7df33da02 (GREEN_NETWORK)
• Endpoint: eth 0/1 on Leaf-1
• EndPoint: eth 0/1 on Leaf-2
• CTAG 30
• AnyCast 10.0.0.1
• VRF R1
	`vlan 30`
	`description L2 Tenant vlan`
	`!`
		VLAN 30 configured on Leaf-1 and Leaf-4

	`evpn default`
	`vlan add 30`
	`!`
		VLAN 30 added to evpn
		Note: VLAN to VNI mapping is set to auto as part of fabric setup.

	`interface Ve 30`
	`  vrf forwarding 99cbf489f3d941c7bbb26cb7df33da03`
	`  ip anycast-address 10.0.0.1`
	`  no shutdown`
	`!`
		Anycast IP address is provisioned as the Gateway IP address on the Interface Ve 30 (GREEN_Network)

	`interface ethernet 0/1`
	`switchport`
	`switchport trunk`
	`allowed vlan add 30`
	`no shutdown`
	`!`
		Interface configurations on both the switches, Leaf-1 and Leaf-4.

Table 7: L3 Service switch configuration (continued)

EPG	Switch Configuration	Comments

89cbf489-f3d9-41c7-bbb2-6cb7df33da02 (GREEN_NETWORK)
• Endpoint: eth 0/1 on Leaf-1
• EndPoint: eth 0/1 on Leaf-2
• CTAG 30
• AnyCast 10.0.0.1
• VRF R1
	`vlan 40`
	`description L2 Tenant vlan`
	`router-interface Ve 40`
	`!`
		VLAN 40 configured on Leaf-1 & Leaf-4

	`evpn default`
	`vlan add 40`
	`!`
		VLAN 40 added to evpn.
		
		Note: VLAN to VNI mapping is set to auto as part of fabric setup.

	`interface Ve 40`
	`vrf forwarding 99cbf489f3d941c7bbb26cb7df33da03`
	`ip anycast-address 9.0.0.1`
	`no shutdown`
	`!`
		Anycast IP address is provisioned as the Gateway IP address on the Interface Ve 40 (RED_Network).

	`interface ethernet 0/1`
	`switchport`
	`switchport trunk allowed vlan add 40`
	`no shutdown`
	`!`
		Interface configurations on both the switches Leaf-1 and Leaf-4.

MCT Support

The following figure shows an overview of an MCT network.

