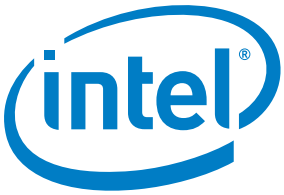


Enabling Intel® Omni-Path in OpenStack

Application Note

March 2020

In 2020, Cornelis Networks acquired the Omni-Path business of Intel Corporation, including the Intel® Omni-Path and Intel® True Scale products.

Cornelis Networks is now the manufacturer of these products, renamed Omni-Path and True Scale.

As Cornelis Networks creates and transitions to its own written materials for the products, you can reference the attached legacy materials.

All questions and product support requests should be directed to support@cornelisnetworks.com.

Document Number: 616682-1.2

Legal Disclaimer

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Contents

Figures

Tables

Revision History

Date	Revision	Description
March 2020	1.2	Updated MAC address information and changed image version to CentOS-8.
February 2020	1.1	Minor update.
December 2019	1.0	Initial release.

§

1 Introduction

This document describes how to set up Intel® Omni-Path Architecture (Intel® OPA) network fabrics with OpenStack*.

Note: Chapter 6, Example OpenStack Installation, provides an example OpenStack installation using DevStack as Proof-of-Concept and is not recommended for production system installation. Please refer to the *OpenStack Installation Guide* (https://docs.openstack.org/install-guide/) for setting up a production environment.

Information about the general installation of OpenStack is out of scope for this document. An introduction can be found at:

https://www.openstack.org/software/start/

Intel® Omni-Path publications are available at the following URL, under *Latest Release Library*:

https://www.intel.com/content/www/us/en/design/products-and-solutions/networking-and-io/fabric-products/omni-path/downloads.html

1.1 Terminology

The table below defines some of the uncommon acronyms found in this document.

Table 1. Terminology

Term	Description
BMC	Baseboard Management Controller
DHCP	Dynamic Host Configuration Protocol
FM	Intel® Omni-Path Fabric Suite Fabric Manager
HFI	Host Fabric Interface
HPC	High performance computing
IHV	Independent Hardware Vendor
IPMI	Intelligent Platform Management Interface
NIC	Network Interface Card
ODM	Original Design Manufacturer

Term	Description
OEM	Original Equipment Manufacturer
OPA	Intel® Omni-Path Architecture (Intel® OPA)
pip	Package Manager for Python Packages
PXE	Preboot Execution Environment
VLAN	Virtual Local Access Network (LAN)

2 Overview

2.1 Using Intel® Omni-Path Architecture for Multi-Tenant Security with OpenStack

OpenStack is a popular set of management software for administering Linux* clusters. Versions of OpenStack are now available from various Linux distributions. As an open source project, users can download and contribute to OpenStack.

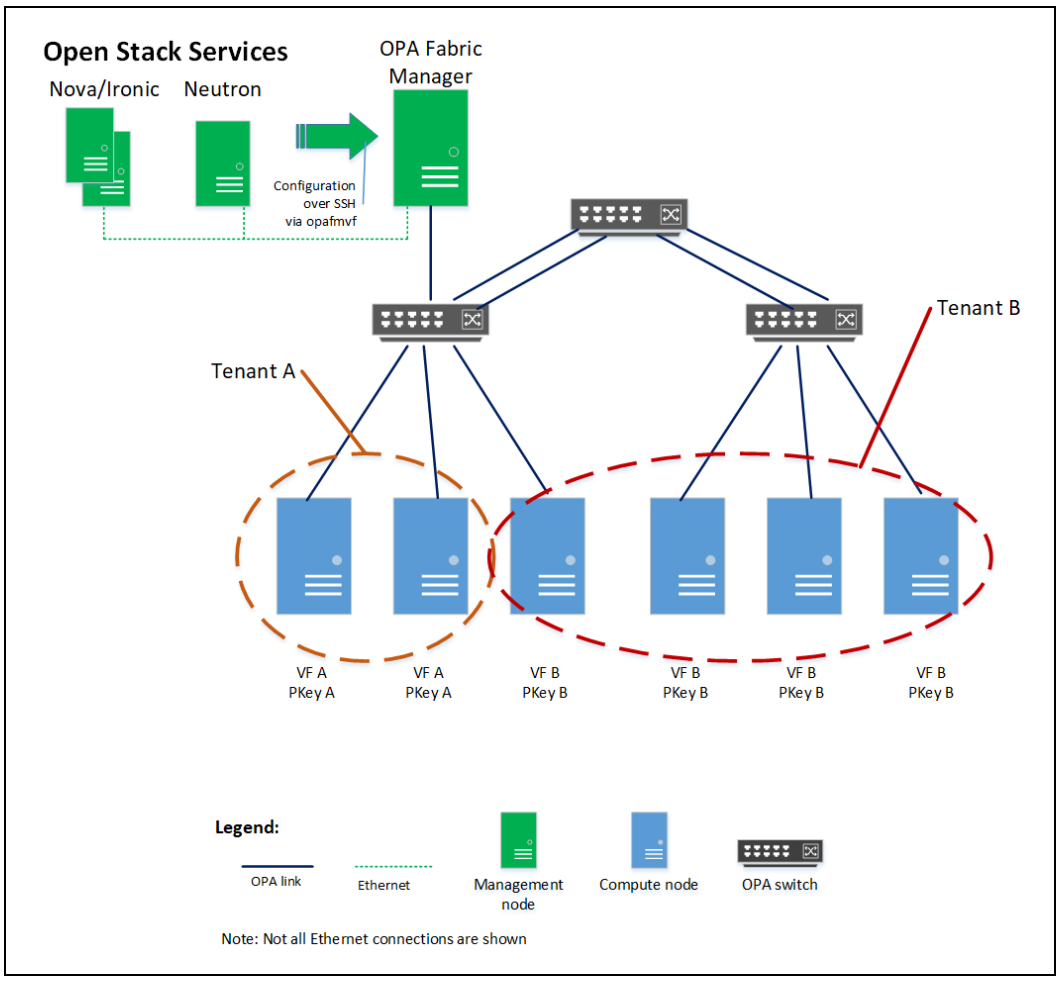
Intel® Omni-Path Architecture is a high performance fabric optimized for low latency and high bandwidth applications, especially in High Performance Computing (HPC) and AI Deep Learning training. A wide ecosystem of OEMs, ODMs, and IHVs provide PCIe adapters, servers, switches, cables, and storage solutions optimized and integrated with Intel® Omni-Path.

OpenStack provides mechanisms to automate administration of multi-tenant networks for cloud providers. Recent developments for OpenStack can permit OpenStack to administer multi-tenant Intel® Omni-Path clusters.

Intel® Omni-Path provides support for multi-tenant environments through its Virtual Fabrics (vFabrics) feature (refer to the *Intel® Omni-Path Fabric Suite Fabric Manager User Guide*). vFabrics bring to the Intel® Omni-Path Fabric many of the capabilities of Ethernet VLANs and Fibre Channel Zoning.

