## Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.4</td>
<td>June 9, 2015</td>
<td>Document updated for release of 1.4 on Intel® Open Network Platform Server 1.4</td>
</tr>
<tr>
<td>1.3</td>
<td>February 23, 2015</td>
<td>Document updated for release of 1.3 of Intel® Open Network Platform Server 1.3</td>
</tr>
<tr>
<td>1.2</td>
<td>December 15, 2014</td>
<td>Document prepared for release 1.2 of Intel® Open Network Platform Server 1.2</td>
</tr>
<tr>
<td>1.1.1</td>
<td>October 29, 2014</td>
<td>Changed two links to the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <a href="https://01.org/sites/default/files/page/vbng-scripts.tgz">https://01.org/sites/default/files/page/vbng-scripts.tgz</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <a href="https://01.org/sites/default/files/page/qat_patches_netkeyshim.zip">https://01.org/sites/default/files/page/qat_patches_netkeyshim.zip</a></td>
</tr>
<tr>
<td>1.1</td>
<td>September 18, 2014</td>
<td>Minor edits throughout the document</td>
</tr>
<tr>
<td>1.0</td>
<td>August 21, 2014</td>
<td>Initial document for release of Intel® Open Network Platform Server 1.1</td>
</tr>
</tbody>
</table>
Contents

1.0 Audience and Purpose .................................................................................................................. 7

2.0 Summary ........................................................................................................................................ 8
  2.1 Network Services Examples ........................................................................................................... 10
    2.1.1 Suricata (Next Generation IDS/IPS Engine) .......................................................................... 10
    2.1.2 vBNG (Broadband Network Gateway) ..................................................................................... 10

3.0 Hardware Components .................................................................................................................. 11

4.0 Software Versions .......................................................................................................................... 12
  4.1 Obtaining Software Ingredients .................................................................................................... 13

5.0 Installation and Configuration Guide ............................................................................................. 14
  5.1 Automated Installation Using Scripts ............................................................................................ 14
  5.2 Manual Installation Procedure ...................................................................................................... 14
    5.2.1 Instructions Common to Controller and Compute Nodes ......................................................... 15
      5.2.1.1 BIOS Settings ...................................................................................................................... 15
      5.2.1.2 Operating System Installation and Configuration ................................................................. 15
    5.2.2 Controller Node Setup .............................................................................................................. 20
      5.2.2.1 OpenStack (Kilo) ................................................................................................................ 20
      5.2.2.2 OpenStack Installation Procedures ...................................................................................... 22
    5.2.3 Compute Node Setup .............................................................................................................. 24
      5.2.3.1 Host Configuration .............................................................................................................. 25