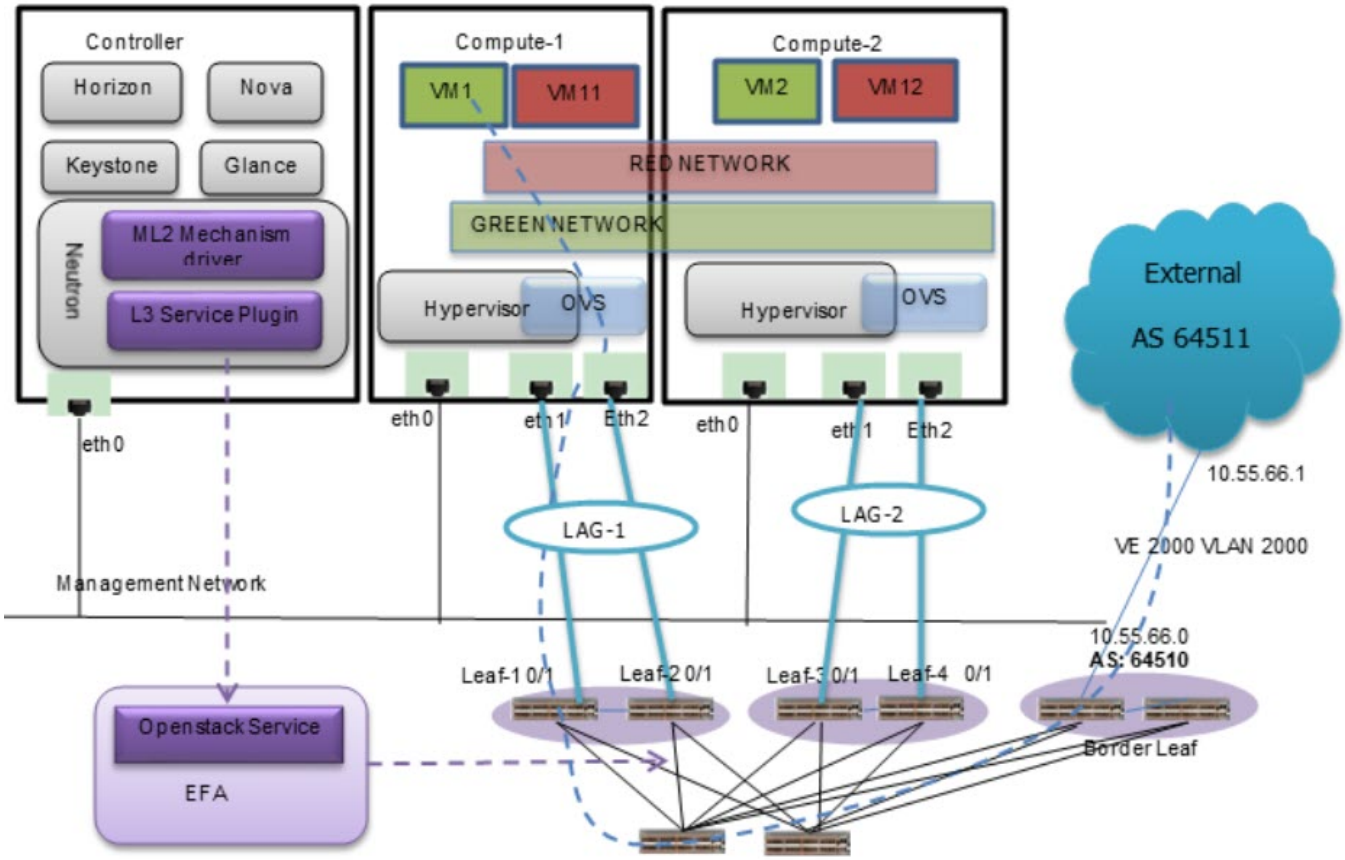


Figure 10: Overview of an MCT network

The following table shows that the bonds on the Compute Node that match with the Port-Channels (LAG) on the MCT Leaf nodes.

Compute	NIC	BOND/PO	Switch
Compute 1	eth1	Bond-1 PO-1	Leaf-1 eth0/1
	eth2		Leaf-2 eth0/1
Compute 2	eth1	Bond-2 PO-2	Leaf-3 eth0/1
	eth2		Leaf-4 eth0/1

MCT Tenant Mapping

The following table shows the tenant mapping of Port Channels formed using the EndPoints on the Leaf devices.

Compute	Port Channel	
Compute 1	PO-1	EndPoint eth 0/1 on Leaf-1
		EndPoint eth 0/1 on Leaf-2

Compute	Port Channel	
Compute 2	PO-1	EndPoint eth 0/1 on Leaf-3
		EndPoint eth 0/1 on Leaf-4

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

L2 Topology Examples

The following figure shows an L2 topology with a single NIC from the Compute Node towards switches in the IP Fabric.

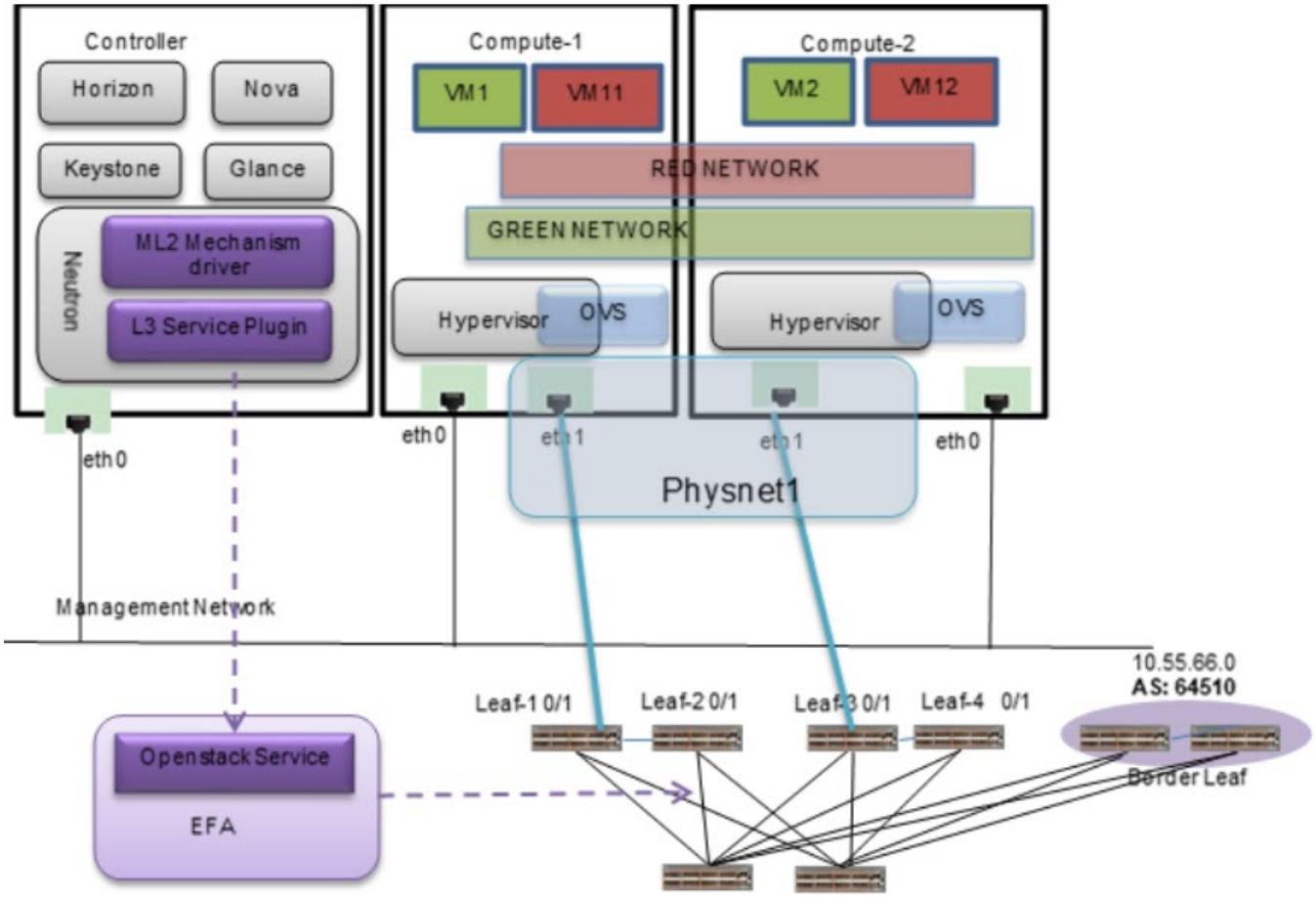


Figure 11: Single NIC towards the fabric (OVS)

Hardware	Configuration

Physnet Mapping	`[ml2_type_vlan]`
	`network_vlan_ranges = physnet1:100:500`
	
	`[ovs]`
	`bridge_mappings = physnet1:`**`bridge-ovs`**

OVS Bridge (on each Compute Node)	`ovs-vsctl add-br` **`bridge-ovs`** `ovs-vsctl add-port bridge-ovs` **`eth1`**

The following figure shows an L2 topology with multiple NICs from the Compute Node towards switches in the IP Fabric. This topology uses LAG towards the MCT pair of leaf switches.