Using vFabrics, the administrator can slice up the physical fabric into many overlapping virtual fabrics as shown in Figure 1. The goal of vFabrics is to permit multiple tenants or applications to be run on the same fabric at the same time with limited interference. The administrator can control the degree of isolation. Each vFabric can be assigned Quality of Service (QoS) and security policies to control how common resources in the fabric are shared among vFabrics.

Figure 1. Network Overview Using VFabrics



For multi-tenant environments, the most important set of features are those related to security. Security within Intel® Omni-Path is implemented via PKeys (very similar to Ethernet VLANs). As shown in Figure 1, PKeys can restrict which nodes may talk to each other. The PKeys are configured within the switches and HFIs by the centralized Fabric Manager (FM) and security is enforced by the switches.

One goal of security is to prevent would-be attackers from destabilizing or compromising the integrity of the fabric by utilizing a compromised node within the fabric. Intel® Omni-Path provides a higher level of security than InfiniBand* through a number of innovative features including:

- Unlike InfiniBand, Intel® Omni-Path secures the subnet management communication paths at the switches so that only authorized nodes may initiate subnet management actions. This security is configured at the switches so that an unauthorized node cannot attempt to run a fake FM to take over the fabric. This method provides greater security than the InfiniBand 32-bit non-cryptographic MKey that most networks either do not implement or tend to use the same value for all nodes.

- In addition, Intel® Omni-Path offers subnet management denial of service protection through hardware enforced rate limiting for subnet management packets. This protection occurs at the switches.
- The switches and FM also provide anti-spoofing features that prevent nodes from spoofing the address (GUID or LID) of another node as well as prevent hosts from pretending to be switches. This mechanism helps to ensure that the proper nodes are assigned to each Virtual Fabric.
- The FM also provides configuration consistency checking between redundant FMs. This can prevent administrator mistakes that might lead to unexpected changes in fabric security or configuration.
- The FM limits access to information, such that non-authorized nodes cannot obtain information about other nodes in other tenants.
- Finally, the FM monitors hardware error counters and other fabric events and can quarantine suspicious nodes from the fabric. Quarantining can take down the nodes access to the fabric and prevent future attack attempts from the node.

2.2 Workflow Overview

To enable Intel® OPA network fabrics in OpenStack, use Intel's networking-omnipath (https://opendev.org/x/networking-omnipath.git) project on OpenDev. This project applies a mechanism driver in Neutron to enable the Intel® Omni-Path backend. Neutron provides the networking resources in OpenStack.

Provisioning an Intel® OPA high performance computing (HPC) node in OpenStack is accomplished using the Bare Metal service, Ironic*.

The following steps provide a detailed workflow for how an Intel® OPA (hereafter referred to as OPA) node is provisioned in a multi-tenant environment in OpenStack. Note that these steps are similar to those used to manage an Ethernet multi-tenant network with OpenStack; the main difference being the use of the networking-omnipath Neutron ML2 driver to configure OPA vFabrics via the OPA FM.

1. The user registers an OPA node in Ironic as a Bare Metal node.

   This includes details such as Baseboard Management Controller (BMC) address, username, and password used for communicating to the node.

2. The user creates two Bare Metal ports for the Bare Metal node registered in Step 1.

   These ports represents a physical interface attached to the node.

   – Ethernet PXE port: This port is used for Preboot Execution Environment (PXE) booting the node.

     The MAC address, physical network, and other parameters are required to create the port in Ironic.

   – OPA port: This port represents an OPA HFI interface.

You must provide a MAC address and client id details during creation. The client id is a 20-byte id in the format `<12-byte vendor prefix>:<8 byte port GUID>`.

3. The user creates a multi-tenant network (for example, opa-network) in Neutron.

   Neutron calls the *networking-omnipath* driver to create a virtual fabric (VF) in the Intel® Omni-Path Fabric Suite Fabric Manager.

4. The user sends a request to Nova to boot the OPA node.

   Nova, the OpenStack Compute service, creates virtual interfaces on the *opa-network* and sends a provision request to Ironic.

5. Ironic performs PXE booting of the deploy image to install the guest operating system (OS image).

   Once completed, the Bare Metal ports are bound to the virtual interfaces in Neutron.

6. Ironic sends a port binding request to Neutron and Neutron sends the request to the OPA Fabric Manager to assign the OPA port to the Virtual Fabric.

7. A power-on action is performed on the node. The guest image is booted and will be able to use the OPA fabric.

3 Quick Start

Note: This section assumes that you already have OpenStack installed. If you need to install OpenStack for testing purposes, refer to Chapter 6, Example OpenStack Installation. Additional information can be found at https://docs.openstack.org/train/install/. If you are using a distro-supplied OpenStack version, consult your OS distributor for further information.

Enabling Intel® Omni-Path with OpenStack requires the `networking-omnipath` ML2 plugin—an ML2 mechanism driver that integrates OpenStack Neutron API with the Intel® Omni-Path backend. It enables the Intel® Omni-Path fabric in an OpenStack cloud. Note that a network in the OpenStack networking realm corresponds to a virtual fabric on the Intel® Omni-Path side.

Note: Be sure to enable passwordless SSH login for root user to the OPA FM node for the OpenStack controller node. For networking-omnipath to communicate to the OPA FM node, add the OpenStack controller node's public ssh key to the FM node authorized key list.

Note: To secure your OPA network, disable the virtual fabric named "Default" in the OPA FM config file's VirtualFabrics section. OpenStack will create the desired per-tenant virtual fabrics.

Complete chapters 3 through 5 in sequence unless directed to go to another section.

3.1 Installing networking-omnipath

`networking-omnipath` can be installed from source, python package or DevStack.

Note: DevStack is used for constructing an OpenStack environment for development and testing purposes.

Use one of the following methods to install `networking-omnipath`.

3.1.1 From Source Code

1. Execute the following commands:

```
$ git clone https://opendev.org/x/networking-omnipath.git
$ cd networking-omnipath
$ sudo python setup.py install
```

2. Go to Configuring networking-omnipath.

3.1.2 Using pip – Package Manager for Python Packages

1. Execute the following commands:

```
$ sudo pip install networking-omnipath
```

2. Go to Configuring networking-omnipath.

3.1.3 Using DevStack

1. Add the following lines to `local.conf`.

   Replace the details below with your OPA FM node's information. This describes to OpenStack how to ssh into the FM node.

```
# Enable networking-omnipath plugin
enable_plugin networking-omnipath https://opendev.org/x/networking-
omnipath.git

# ml2_conf.ini config for omnipath mechanism driver
[[post-config|/etc/neutron/plugins/ml2/ml2_conf.ini]]
[ml2_omnipath]
username = "root"
ssh_key = <PATH_TO_SSH_PRIVATEKEY_OF_CONTROLLER_NODE>
ip_address = <IPV4_IP_of_OPA_FM>
```

2. Run `stack.sh`.

Note: Devstack performs the networking-omnipath configuration automatically and restarts the neutron-server service.

3. Go to Creating a Guest Image.

3.2 Configuring networking-omnipath

When you install the `networking-omnipath` plugin from source and python package, you need to configure Neutron to enable it as an ML2 plugin.