6.0 Virtual Machine Deployment Using OpenStack ............................................................................... 29
  6.1 Preparation with OpenStack ......................................................................................................... 29
    6.1.1 Deploying Virtual Machines .................................................................................................... 29
      6.1.1.1 Default Settings .................................................................................................................. 29
      6.1.1.2 Manual Deployment with Custom Settings ......................................................................... 30
  6.2 Non-Uniform Memory Access (NUMA) Placement and SR-IOV Passthrough for OpenStack ........ 32
    6.2.1 Create Flavor for NUMA Placement .......................................................................................... 32
    6.2.2 Prepare Compute Node for SR-IOV Passthrough .................................................................... 33
    6.2.3 DevStack Configurations ......................................................................................................... 35
    6.2.4 Create VM with NUMA Placement and SR-IOV ................................................................... 36
6.3 CPU Pinning for OpenStack ................................................................. 41
  6.3.1 Prepare Compute Node for CPU Pinning ........................................... 41
  6.3.2 Create Flavor with CPU Pinning ...................................................... 41
  6.3.3 Create VM with CPU Pinning .......................................................... 42
6.4 Using OpenDaylight ............................................................................. 42
  6.4.1 Preparing the OpenDaylight Controller ............................................ 42
  6.4.2 Prepare for DevStack ........................................................................ 43
  6.4.3 Additional Configurations and Operations .......................................... 44
  6.4.4 Monitor Network Flow with OpenDaylight ......................................... 45
7.0 Use Cases with Virtual Network Functions ............................................ 48
  7.1 Generic VNF Configurations ................................................................. 48
    7.1.1 Local VNF ..................................................................................... 48
    7.1.2 Remote VNF ................................................................................ 49
    7.1.3 Network Configuration with Source and Sink VM ............................. 50
  7.2 Installation and Configuration of vIPS .................................................. 51
    7.2.1 Setup ......................................................................................... 51
    7.2.2 Local vIPS Test ........................................................................... 51
    7.2.3 Remote vIPS Test ........................................................................ 53
  7.3 Installation and Configuration of the vBNG ............................................ 55
Appendix A Sample Local.conf Files .......................................................... 58
  A.1 Sample Local.conf Files for OpenDaylight Configurations .................. 58
  A.2 Sample Local.conf Files for SR-IOV Configurations ............................. 61
Appendix B Configuring the Proxy ............................................................... 64
Appendix C Configuring Horizon UI to Deploy Virtual Machines ................ 66
  C.1 Custom VM Image and Zoning .............................................................. 66
  C.2 Creating Additional Networks .............................................................. 69
  C.3 VM Deployment ................................................................................ 71
Appendix D Glossary .................................................................................... 73
Appendix E References ................................................................................ 74
Legal Information ....................................................................................... 75
Figures

Figure 2–1. Intel ONP Server — Hardware and Software Ingredients ............................................. 8
Figure 2–2. Generic Setup with Controller and Two Compute Nodes ........................................... 9
Figure 7–1. Local VNF .................................................................................................................. 48
Figure 7–2. Remote VNF ................................................................................................................ 49
Figure 7–3. Local vIPS sriovNet ................................................................................................... 53
Figure 7–4. Remote vIPS ............................................................................................................... 54
Tables

Table 3–1. Hardware Ingredients (Code-named Wildcat Pass) ...............................................11
Table 4–1. Software Versions ..................................................................................................12
Table 4–2. Software Ingredients ............................................................................................13
Table 5–1. BIOS Settings .........................................................................................................15
1.0 Audience and Purpose

The primary audiences for this document are architects and engineers implementing the Intel® Open Network Platform Server Reference Architecture using Open Source software. Software ingredients include the following:

- DevStack*
- OpenStack*
- OpenDaylight*
- Data Plane Development Kit (DPDK)*
- Regular OpenvSwitch*
- Open vSwitch* with DPDK-netdev
- Fedora*

This document provides a guide for integration and performance characterization using the Intel® Open Network Platform Server (Intel ONP Server). Content includes high-level architecture, setup and configuration procedures, integration learnings, and a set of baseline performance data. This information is intended to help architects and engineers evaluate Network Function Virtualization (NFV) and Software Defined Network (SDN) solutions.

The purpose of documenting configurations is not to imply any preferred methods. Providing a baseline configuration of well-tested procedures, however, can help achieve optimal system performance on an IA Platform when developing an NFV/SDN solution.
2.0 Summary

The Intel ONP Server uses Open Source software to help accelerate SDN and NFV commercialization with the latest Intel Architecture Communications Platform.

This document describes how to set up and configure the controller and compute nodes for evaluating and developing NFV/SDN solutions using the Intel® Open Network Platform ingredients.

Platform hardware is based on an Intel® Xeon® DP Server with the following hardware:

- Intel® dual Xeon® Processor Series E5-2600 V3
- Intel® XL710 4x10 GbE Adapter

The host operating system is Fedora® 21 with Qemu-kvm virtualization technology. Software ingredients include the DPDK, Open vSwitch, Open vSwitch with DPDK-netdev, OpenStack, and OpenDaylight. Figure 2–1 shows the corresponding version information for the components involved.