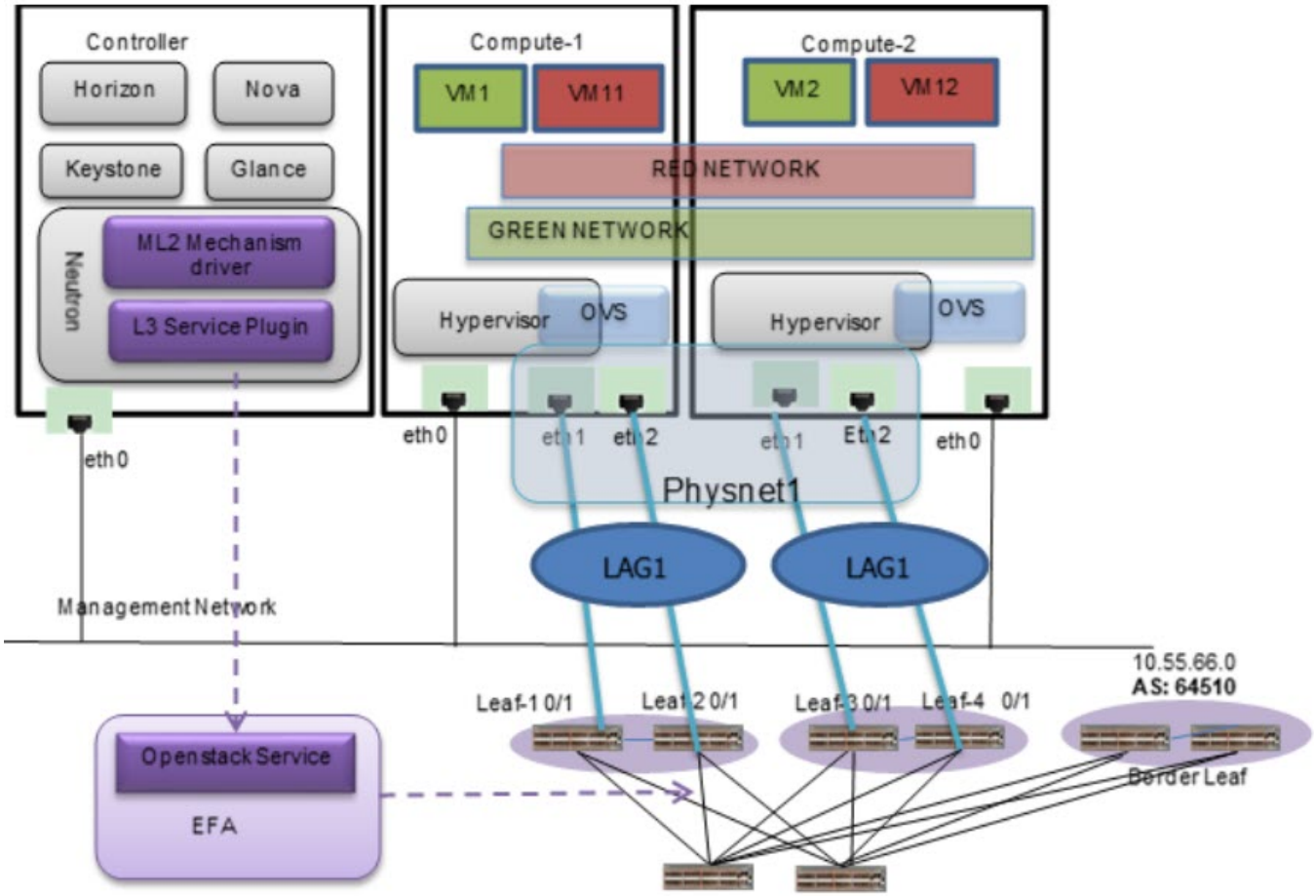


Figure 12: LAG or Bond towards the fabric (OVS)

Hardware	Configuration

Physnet Mapping	`[ml2_type_vlan]`
	`network_vlan_ranges = physnet1:100:500`
	
	`[ovs]`
	`bridge_mappings = physnet1:`**`bridge-ovs`**

OVS Bridge (on each Compute Node)	`ovs-vsctl add-br` **`bridge-ovs`** `sudoovs-vsctl add-bond` **`bridge-ovs bond0 eth1 eth2`** `lacp=active`