Perform the following steps:

1. Configure Openstack neutron server to enable networking-omnipath as an ML2 driver.

   Edit the `/etc/neutron/neutron.conf` file to enable the ML2 core plug-in:

```
[DEFAULT]
core_plugin = neutron.plugins.ml2.plugin.Ml2Plugin
```

2. Configure the ML2 plug-in.

Edit the `/etc/neutron/plugins/ml2/ml2_conf.ini` file to configure the omnipath mechanism driver. Append the following configuration values to the existing values in the config file.

```
[ml2]
mechanism_drivers = omnipath_mech
type_drivers = local,flat,vlan,vxlan
tenant_network_types = vlan

[ml2_type_flat]
flat_networks = *
```

Note: `networking-omnipath` will use vlan type driver by default.

3. Configure the VLAN range.

   Edit the `/etc/neutron/plugins/ml2/ml2_conf.ini` file:

   ```
   [ml2_type_vlan]
   network_vlan_ranges = PHYSICAL_NET:2:2000
   ```

Note: `PHYSICAL_NET` is the value used for `provider:physical_network` in tenant networks.

4. Configure ML2 Omnipath.

   Edit the `/etc/neutron/plugins/ml2/ml2_conf.ini` file. Replace the details below with your OPA FM node' s information. This describes to openstack how to ssh into the FM node.

   ```
   [ml2_omnipath]
   username = "root"
   ssh_key = <PATH_TO_SSH_PRIVATEKEY_OF_CONTROLLER_NODE>
   ip_address = <IPV4_IP_of_OPA_FM>
   ```

5. Restart the `neutron-server` service.

4 Creating a Guest Image

A guest image must be provided to OpenStack for the tenant nodes.

OPA hardware support is currently available in three distributions: Red Hat, SUSE, and CentOS. These images can be deployed unmodified from the distributor to support an Omni-Path network; however, it is expected that many users will require customizations to their guest image.

The examples provided in this document were performed with a CentOS-8 Cloud image. You can download the latest image from:

https://cloud.centos.org/centos/8/

Note: Also required is the ib_ipoib module.

5 Deploying a Node in an OPA Fabric

Once `networking-omnipath` is installed and configured in OpenStack, you can deploy OPA nodes using OpenStack.

5.1 Setting Up a VLAN Network for Intel® Omni-Path

1. Create the VLAN network type (for example, `opa-network`) to be used for Intel® Omni-Path cards:

```
$ source /opt/stack/devstack/openrc admin admin
$ openstack network create opa-network \
      --provider-physical-network opa --provider-network-type vlan
```

Output:

```
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2019-07-29T16:54:38Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | c017d599-9904-42dd-bd7c-23d6dfdac4f1 |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | None                                 |
| is_vlan_transparent       | None                                 |
| mtu                       | 1500                                 |
| name                      | opa-network                          |
| port_security_enabled     | True                                 |
| project_id                | 2b4b0f8af268435781a36da154ae68cc     |
| provider:network_type     | vlan                                 |
| provider:physical_network | opa                                  |
| provider:segmentation_id  | 55                                   |
| qos_policy_id             | None                                 |
| revision_number           | 2                                    |
| router:external           | Internal                             |
| segments                  | None                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      |                                      |
| updated_at                | 2019-07-29T16:55:10Z                 |
+---------------------------+--------------------------------------+
```

This command also creates a virtual fabric on the FM node. Note that throughout this example, we use `opa-network`. However, you may chose a different name.

2. Run `opareport` on the FM node to view the newly created VF:

```
$ opareport -o vfinfo -d 3
```

Output:

```
Getting All Node Records...
Done Getting All Node Records
Done Getting All Link Records
Done Getting All Cable Info Records
Done Getting All SM Info Records
Done Getting vFabric Records
Getting All Port VL Tables...
Done Getting All Port VL Tables

vFabrics:
vFabric Index: 0   Name: Admin
PKey: 0x7fff   SL: 0  Select: 0x3: PKEY SL   PktLifeTimeMult: 1
MaxMtu: unlimited  MaxRate: unlimited   Options: 0x03: Security QoS
QOS: Bandwidth:  50%  PreemptionRank: 0  HoQLife:    8 ms
     NodeGUID          Port Type Name
     0x001175010165acd1    1 FI   phwtstl006 hfi1_0
     0x001175010165b15b    1 FI   phwtstl008 hfi1_0
     0x001175010165b22c    1 FI   phwtstl006 hfi1_1
     0x001175010265bb30    0 SW   OmniPth00117501ff65bb30
     0x001175010265bd26    0 SW   OmniPth00117501ff65bd26

vFabric Index: 1   Name: c017d599-9904-42dd-bd7c-23d6dfdac4f1
PKey: 0x53   SL: 1  Select: 0x3: PKEY SL   PktLifeTimeMult: 1
MaxMtu: unlimited  MaxRate: unlimited   Options: 0x03: Security QoS
QOS: Bandwidth:  50%  PreemptionRank: 0  HoQLife:    8 ms
     NodeGUID          Port Type Name

2 VFs
------------------------------------------------------------------------------
```

The newly created `opa-network` shows as a Virtual Fabric on the FM with Name corresponding to the network's uuid, c017d599-9904-42dd-bd7c-23d6dfdac4f1

3. Create the subnet for the `opa-network`:

```
$ openstack subnet create --subnet-range 192.168.1.0/24 \
      --network opa-network opa-subnet
```

Output:

```
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| allocation_pools  | 192.168.1.2-192.168.1.254            |
| cidr              | 192.168.1.0/24                       |
| created_at        | 2019-07-29T16:55:10Z                 |
| description       |                                      |
| dns_nameservers   |                                      |
| enable_dhcp       | True                                 |
| gateway_ip        | 192.168.1.1                          |
| host_routes       |                                      |
| id                | 878a4e4d-b1e7-4723-ab21-d67d24e4c133 |
| ip_version        | 4                                    |
| ipv6_address_mode | None                                 |
```

```
| ipv6_ra_mode      | None                                 |
| name              | opa-subnet                           |
| network_id        | c017d599-9904-42dd-bd7c-23d6dfdac4f1 |
| project_id        | 2b4b0f8af268435781a36da154ae68cc     |
| revision_number   | 0                                    |
| segment_id        | None                                 |
| service_types     |                                      |
| subnetpool_id     | None                                 |
| tags              |                                      |
| updated_at        | 2019-07-29T16:55:10Z                 |
+-------------------+--------------------------------------+
```

The subnet-range specified will be the range of IPoIB addresses used on the OPA fabric.