![Intel® Open Network Platform for Servers](image)

**Figure 2–1. Intel ONP Server — Hardware and Software Ingredients**
The test cases described in this document were designed to illustrate functionality using the specified ingredients, configurations, and test methodology. Figure 2–2 shows the simple network topology used.

The test cases documented are designed to:

- Verify communication between controller and compute nodes.
- Validate basic controller functionality.
- Demonstrate how to deploy virtualized network functions (VNFs) and generate traffic processed with them.

By doing this, we hope to show users how they can use the “plumbing” that is put in place with great efficiency and optimized performance for Intel Architecture to deploy their VMs and VNFs in a way that enables more CPU cycles to be dedicated to run these VMs and VNFs, instead of pushing network packets to and from physical and virtual networks.

Figure 2–2 shows a generic SDN/NFV setup. In this configuration, the orchestrator and controller (management and control plane) run on one server, and the two compute nodes (data plane) run on two individual server nodes. The differences in network configuration to enable this setup are shown with the management and data ports. Note that many variations of this setup can be deployed.
2.1 Network Services Examples

The following network services in this section are included as examples that have been tested with the Intel® Open Network Platform Server Reference Architecture. They are demonstrated as use cases running as virtualized instances deployed and controlled by OpenStack.

2.1.1 Suricata (Next Generation IDS/IPS Engine)

Suricata is a high-performance Network IDS, IPS, and Network Security Monitoring engine developed by the Open Information Security Foundation, its supporting vendors, and the community. Refer to the following URL: http://suricata-ids.org/

2.1.2 vBNG (Broadband Network Gateway)


A BNG may also be known as a Broadband Remote Access Server (BRAS) and routes traffic to and from broadband remote access devices, such as digital subscriber line access multiplexers. This network function is included as an example of a workload that can be virtualized on the Intel ONP Server.

Additional information on the performance characterization of this vBNG implementation can be found at: https://networkbuilders.intel.com/docs/Network_Builders_RA_vBRAS_Final.pdf

Refer to section 7.3, Installation and Configuration of the vBNG, or Appendix B: Configuring the Proxy, for more information on running the BNG as an appliance.
3.0 Hardware Components

The following table, Tables 3-1, provides details of the platform hardware components used for our testing purposes. The “Notes” column details some of the fine tunings enabled on the hardware.

Table 3-1. Hardware Ingredients (Code-named Wildcat Pass)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td>Intel® Server Board S2600WTT 1100 W power supply</td>
<td>Intel® Xeon® processor-based DP server (Formerly code-named Wildcat Pass) 120 GB SSD 2.5in SATA 6GB/s Intel Wolfsville SSDSC2BB120G4. Supports SR-IOV.</td>
</tr>
<tr>
<td>Processors</td>
<td>Intel® Dual Xeon® Processor E5-2697 V3 2.6 GHz, 35 MB, 145 W, 14 cores</td>
<td>(Formerly code-named Haswell) 14 cores, 2.60 GHz, 145 W, 35 MB total cache per processor, 9.6 GT/s QPI, DDR4-1600/1866/2133, 28 hyper-threaded cores per CPU for 56 total cores.</td>
</tr>
<tr>
<td></td>
<td>Intel® Dual Xeon® Processor Series E5-2699 V3 2.3 GHz, 45 MB, 145 W, 18 cores</td>
<td>(Formerly code-named Haswell) 18 cores, 2.3 GHz, 145 W, 45 MB total cache per processor, 9.6 GT/s QPI, DDR4-1600/1866/2133, 36 hyper-threaded cores per CPU for 72 total cores.</td>
</tr>
<tr>
<td>Memory</td>
<td>8 GB DDR4 RDIMM Crucial CT8G4RFS423</td>
<td>64 GB RAM (8 x 8 GB)</td>
</tr>
<tr>
<td>NICs (XL710)</td>
<td>Intel® Ethernet Controller XL710 4x 10GbE has been tested with Intel FTLX8571D3BCV-IT and Intel AFBR-703sDZ-IN2 850nm SFPs</td>
<td>(Code-named Fortville) NICs are on socket zero.</td>
</tr>
<tr>
<td>NICs (R520)</td>
<td>Intel® Ethernet Controller R520 2x 10GbE Firmware version 0x 18bf0001</td>
<td>(Code-named Niantic) NICs are on socket zero.</td>
</tr>
<tr>
<td>BIOS</td>
<td>SE5C610.86B.01.01.0008.031920151331 Release Date: 03/19/2015</td>
<td>Intel® Virtualization Technology for Direct I/O (Intel® VT-d) enabled only for SR-IOV PCI passthrough tests, hyper-threading enabled.</td>
</tr>
<tr>
<td>Quick Assist Technology</td>
<td>Intel® Communications Chipset 8950 (Coleto Creek)</td>
<td>Walnut Hill PCIe card 1x Coleto Creek; supports SR-IOV.</td>
</tr>
</tbody>
</table>
4.0 Software Versions