Multiple VIM/VPOD Instances

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

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

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

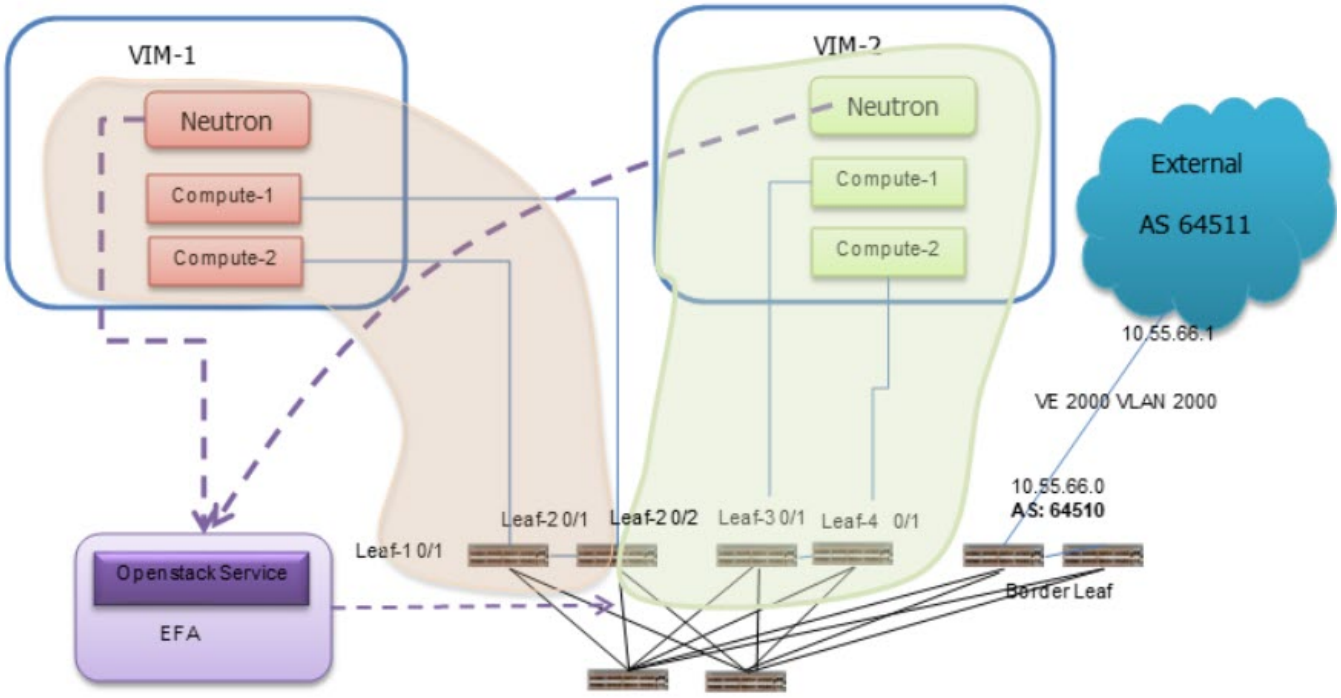


Figure 13: Multiple VIM or VPOD instances

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

The following example shows multiple VIM/VPOD instances.

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

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

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

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

Segregate VM Instance

EFA Tenant must be created using the same name as the `region_name` in the OpenStack.

Procedure

Segregate the VM instance in the `/etc/neutron/plugins/ml2/ml2_conf_extreme.ini` file.

```
[ml2_extreme]
efa_rest_token = extremenetworks_user_auth_key
efa_port = 80
efa_host = 10.24.51.170
region_name = VIM-1
[efa_topology]
efa_pn_mapping_file = /home/ubuntu/pn.csv
efa_link_mapping_file = /home/ubuntu/link.csv
```

Deploy Neutron in a Single Homed Leaf Network

About This Task

This procedure provides steps to deploy Neutron in a single homed leaf network with the following connections:

- Compute Host Compute-1: Network interface card eth1 connected to switch Leaf-1 (single homed leaf) port 0/1
- Compute Host Compute-2: Network interface card eth1 connected to switch Leaf-4 (single homed leaf) port 0/1