5.2 Setting Up a FLAT Tenant Network for OPA Nodes

1. Create the Flat tenant network (for example, `provision`) to be used to launch the instances:

```
$ openstack network create provision \
     --provider-network-type flat --provider-physical-network public
```

Output:

```
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2019-07-29T16:58:38Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | 5c179c83-3d38-4d17-aa37-0991d473b244 |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | None                                 |
| is_vlan_transparent       | None                                 |
| location                  | Munch({'project': Munch({'domain_id':
'default', 'id': u'2b4b0f8af268435781a36da154ae68cc ', 'name':
'admin', 'domain_name': None}), 'cloud': '', 'region_name':
'RegionOne', 'zone': None})                                      |
| mtu                       | 1500                                 |
| name                      | provision                            |
| port_security_enabled     | True                                 |
| project_id                | 2b4b0f8af268435781a36da154ae68cc     |
| provider:network_type     | flat                                 |
| provider:physical_network | public                               |
| provider:segmentation_id  | None                                 |
| qos_policy_id             | None                                 |
| revision_number           | 2                                    |
| router:external           | Internal                             |
| segments                  | None                                 |
| shared                    | False                                |
```

```
| status                    | ACTIVE                                   |
| subnets                   |                                          |
| tags                      |                                          |
| updated_at                | 2019-07-29T16:59:10Z                     |
+---------------------------+------------------------------------------+
```

2. Create a subnet (for example, `provision-subnet`) on the FLAT Tenant network created in Step 1:

```
$ openstack subnet create provision-subnet --network provision --dhcp
--subnet-range 192.168.200.0/24
```

Output:

```
+-------------------+------------------------------------------------+
| Field             | Value                                          |
+-------------------+------------------------------------------------+
| allocation_pools  | 192.168.200.4-192.168.200.254                  |
| cidr              | 192.168.200.0/24                               |
| created_at        | 2019-07-29T17:05:10Z                           |
| description       |                                                |
| dns_nameservers   |                                                |
| enable_dhcp       | True                                           |
| gateway_ip        | 192.168.200.1                                  |
| host_routes       |                                                |
| id                | 9c020592-f2fb-401f-ae09-2d8888e24d55           |
| ip_version        | 4                                              |
| ipv6_address_mode | None                                           |
| ipv6_ra_mode      | None                                           |
| location          | Munch({'project': Munch({'domain_id':
'default', 'id': u'2b4b0f8af268435781a36da154ae68cc', 'name':
'admin', 'domain_name': None}), 'cloud': '', 'region_name':
'RegionOne', 'zone': None})                                      |
| name              | provision-subnet                               |
| network_id        | 5c179c83-3d38-4d17-aa37-0991d473b244           |
| prefix_length     | None                                           |
| project_id        | 2b4b0f8af268435781a36da154ae68cc               |
| revision_number   | 0                                              |
| segment_id        | None                                           |
| service_types     |                                                |
| subnetpool_id     | None                                           |
| tags              |                                                |
| updated_at        | 2019-07-29T17:06:10Z                           |
+-------------------+------------------------------------------------+
```

Note: You can set up a multitenant network to launch your instances as described in https://docs.openstack.org/ironic/latest/admin/multitenancy.html.

5.3 Adding Images to Glance

Bare Metal provisioning requires two sets of images:

- Deploy images are used by the Bare Metal service to prepare the Bare Metal server for actual OS deployment.
- Guest images are installed on the Bare Metal server to be used by the end user. The guest image was created in Creating a Guest Image.

In this section we will create the deploy images and add all the images to Glance, the Image Service.

1. Download the deploy images from the following link and extract it:

   https://images.rdoproject.org/master/delorean/consistent/ironic-python-agent.tar

2. Add the deploy images to the Image service:

   For `kernel`:

```
$ openstack image create tripleo-deploy-kernel --public \
      --disk-format aki --container-format aki \
      --file ironic-python-agent.kernel
```

   Output:

```
+------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | da442b3aae20aa0c342e3e2050e3cefb                     |
| container_format | aki                                                  |
| created_at       | 2019-08-29T10:05:55Z                                 |
| disk_format      | aki                                                  |
| file             | /v2/images/faa6d0ed-e58d-4f3f-bc03-67df1b777767/file|
| id               | faa6d0ed-e58d-4f3f-bc03-67df1b777767                 |
| min_disk         | 0                                                    |
| min_ram          | 0                                                    |
| name             | tripleo-deploy-kernel                                |
| owner            | 2b4b0f8af268435781a36da154ae68cc                     |
| properties       | os_hash_algo='sha512',
os_hash_value='7eb4fa2cd07d0406647c790b63461ed073aa72ae929ed6464fff2c436c
62ccf4ab2ca5f43a5da72a55133b22461fa0eccdfef48f4b74be3f4aea2ddaa6fd44bd',
os_hidden='False'                                                          |
| protected        | False                                                |
| schema           | /v2/schemas/image                                    |
| size             | 6648000                                              |
| status           | active                                               |
| tags             |                                                      |
| updated_at       | 2019-08-29T10:05:56Z                                 |
| virtual_size     | None                                                 |
| visibility       | public                                               |
+------------------+------------------------------------------------------+
```

For `initrd`:

```
$ openstack image create tripleo-deploy-initrd --public \
      --disk-format ari --container-format ari \
      --file ironic-python-agent.initramfs
```

Output:

```
+------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | 2c234904727bf7b436356683630c7900                     |
| container_format | ari                                                  |
| created_at       | 2019-08-29T10:08:38Z                                 |
| disk_format      | ari                                                  |
| file             | /v2/images/1e1254bb-2a50-40c1-99d8-18e1e7de44d8/file|
| id               | 1e1254bb-2a50-40c1-99d8-18e1e7de44d8                 |
| min_disk         | 0                                                    |
| min_ram          | 0                                                    |
| name             | tripleo-deploy-initrd                                |
| owner            | 2b4b0f8af268435781a36da154ae68cc                     |
| properties       | os_hash_algo='sha512',
os_hash_value='4cfef7335281fed5c50903de24a545bc94b28538120e1f44dd38127a75
b28a8d08ad1b85203295e2de9f3d77a4903e33fe35ae480490539c288f4ac55cf1903f',
os_hidden='False'                                                       |
| protected        | False                                                |
| schema           | /v2/schemas/image                                    |
| size             | 297841549                                            |
| status           | active                                               |
| tags             |                                                      |
| updated_at       | 2019-08-29T10:08:44Z                                 |
| virtual_size     | None                                                 |
| visibility       | public                                               |
+------------------+------------------------------------------------------+
```

3. Add the guest OS image.

   Provide the location of the image file.

Note: CentOS is used as an example.

```
$ openstack image create --public --disk-format qcow2 \
      --container-format bare --file CentOS-7-x86_64-GenericCloud.qcow2
centos
```

Output:

```
+------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | 359d91b5a588c0fe1150c0642247ec4a                     |
| container_format | bare                                                 |
| created_at       | 2019-08-29T10:35:38Z                                 |
| disk_format      | qcow2                                                |
| file             | /v2/images/b6c90bd6-fcd9-4a84-97bb-8871bcec4fb4/file|
| id               | b6c90bd6-fcd9-4a84-97bb-8871bcec4fb4                 |
```

```
| min_disk          | 0                                                      |
| min_ram           | 0                                                      |
| name              | centos                                                 |
| owner             | 2b4b0f8af268435781a36da154ae68cc                       |
| properties        | os_hash_algo='sha512',
os_hash_value='aab05d5e5ba5e9c5534683bff2d52e486caa4aff23a83153375451f19b
75d72c08cc548401b65fb1ff4e377f6251fccaba567f0bb4bc5ccdcd1b2f2afc709608',
os_hidden='False'                                                        |
| protected         | False                                                  |
| schema            | /v2/schemas/image                                      |
| size              | 1058930688                                             |
| status            | active                                                 |
| tags              |                                                        |
| updated_at        | 2019-08-29T10:36:00Z                                   |
| virtual_size      | None                                                   |
| visibility        | public                                                 |
+-------------------+--------------------------------------------------------+
```

5.4 Enrolling the OPA Node in Ironic

For this example, we assume `opa-0`, `opa-1`, and so on are OPA bare metal nodes that we will enroll and provision through Ironic.