The function of the software ingredients involved are described with their version or configuration information in Table 4-1. For open-source components, a specific commit ID set is used for this integration. Note that the commit IDs used are the latest working set at the time of this release.

Table 4–1. Software Versions

<table>
<thead>
<tr>
<th>Software Component</th>
<th>Function</th>
<th>Version/Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fedora 21 x86_64</td>
<td>Host OS</td>
<td>Kernel version: 3.18.8-201.fc21.x86_64</td>
</tr>
<tr>
<td>Real-Time Kernel</td>
<td>Targeted towards the Telco environment, which is sensitive to low latency</td>
<td>Real-Time Kernel v3.14.36-rt34</td>
</tr>
<tr>
<td>Qemu-kvm</td>
<td>Virtualization technology</td>
<td>QEMU-KVM 2.1.3-6.fc21.x86_64</td>
</tr>
<tr>
<td>DPDK</td>
<td>Network stack bypass and libraries for packet processing; includes user space poll mode drivers</td>
<td>1.8.0</td>
</tr>
<tr>
<td>Open vSwitch</td>
<td>vSwitch</td>
<td>OVS: openvswitch v.2.3.1-git4750c96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OVS with DPDK-netdev: openvswitch v.2.3.90</td>
</tr>
<tr>
<td>OpenStack</td>
<td>SDN orchestrator</td>
<td>Kilo Release</td>
</tr>
<tr>
<td>DevStack</td>
<td>Tool for OpenStack deployment</td>
<td><a href="https://github.com/openstack-dev/devstack.git">https://github.com/openstack-dev/devstack.git</a> tag: stable/kilo</td>
</tr>
<tr>
<td>OpenDaylight</td>
<td>SDN controller</td>
<td>Helium-SR3</td>
</tr>
<tr>
<td>Suricata</td>
<td>IPS application</td>
<td>Suricata v2.0.2</td>
</tr>
</tbody>
</table>
## 4.1 Obtaining Software Ingredients

All of the open-source software ingredients involved are downloaded from their source repositories. Due to the number of ingredients involved, you will need to follow the instructions for each of the software component to deploy them.