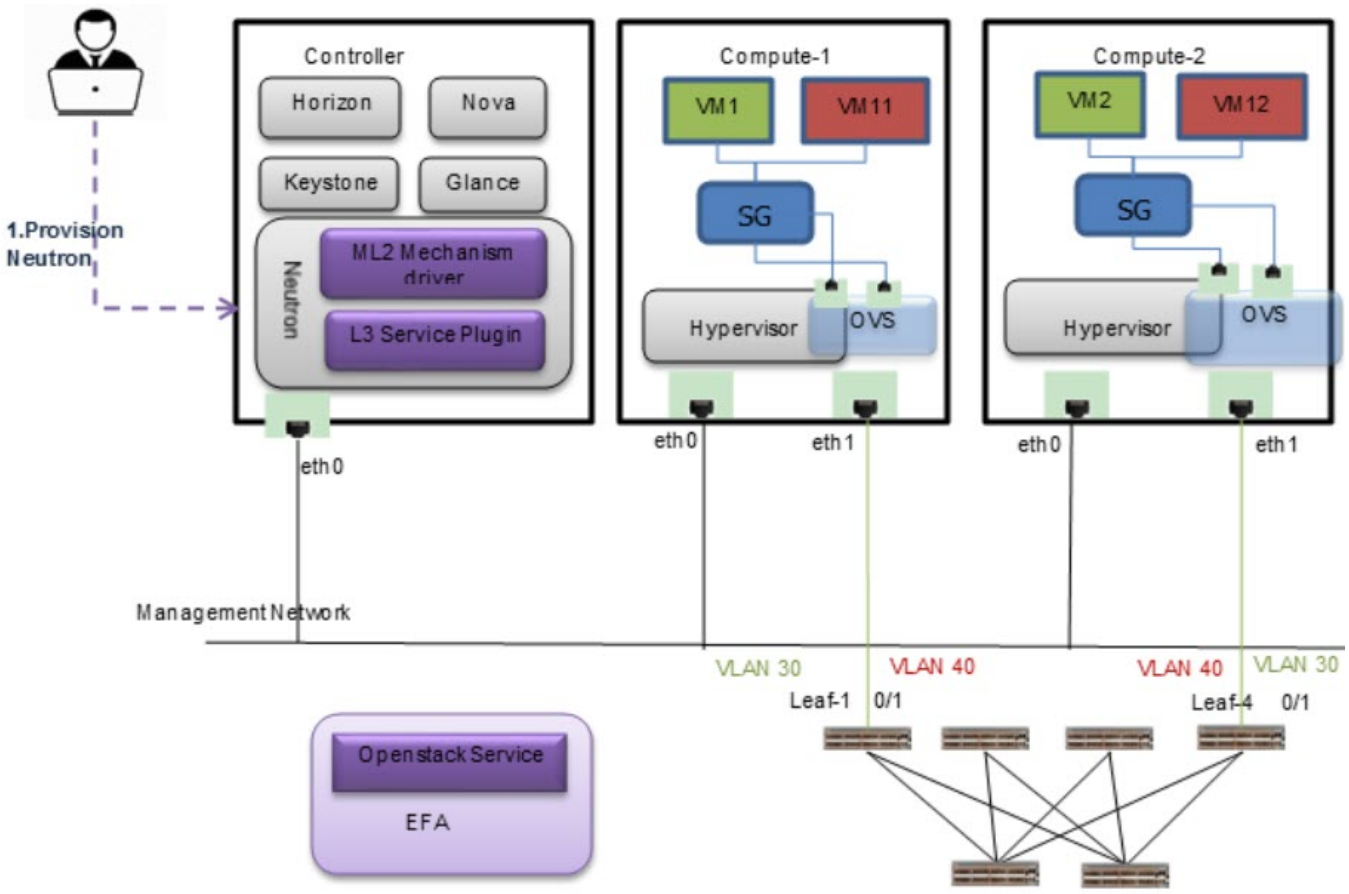


Figure 14: Compute node connecting to single homed leaf

Note

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

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Build the topology.

```
# sudo efa-pn-mapping add -H compute-1 -p physnet1 -n eth1
# sudo efa-pn-mapping add -H compute-2 -p physnet1 -n eth1
# sudo efa-link-mapping add -H compute-1 -n eth1 -s leaf-1-ip-address -p 0/1
# sudo efa-link-mapping add -H compute-2 -n eth1 -s leaf-4-ip-address -p 0/1
```

These configurations can be included in `pn.csv` and `link.csv` files for bulk topology setup.

3. Create the Neutron network and subnet.

```
# openstack network create net1 --provider-network-type vlan --provider-physical-
network physnet1 --provider-segment 30

# openstack subnet create subnet1 --network net1 --subnet-range 20.0.5.0/24
```

4. Create a Neutron port.

```
# openstack port create port0 --network net1
```

5. Create a VM.

```
# openstack server create --flavor m1.tiny --image cirros-0.3.5-x86_64-disk --nic port-id=port0 --availability-zone compute-1  instance-1
```

Deploy Neutron in an MCT Network

About This Task

This procedure provides steps to deploy Neutron in an MCT network with the following connections:

- Compute Host Compute-1: Network interface card eth1 and eth2 connected to switch Leaf-1 port 0/1 and switch Leaf-2 0/1(MCT pair)
- Compute Host Compute-2: Network interface card eth1 and eth2 connected to switch Leaf-3 port 0/1 and switch Leaf-4 0/1(MCT pair)