In this procedure, a Bare Metal node (`opa-0`) is created with two ports; one port for provisioning and one port for opa-network.

To enroll the OPA nodes in Ironic, perform the following steps:

1. Input the BMC card details (username, password, and address) of the node you want to provision as well as the name you want to assign to the node (`opa-0` in this example).

   ```
   $ openstack baremetal node create --driver ipmi --name opa-0 \
         --driver-info ipmi_username=root \
         --driver-info ipmi_password=<password> \
         --driver-info ipmi_address=10.228.211.27
   ```

   Output:

   ```
   +------------------------+------------------------------------------+
   | Field                  | Value                                    |
   +------------------------+------------------------------------------+
   | allocation_uuid        | None                                     |
   | automated_clean        | None                                     |
   | bios_interface         | no-bios                                  |
   | boot_interface         | pxe                                      |
   | chassis_uuid           | None                                     |
   | clean_step             | {}                                       |
   | conductor              | phkpstl022                               |
   | conductor_group        |                                          |
   | console_enabled        | False                                    |
   | console_interface      | no-console                               |
   | created_at             | 2019-07-26T06:18:47+00:00                |
   | deploy_interface       | iscsi                                    |
   ```

```
| deploy_step             | {}                                                |
| description             | None                                              |
| driver                  | ipmi                                              |
| driver_info             | {u'ipmi_address': u'10.228.211.27',
u'ipmi_username': u'root', u'ipmi_password': u'******'}              |
| driver_internal_info    | {}                                                |
| extra                   | {}                                                |
| fault                   | None                                              |
| inspect_interface       | no-inspect                                        |
| inspection_finished_at  | None                                              |
| inspection_started_at   | None                                              |
| instance_info           | {}                                                |
| instance_uuid           | None                                              |
| last_error              | None                                              |
| maintenance             | False                                             |
| maintenance_reason      | None                                              |
| management_interface    | ipmitool                                          |
| name                    | opa-0                                             |
| network_interface       | neutron                                           |
| owner                   | None                                              |
| power_interface         | ipmitool                                          |
| power_state             | None                                              |
| properties              | {}                                                |
| protected               | False                                             |
| protected_reason        | None                                              |
| provision_state         | enroll                                            |
| provision_updated_at    | None                                              |
| raid_config             | {}                                                |
| raid_interface          | no-raid                                           |
| rescue_interface        | no-rescue                                         |
| reservation             | None                                              |
| resource_class          | None                                              |
| storage_interface       | noop                                              |
| target_power_state      | None                                              |
| target_provision_state  | None                                              |
| target_raid_config      | {}                                                |
| traits                  | []                                                |
| updated_at              | None                                              |
| uuid                    | 2350d119-2174-4c3a-9cd8-6576a5a709cb              |
| vendor_interface        | ipmitool                                          |
+-------------------------+---------------------------------------------------+
```

```
$ RAMFS_IMAGE=$(openstack image list | grep tripleo-deploy-initrd |
awk '{print $2}')
```

```
$ KERNEL_IMAGE=$(openstack image list | grep tripleo-deploy-kernel |
awk '{print $2}')
```

```
$ openstack baremetal node set opa-0 \
      --driver-info deploy_kernel=$KERNEL_IMAGE \
      --driver-info deploy_ramdisk=$RAMFS_IMAGE
```

```
$ openstack baremetal node set opa-0 --resource-class=baremetal
```

```
$ openstack baremetal node set opa-0 --property cpu_arch=x86_64
$ openstack baremetal node set opa-0 --property
capabilities="boot_option:local"
```

```
$ openstack baremetal node add trait opa-0 CUSTOM_GOLD
Added trait CUSTOM_GOLD
```

2. Create a Bare Metal OPA port to serve on the OPA network.

   This port represents the HFI port `ib0` for the OPA node. Use the OPA Port GUID of the OPA node to create this port. Refer to Figure 2.

```
$ NODE_ID=$(openstack baremetal node list | grep opa-0 | awk '{print
$2}')
$ openstack baremetal port create "00:11:75:60:35:7f" \
      --node $NODE_ID --pxe-enabled false \
      --extra client-id="0xfe80000000000000001175010160357f"
```

   Output:

```
+-----------------------+-------------------------------------------+
| Field                 | Value                                     |
+-----------------------+-------------------------------------------+
| address               | 00:11:75:60:35:7f                         |
| created_at            | 2019-07-30T08:37:17+00:00                 |
| extra                 | {u'client-id':
u'0xfe80000000000000001175010160357f'}                              |
| internal_info         | {}                                        |
| is_smartnic           | False                                     |
| local_link_connection | {}                                        |
| node_uuid             | 2350d119-2174-4c3a-9cd8-6576a5a709cb      |
| physical_network      | None                                      |
| portgroup_uuid        | None                                      |
| pxe_enabled           | False                                     |
| updated_at            | None                                      |
| uuid                  | e8a7db91-0b09-494a-9d1b-f638c1fb7fcd      |
+-----------------------+-------------------------------------------+
```

Note: Separate ports need to be created for every HFI or Ethernet NIC available in the node. If a node has two HFIs and one Ethernet NIC, then a total of three Bare Metal ports should be created in Ironic.

Ironic register ports with the Ethernet MAC address format (48 bits/6 bytes). Each MAC address must be unique in a network. The Intel® Omni-Path PortGUID has 64 bits (8 bytes).

Intel recommended that you create a MAC address for the OPA ports by concatenating the 24-bit Intel® Omni-Path OUI (0x001175) with the lower three bytes of the port GUID. To form a MAC address for OpenStack, use the high three bytes (00:11:75 in this example) and the low three bytes (60:35:7f in this example) of the PortGUID (00:11:75:60:35:7f in this example). Note that dual

ported, single ASIC HFIs such as dual ported Intel® Xeon® Scalable Processor (-F), and some dual ported cards in third party storage servers where both ports are active are not supported at this time. This approach only results in a unique MAC address for servers with a single active OPA port or servers with two distinct HFI ASICs or cards. **Please consult technical support for configurations in such situations as the MAC address will not be unique for each port.**

An Intel® Omni-Path port requires client ID. The client id is <12-byte vendor prefix>:<8 byte port GUID>. The client-id for this example will be 0xfe80000000000000001175010160357f.

To list all of the HFI Port GUIDs in a cluster, run the following command on the Fabric Manager node:

```
$ opareport -o comps -d 3 -x -F nodetype:FI|opaxmlextract -d \; -e
NodeDesc -e PortInfo.GUID -s Focus -s SMs -s Neighbor
```

3. Create a Bare Metal port for PXE booting (switch_id and port_id are from the openvswitch service running on the controller node).

   This port represents the ethernet interface `eno2` on the OPA node. Use the MAC address of this interface to create the port.

```
$ DATAPATH_ID=$(sudo ovs-vsctl get Bridge br-ex datapath-id)
$ openstack baremetal port create "00:1e:67:c7:a9:7d" \
      --node $NODE_ID --pxe-enabled true \
      --local-link-connection switch_id="$DATAPATH_ID" \
      --local-link-connection port_id=br-ex \
      --physical-network public
```

   Output:

```
+-----------------------+-------------------------------------------+
| Field                 | Value                                     |
+-----------------------+-------------------------------------------+
| address               | 00:1e:67:c7:a9:7d                         |
| created_at            | 2019-07-30T08:40:50+00:00                 |
| extra                 | {}                                        |
| internal_info         | {}                                        |
| is_smartnic           | False                                     |
| local_link_connection | {u'port_id': u'br-ex', u'switch_id':
u'0000001e67d5e4a3'}                                                |
| node_uuid             | 2350d119-2174-4c3a-9cd8-6576a5a709cb      |
| physical_network      | public                                    |
| portgroup_uuid        | None                                      |
| pxe_enabled           | True                                      |
| updated_at            | None                                      |
| uuid                  | 40acad57-2acc-4b4b-9879-83f5781b7fb1      |
+-----------------------+-------------------------------------------+
```

Note: Be sure to double-check the MAC addresses to be provisioned for each Bare Metal node. A mismatch will prevent provisioning and the nodes will be stuck in a *clean wait* or *enroll* state.

4. Make the node available for provisioning:

```
openstack baremetal node manage opa-0
openstack baremetal node provide opa-0
```

5. Verify that the node is available as a hypervisor to Nova.

```
$ openstack hypervisor list
```

Output:

```
+----+--------------------------------------+-----------------+---------------+-------+
| ID | Hypervisor Hostname                  | Hypervisor Type | Host IP       | State |
+----+--------------------------------------+-----------------+---------------+-------+
| 22 | 2350d119-2174-4c3a-9cd8-6576a5a709cb | ironic          | 10.228.208.192 | up    |
+----+--------------------------------------+-----------------+---------------+-------+
```

You should see the node uuid listed in the Hypervisor Hostname column. The Host IP shown will be that of the OpenStack controller.

6. If still the node is not visible, run the following command to verify:

```
nova-manage cell_v2 discover_hosts --by-service --verbose
```

7. Repeat steps 1 - 6 for all remaining OPA bare metal nodes (`opa-1`, `opa-2`, and so on).

5.5 Launching a Nova Instance with the Image

Once the node is available to Nova, you can launch the Nova instance by booting the OPA Bare Metal node. Refer to the following steps:

1. Create a Nova keypair:

```
$ openstack keypair create --public-key ~/.ssh/id_rsa.pub testkey
```

Output:

```
+-------------+-------------------------------------------------+
| Field       | Value                                           |
+-------------+-------------------------------------------------+
| fingerprint | 15:54:51:eb:46:e9:1c:ce:3e:28:b5:ab:45:56:72:2f |
| name        | testkey                                         |
| user_id     | 8808ab3f87ab4bf6932b388dfd514aa4                |
+-------------+-------------------------------------------------+
```

2. Add the local boot option to the `baremetal` flavor, or your flavor that represents a baremetal resource, so that the node's subsequent boot occurs from the local boot loader installed on the disk.

```
$ openstack flavor set baremetal --property capabilities:boot_option="local"
```

3. Create a `cloud-config` file and copy the following content, to be used for user data from the metadata server.

   Use a hashed password for security. In this example file, the password of `root` user is set.

```
#cloud-config
chpasswd:
  list: |
    root:password
  expire: False
```

4. Boot the server.

```
$ openstack server create --config-drive true --image centos \
  --network opa-network --network provision --key-name testkey \
  --flavor baremetal --user-data cloud-config opa-node-0
```

   Output:

```
+-------------------------------------+----------------------------------------+
| Field                               | Value                                  |
+-------------------------------------+----------------------------------------+
| OS-DCF:diskConfig                   | MANUAL                                 |
| OS-EXT-AZ:availability_zone         |                                        |
| OS-EXT-SRV-ATTR:host                | None                                   |
| OS-EXT-SRV-ATTR:hypervisor_hostname | None                                   |
| OS-EXT-SRV-ATTR:instance_name       |                                        |
| OS-EXT-STS:power_state              | NOSTATE                                |
| OS-EXT-STS:task_state               | scheduling                             |
| OS-EXT-STS:vm_state                 | building                               |
| OS-SRV-USG:launched_at              | None                                   |
| OS-SRV-USG:terminated_at            | None                                   |
| accessIPv4                          |                                        |
| accessIPv6                          |                                        |
| addresses                           |                                        |
| adminPass                           | R43H7c9jiC2S                           |
| config_drive                        | True                                   |
| created                             | 2019-08-30T16:34:47Z                   |
| flavor                              | baremetal (8efef170-efa0-481f-9bb9-
66188571a865)                                                                |
| hostId                              |                                        |
| id                                  | 97f53137-52f8-49f7-9973-f830c67c6640   |
| image                               | centos (b6c90bd6-fcd9-4a84-97bb-
8871bcec4fb4)                                                                |
| key_name                            | testkey                                |
| name                                | opa-node-0                             |
| progress                            | 0                                      |
| project_id                          | 2b4b0f8af268435781a36da154ae68cc       |
| properties                          |                                        |
| security_groups                     | name='default'                         |
| status                              | BUILD                                  |
| updated                             | 2019-08-30T16:34:47Z                   |
| user_id                             | 8808ab3f87ab4bf6932b388dfd514aa4       |
| volumes_attached                    |                                        |
+-------------------------------------+----------------------------------------+
```

Note: To create a server with multiple HFIs, use the `--network <network-name>` option, multiple times. For example, to create a server with two HFIs and one Ethernet NIC, use the following command:

```
$ openstack server create --image centos --network opa-network
  --network opa-network --network provision ...
```

The screen will show a list of servers. The server being created will show as BUILD. It will take some time for the server to be created. When it is finished, the server will show as ACTIVE.

5. Verify the status of the server:

```
$ openstack server list
```

Output:

```
+--------------------------------------+------------+--------+-----------------------------------------------------+--------+--------+
| ID                                   | Name       | Status | Networks                                            | Image  | Flavor |
+--------------------------------------+------------+--------+-----------------------------------------------------+--------+--------+
| 97f53137-52f8-49f7-9973-f830c67c6640 | opa-node-0 | ACTIVE | opa-network=192.168.1.251; provision=192.168.200.163 | centos |baremetal|
+--------------------------------------+------------+--------+-----------------------------------------------------+--------+--------+
```

On the FM side, this command assigns the OPA Bare Metal port to the Virtual Fabric that was previously created and then it triggers the FM to reconfigure.