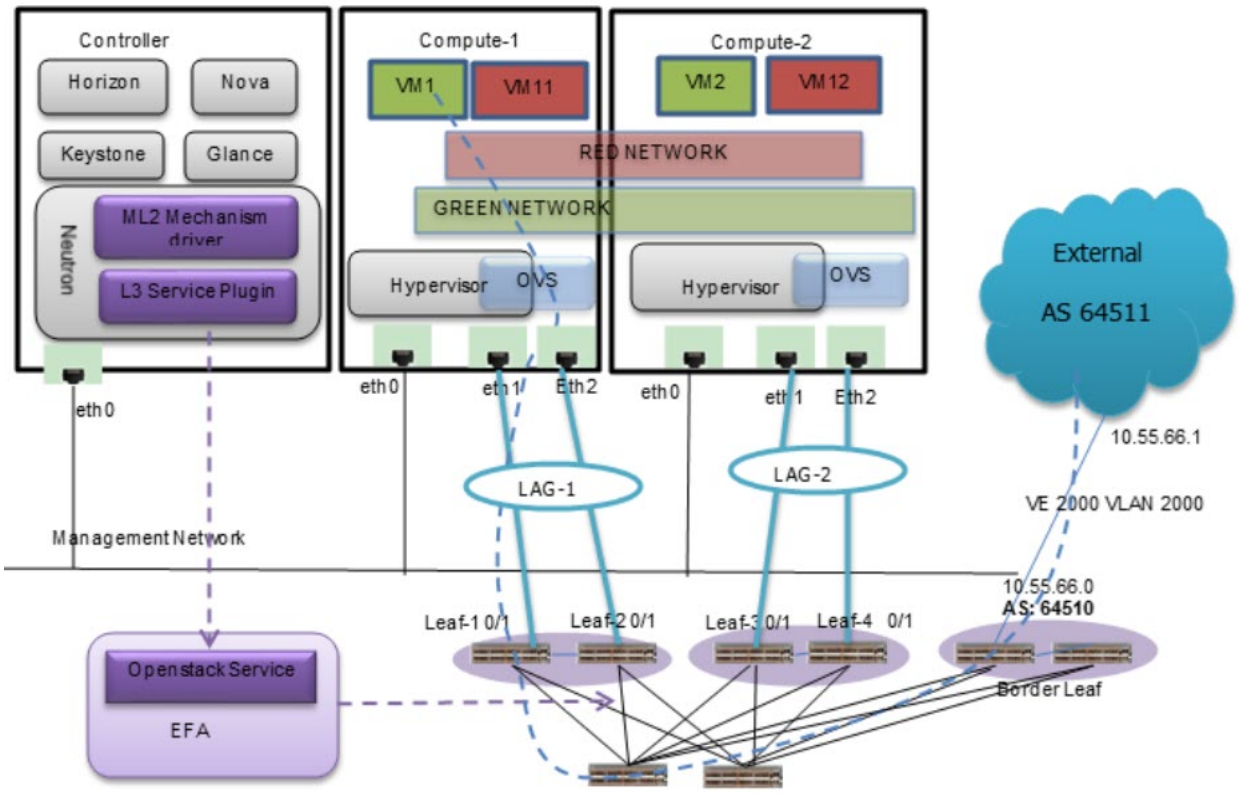


Figure 15: Compute nodes (CCEP) MCT connections

Note

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

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Add the ovs bond at compute-1 and compute-2 using 802.3 ad.

```
# sudo ovs-vsctl add-bond br0 bond0 eth1 eth2 lacp=active
```

3. Build the topology.

```
# sudo efa-pn-mapping add -H compute-1 -p physnet1 -n eth1
# sudo efa-pn-mapping add -H compute-1 -p physnet1 -n eth2
# sudo efa-pn-mapping add -H compute-2 -p physnet1 -n eth1
# sudo efa-pn-mapping add -H compute-2 -p physnet1 -n eth2

# sudo efa-link-mapping add -H compute-1 -n eth1 -s leaf-1-ip-address -p 0/1 -P
lag_compute1
# sudo efa-link-mapping add -H compute-1 -n eth2 -s leaf-2-ip-address -p 0/1 -P
lag_compute1
# sudo efa-link-mapping add -H compute-2 -n eth1 -s leaf-3-ip-address -p 0/1 -P
lag_compute2
# sudo efa-link-mapping add -H compute-2 -n eth2 -s leaf-4-ip-address -p 0/1 -P
lag_compute2
```

   These configurations can be included in `pn.csv` and `link.csv` files for bulk topology setup.

4. Create the Neutron network and subnet.

```
# openStack network create GREEN_NETWORK --provider-network-type vlan --provider-
physical-network physnet1 --provider-segment 2000

# openstack subnet create subnet1 --network GREEN_NETWORK --subnet-range 20.0.5.0/24
```

5. Create a Neutron port.

```
# openstack port create port0 --network GREEN_NETWORK
```

6. Create a VM.

```
# openstack server create --flavor m1.tiny --image cirros-0.3.5-x86_64-disk --nic port-
id=port0 --availability-zone compute-1 instance-1
```