6. Run `opareport` on the FM node to view the newly created port in the OPA vFabric.

```
$ opareport -o vfinfo -d 3
```

Output:

```
Getting All Node Records...
Done Getting All Node Records
Done Getting All Link Records
Done Getting All Cable Info Records
Done Getting All SM Info Records
Done Getting vFabric Records
Getting All Port VL Tables...
Done Getting All Port VL Tables

vFabrics:
vFabric Index: 0   Name: Admin
PKey: 0x7fff   SL: 0  Select: 0x3: PKEY SL   PktLifeTimeMult: 1
MaxMtu: unlimited  MaxRate: unlimited   Options: 0x03: Security QoS
QOS: Bandwidth:  50%
PreemptionRank: 0  HoQLife:    8 ms
     NodeGUID          Port Type Name
     0x001175010165acd1   1 FI   phwtstl006 hfi1_0
     0x001175010160357f   1 FI   opa-node-0.novalocal
     0x001175010165b15b   1 FI   phwtstl008 hfi1_0
     0x001175010165b22c   1 FI   phwtstl006 hfi1_1
     0x001175010265bb30   0 SW   OmniPth00117501ff65bb30
     0x001175010265bd26   0 SW   OmniPth00117501ff65bd26

vFabric Index: 1   Name: c017d599-9904-42dd-bd7c-23d6dfdac4f1
PKey: 0x53   SL: 1  Select: 0x3: PKEY SL   PktLifeTimeMult: 1
```

```
MaxMtu: unlimited  MaxRate: unlimited   Options: 0x03: Security QoS
QOS: Bandwidth:  50%  PreemptionRank: 0  HoQLife:    8 ms
     NodeGUID          Port Type Name
     0x001175010160357f   1 FI   opa-node-0.novalocal
2 VFs
-------------------------------------------------------------------------
```

The newly created port will now appear within the OpenStack vFabric (c017d599-9904-42dd-bd7c-23d6dfdac4f1).

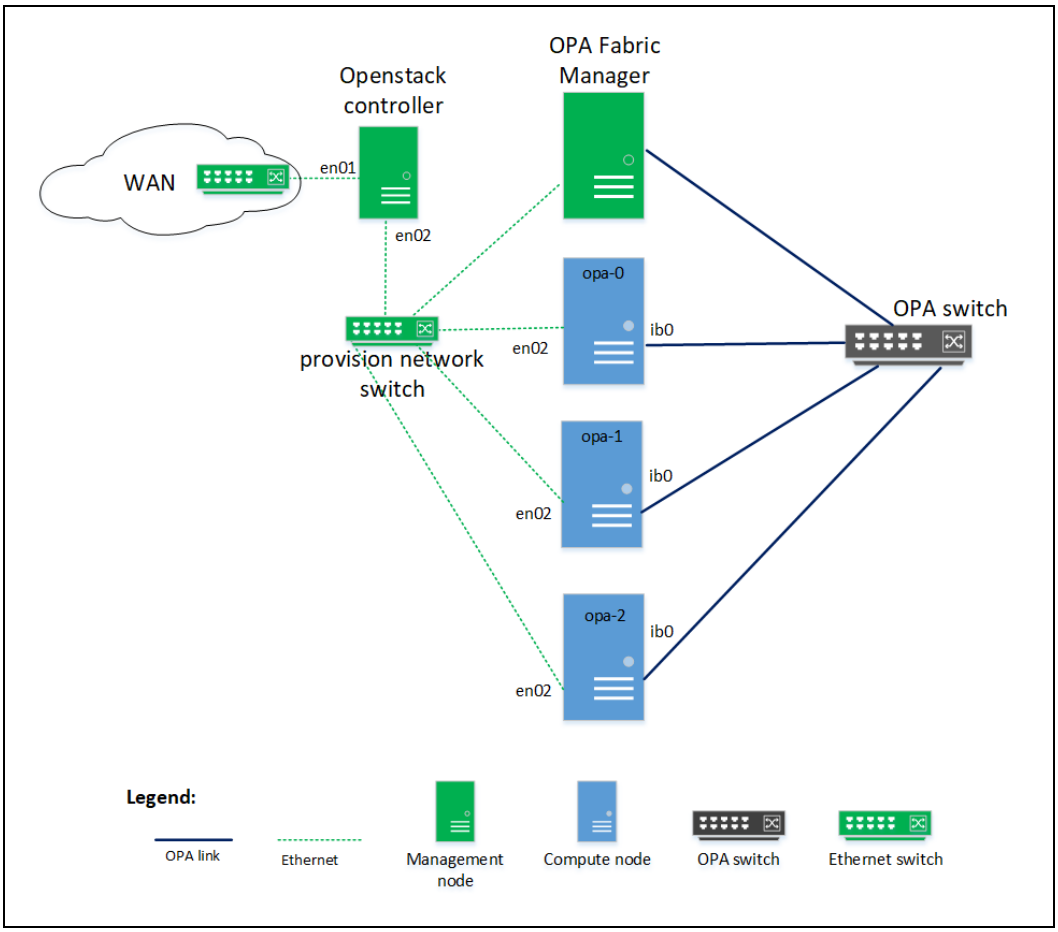
6 Example OpenStack Installation

This section describes an example OpenStack installation using DevStack. Follow this section only when you don't have an OpenStack setup running.

6.1 Example Architecture

The following figure shows an example Intel® Omni-Path network fabric architecture using OpenStack. In general, the architecture requires an OpenStack controller, at least one Bare Metal node, and an Intel® Omni-Path Fabric Manager node.

Figure 2. Network Architecture (Example)



The example network consists of the following components:

- Openstack controller: The node where OpenStack services run.
- Bare Metal nodes (`opa-0, opa-1, opa-2`): The OPA nodes to be deployed within the network.
- FM node: The node running the Intel® Omni-Path Fabric Manager (FM).
- Provision(ing) network switch: The switch used for Bare Metal provisioning.
- OPA switch: The switch used for tenant workload traffic.
- External network switch: The switch used to access the WAN and public network.

6.2 Setup Overview

Note: When setting up your architecture, you may have different requirements than the ones shown in the example.

For the example architecture described in Figure 2, two independent networks are used:

- Flat (non-multi-tenant) network used for provisioning nodes using PXE. This interface is `eno2`.
- Multi-tenant network for Intel® Omni-Path fabrics. This interface is `ib0`.

Note: The provisioning network can be set up as a multi-tenant network as long as it includes the supported network hardware. For more information, refer to https://docs.openstack.org/ironic/latest/admin/multitenancy.html.

Setup requirements include:

- All the OpenStack services reside on a single controller node and are set up using DevStack.
- All Ethernet `eno2` interfaces are connected to the provision network switch.
- All HFI `ib0` interfaces are connected to the OPA switch.

(Optional) All corresponding interfaces have the same names in all nodes.

The following lists the cluster requirements:

- Three Bare Metal nodes:
  - High-performing servers with one network interface card (NIC) and one HFI, configured for PXE boot over Ethernet and Intel® Omni-Path interface, respectively
  - All Bare Metal nodes must have Intelligent Platform Management Interface (IPMI) and BMC
- One OpenStack Controller:
  - Running RHEL* 7.6 or CentOS*-7 (1908)

  - Two Ethernet cards: One for Internet access and one for provision network for deployment
  - OpenStack version: Stein
  - RAM: 32 GB
  - Hard Drive: 250 GB
- Fabric Manager:
  - IFS version: 10.9.0.2 or better on RHEL 7.6
  - One Ethernet switch for provisioning
  - One OPA switch

6.3 OpenStack Installation

This section provides instructions for setting up a functional OpenStack all-in-one environment using DevStack on the controller node. DevStack installs the all the OpenStack core components by default. We are enabling the following projects to be installed by DevStack for our need:

- Ironic: For provisioning OPA nodes.
- `networking-omnipath`: ML2 plugin for OPA card on VLAN network.
- `networking-baremetal`: ML2 plugin for configuring baremetal ports on flat network.

Note: You can use tenant networks for provisioned instances by using the Neutron network interface with supported hardware. Refer to *Multi-tenancy in the Bare Metal Service* (https://docs.openstack.org/ironic/latest/admin/multitenancy.html) for detailed instruction of building a multi-tenant provisioning environment for Ironic.

The OpenStack controller node has two interfaces:

- `eno1` is connected to an external switch for WAN and public network access.
- `eno2` is connected to the provision network switch. This interface provides PXE and Dynamic Host Configuration Protocol (DHCP) services to the provisioned instances. `eno2` should be configured as the interface for the bridging to external network (br-ex) bridge.

In this setup, we are using two networks in Neutron for:

- Provision network: 192.168.200.0/24
- OPA network: 192.168.1.0/24

6.4 Prerequisites to Installing DevStack

Before installing DevStack, perform the following:

If your OpenStack controller server is running behind a proxy server, you need to set environment variables `http_proxy`, `https_proxy`, and `no_proxy`. The `no_proxy` environment variable should contain `localhost`, as well as the IP of the OpenStack controller node.

6.5 Installing DevStack

On the controller node, perform the following steps:

1. Create the `stack` user and add it to `sudo`:

```
sudo useradd -s /bin/bash -d /opt/stack -m stack
echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
```

2. Switch to the `stack` user and clone DevStack:

```
sudo su – stack
git clone http://opendev.org/openstack/devstack.git devstack -b
stable/stein
```

3. Create `local.conf` in devstack directory:

```
cd devstack
cat >local.conf <<END
[[local|localrc]]
# Use OpenDev and make sure latest commits are always fetched
GIT_BASE=${GIT_BASE:-https://opendev.org}
RECLONE=yes

# Credentials
ADMIN_PASSWORD=password
DATABASE_PASSWORD=password
RABBIT_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=password
SWIFT_HASH=password
SWIFT_TEMPURL_KEY=password

# Checkout stable/stein branches
NOVA_BRANCH=stable/stein
NEUTRON_BRANCH=stable/stein
GLANCE_BRANCH=stable/stein
CINDER_BRANCH=stable/stein
SWIFT_BRANCH=stable/stein
KEYSTONE_BRANCH=stable/stein
HORIZON_BRANCH=stable/stein
REQUIREMENTS_BRANCH=stable/stein
PLACEMENT_BRANCH=stable/stein
```

```
# Enable networking-baremetal plugin
enable_plugin networking-baremetal
https://opendev.org/openstack/networking-baremetal.git stable/stein
enable_service ir-neutronagt

# Enable Ironic plugin
enable_plugin ironic https://opendev.org/openstack/ironic
stable/stein enable_service networking_baremetal

# Disable nova novnc service, ironic does not support it anyway
disable_service n-novnc

# Enable Swift for the direct deploy interface
enable_service s-proxy
enable_service s-object
enable_service s-container
enable_service s-account

# Enable Horizon
enable_service horizon

# Disable Cinder
disable_service cinder c-sch c-api c-vol

# Swift temp URL's are required for the direct deploy interface
SWIFT_ENABLE_TEMPURLS=True

# Some Ironic options
IRONIC_BAREMETAL_BASIC_OPS=True
DEFAULT_INSTANCE_TYPE=baremetal
IRONIC_DEPLOY_DRIVER=ipmi
IRONIC_VM_COUNT=0

# To build your own IPA ramdisk from source, set this to True
IRONIC_BUILD_DEPLOY_RAMDISK=False

# Make Nova use the ironic driver instead of libvirt
VIRT_DRIVER=ironic

# Do not create devstack default networks 10.0.0.0/22 and
172.24.4.0/24 NEUTRON_CREATE_INITIAL_NETWORKS=False

# Instead reserve provision network's subnet range
IPV4_ADDRS_SAFE_TO_USE=192.168.200.0/24

# "public" physnet will connect to br-ex bridge. It will support the
provision network
OVS_PHYSICAL_BRIGE=br-ex
PHYSICAL_NETWORK=public
OVS_BRIDGE_MAPPINGS=public:br-ex

# Actually create the provision network for Ironic (will use "public"
physnet)
IRONIC_PROVISION_NETWORK_NAME=provision
IRONIC_PROVISION_SUBNET_PREFIX=192.168.200.0/24
IRONIC_PROVISION_SUBNET_GATEWAY=192.168.200.3
IRONIC_PROVISION_PROVIDER_NETWORK_TYPE=flat
```

```
IRONIC_PROVISION_ALLOCATION_POOL="start=192.168.200.4,end=192.168.200
.254"

# Log all output to files
LOGFILE=/opt/stack/devstack.log
LOGDIR=/opt/stack/logs
IRONIC_VM_LOG_DIR=/opt/stack/ironic-bm-logs

# Enable networkin-omnipath plugin
enable_plugin networking-omnipath https://opendev.org/x/networking-
omnipath.git

# ml2_conf.ini config for omnipath mechanism driver
[[post-config|/etc/neutron/plugins/ml2/ml2_conf.ini]]
[ml2_omnipath]
username = "root"
ssh_key = <PATH_TO_SSH_PRIVATEKEY_OF_CONTROLLER_NODE>
ip_address = <IPV4_IP_of_OPA_FM>

[[post-config|/etc/neutron/plugins/ml2/ironic_neutron_agent.ini]]
[ironic]
memcached_servers = localhost:11211
signing_dir = /var/cache/neutron cafile = /opt/stack/data/ca-
bundle.pem
project_name = service
user_domain_name = Default
password = password
username = ironic
auth_url = http://10.228.208.192/identity # change IP address to
OpenStack controller
auth_type = password

[[post-config|/etc/ironic/ironic.conf]]
[DEFAULT]
enabled_network_interfaces = flat,neutron
default_network_interface = neutron
[neutron]
provisioning_network = provision

END
```

4. Run `stack.sh` for installation:

```
./stack.sh
```

5. Add `eno2` to the `br-ex` bridge to provide access to the virtual network:

```
sudo ovs-vsctl add-port br-ex eno2
```

6. Apply the following iptables rules to allow traffic:

```
sudo iptables -t nat -I POSTROUTING -o eno1 -j MASQUERADE
sudo iptables -I FORWARD -i eno1 -o br-ex -m state  --state
RELATED,ESTABLISHED  -j ACCEPT
sudo iptables -I FORWARD -i br-ex -o eno1 -j ACCEPT
```

7. To deploy an OPA node after DevStack installation, perform the steps in the following section:

   a) Creating a Guest Image

   b) Deploying a Node in an OPA Fabric