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

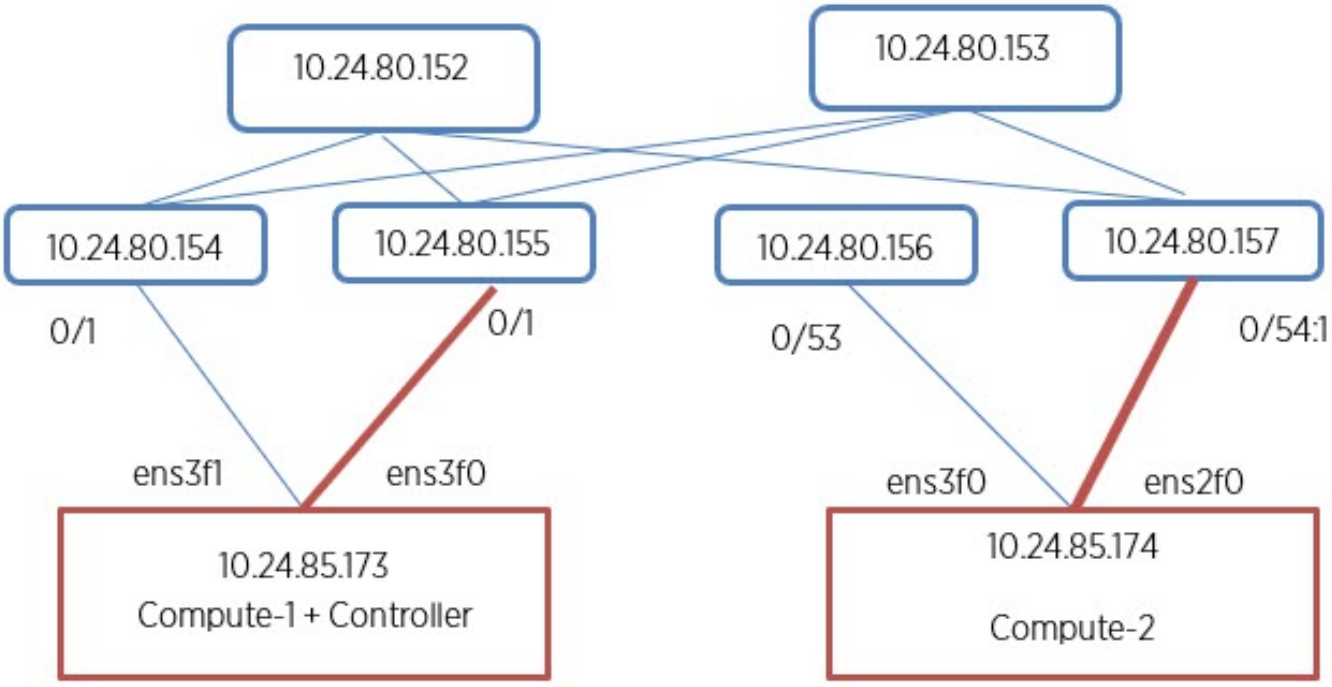


Figure 16: SR-IOV network (VF passthrough)

Create PCI Passthrough Whitelist

Procedure

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

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

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

Configure SR-IOV Agent

Procedure

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

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

2. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

3. 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
```

4. Repeat the procedure for Compute node 2.

Configure Nova Scheduler

Procedure

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

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

2. Restart the Nova Scheduler.

Configure Mechanism Drivers for SR-IOV

Procedure

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

```
[ml2]
tenant_network_types = vlan
type_drivers = vlan
mechanism_drivers = openvswitch, sriovnicswitch, extreme_efa
```

```
[ml2_type_vlan]
network_vlan_ranges = physnet1:2:500
```

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

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

3. Restart the Neutron.

Create Network for VF-PT

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. 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

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. 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)
```

Create Virtual Machines

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. 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
```

3. Repeat the procedure for Compute node 2.

Create SR-IOV Direct Ports

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. 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)
```

3. Repeat the procedure for the second port.

Create SR-IOV Direct-Physical Port

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. 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)
```

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

Delete SR-IOV Entities

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. 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

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

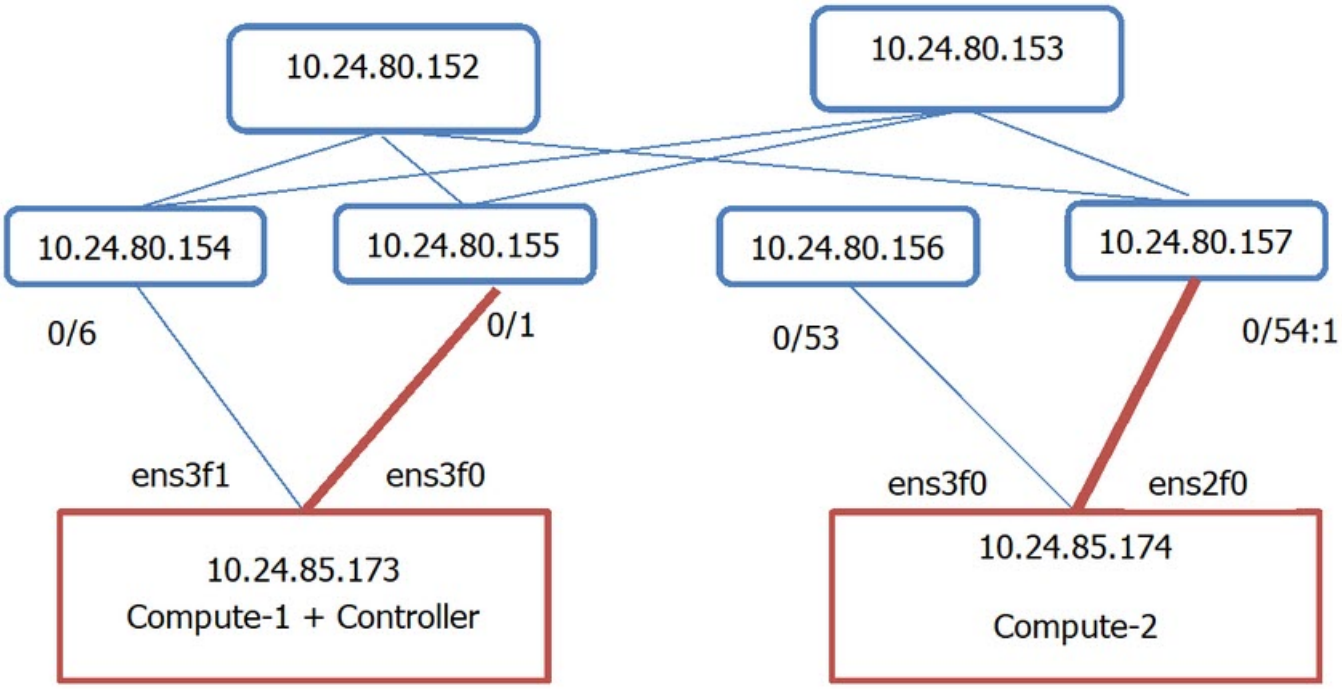


Figure 17: Overview of multi segment network

Configure Segments

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

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Configure the segment network.

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

Rename Network Segment

Network must be formed with SR-IOV provider network.

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Rename the network segment.

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

Create Network Segment

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Create a second segment.

```
# openstack network segment create --physical-network physnet1 \ --network-type vlan --segment 2016 --network multisegment dhcp_segment
```

Create Subnet on Second Segment

Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. 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

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. Create a port on SR-IOV segment.

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

Create a VM Using the Port

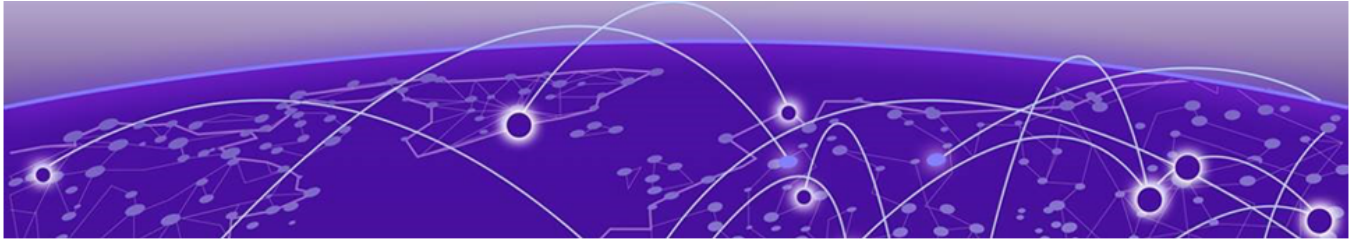
Procedure

1. Log in to the network node using SSH and go to the EFA folder.

```
# cd efa
```

2. 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
```



Appendix: OpenStack Service Command Reference

efa openstack debug

Displays OpenStack debug information.

Syntax

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

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

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

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

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

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

Parameters

cleanup *--name*

Cleans up all OpenStack assets associated to a tenant

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

Deletes the selected network element

network

Specifies the name of the network

network-interface

Specifies the name of the network-interface

tenant

Specifies the name of the tenant

router

Specifies the name of the router

router-interface

Specifies the name of the router-interface

efa openstack execution

Provides OpenStack execution commands.

Syntax

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

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

Parameters

delete

Deletes execution entries older than the specified days.

--days int32

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

--id

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

--help

Provides help for execution.

--limit int32

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

show

Lists all Networks and its summary information.

--status

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

efa openstack network show

Displays OpenStack network information.

Syntax

efa openstack network show

Parameters

network

Lists all networks

Examples

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

efa openstack network-interface show

Displays OpenStack network-interface information.

Syntax

efa openstack network-interface show

Parameters

network-interface

Lists all network-interfaces

Examples

```
# efa openstack network-interface show

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

efa openstack router show

Displays OpenStack router information.

Syntax

efa openstack router show

Parameters

Router

Lists all routers

Examples

```
# efa openstack router show

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

efa openstack router-interface show

Displays OpenStack router-interface information.

Syntax

efa openstack router-interface show

Parameters

router-interface

Lists all router-interfaces

Examples

```
# efa openstack router-interface show

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

efa openstack subnet show

Displays OpenStack subnet information.

Syntax

efa openstack subnet show

Parameters

subnet

Lists all subnets

Examples

```
# efa openstack subnet show

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

Appendix: Neutron REST Endpoints

Neutron REST Endpoints on page 70

Neutron REST Endpoints

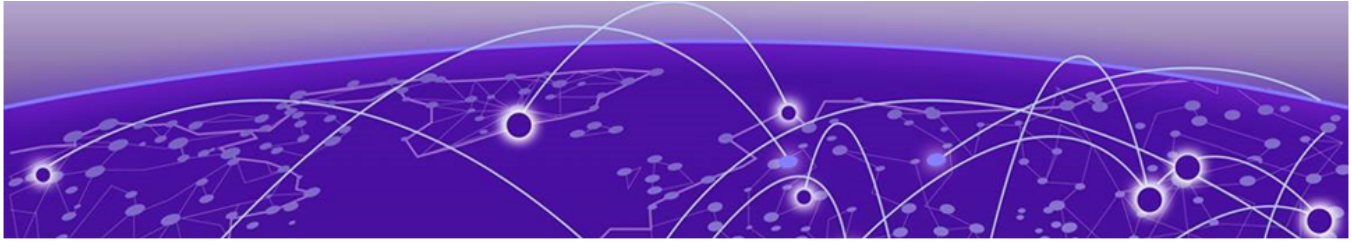
The following table lists the API handled Extreme M12 drivers. For more information on APIs, refer to OpenStack Networking API Guide.

API	Description
Network API	
`/v2.0/networks/{network_id}`	Show network details
`/v2.0/networks/{network_id}`	Update network
`/v2.0/networks/{network_id}`	Delete network
`/v2.0/networks`	List networks
`/v2.0/networks`	Create network
Port API	
`/v2.0/ports`	Create port
`/v2.0/ports/{port_id}`	Update port
`/v2.0/ports/{port_id}`	Show port details
`/v2.0/ports/{port_id}`	Delete port
`/v2.0/ports`	List ports
L3 API	

`POST /v2.0/routers/{router_id}`
`DELETE /v2.0/routers/{router_id}`
`PUT /v2.0/routers/{router_id}/ add_router_interface [Added router interface]`
`PUT /v2.0/routers/{router_id}/ remove_router_interface [Remove router interface]`
`POST /v2.0/subnets`
`DELETE /v2.0/subnets/{subnet_id}`	

Topology Extension API	
`/v2.0/efa_topologies`	Create Link

API	Description
/v2.0/efa_topologies{link_id}	Show link details
/v2.0/efa_topologies{link_id}	Delete link
/v2.0/efa_topologies	List Links
/v2.0/efa_nic_mappings	Create provider network mapping
/v2.0/efa_nic_mappings {nic_id}	Show nic details
/v2.0/ efa_nic_mappings {nic_id}	Delete nic
/v2.0/ efa_nic_mappings	List nics



Appendix: Event Logging

Event Logging

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

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