**Table 4–2. Software Ingredients**

<table>
<thead>
<tr>
<th>Software Component</th>
<th>Software Subcomponents</th>
<th>Patches</th>
<th>Location</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPDK</td>
<td>DPDK poll mode driver, sample apps (bundled)</td>
<td></td>
<td><a href="http://dpdk.org/git/dpdk">http://dpdk.org/git/dpdk</a></td>
<td>DPDK download will be done through the DevStack script during installation.</td>
</tr>
<tr>
<td>OpenvSwitch</td>
<td></td>
<td>OVS: OpenvSwitch v.2.3.1-git4750c96 OVS with DPDK-netdev: OpenvSwitch v.2.3.90</td>
<td></td>
<td>OVS download will be done through the DevStack script during installation.</td>
</tr>
<tr>
<td>OpenStack</td>
<td></td>
<td></td>
<td>Kilo release. To be deployed using DevStack</td>
<td></td>
</tr>
<tr>
<td>OpenDaylight</td>
<td></td>
<td></td>
<td><a href="https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/distribution-karaf/0.2.3-Helium-SR3/distribution-karaf-0.2.3-Helium-SR3.tar.gz">https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/distribution-karaf/0.2.3-Helium-SR3/distribution-karaf-0.2.3-Helium-SR3.tar.gz</a></td>
<td>ODL download will be done through the DevStack script during installation.</td>
</tr>
<tr>
<td>Intel® ONP Server Release 1.4 Script</td>
<td>Helper scripts to setup SRT 1.4 using DevStack</td>
<td></td>
<td><a href="https://download.01.org/packet-processing/ONPS1.4/onps_server_1.4.tar.gz">https://download.01.org/packet-processing/ONPS1.4/onps_server_1.4.tar.gz</a></td>
<td></td>
</tr>
<tr>
<td>Suricata</td>
<td></td>
<td>Suricata version 2.0.2</td>
<td></td>
<td>yum install Suricata</td>
</tr>
</tbody>
</table>
5.0 Installation and Configuration Guide

This section describes the installation and configuration instructions to prepare the controller and compute nodes.

5.1 Automated Installation Using Scripts

In order to go with the base set of settings recommended by Intel, using the scripts provided for installation would be optimal. Many of the manual OS and OpenStack installations with DevStack are automated. The bulk of this procedure is condensed into a few steps that can be executed using these scripts. All the ingredients listed in Table 4–1, except OpenDaylight, can be installed using this method.

Before continuing with the scripts, update the BIOS settings using Table 5-1. You can download the scripts by clicking the onps_server_1_4.tar.gz tarball. Follow the instructions listed in the README file for detailed execution steps of the scripts. Once completed, jump to section 6, Virtual Machine Deployment Using OpenStack, for the next steps.

The option of enabling Real-Time kernel on compute node is also provided in the scripts used for automated installation.

Note: The automation scripts are only tested on Fedora21 OS and will not necessarily work on other flavors of Linux.

5.2 Manual Installation Procedure

This section provides instructions on how to manually prepare both the controller and compute nodes. Although it is considered relatively easy to use this solutions guide for other Linux distributions, the preferred operating system is Fedora 21. Details on the procedure common to the compute and controller nodes can be found in section 5.2.1, OpenStack (Kilo), to help provide a good start.

Note: It is assumed that commands requiring root privileges are listed beginning with `#` and commands requiring only user privileges are listed beginning with the `$` sign.
5.2.1 Instructions Common to Controller and Compute Nodes

If you are looking to fine-tune a specific set of components either in the operating system or OpenStack, you can choose to follow the manual installation procedure. Update the BIOS settings as described in section 5.2.1.1, BIOS Settings.

5.2.1.1 BIOS Settings

During the boot time, enter the BIOS menu and update the following configuration as described in Table 5-1. These settings are common to both controller and compute nodes.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Setting for Controller Node</th>
<th>Setting for Compute Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® Virtualization Technology</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>Intel® Hyper-Threading Technology (HTT)</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>Intel® VT for Directed I/O (VT-d)</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

5.2.1.2 Operating System Installation and Configuration

The following are some generic instructions for installing and configuring the operating system. Other ways of installing the operating system, such as network installation, PXE boot installation, USB key installation, etc., are not described in this solutions guide.

Get the Fedora DVD

1. Download the 64-bit Fedora 21 DVD from the following site:
   or directly from the URL:
   [http://download.fedoraproject.org/pub/fedora/linux/releases/21/Server/x86_64/iso/Fedora-Server-DVD-x86_64-21.iso](http://download.fedoraproject.org/pub/fedora/linux/releases/21/Server/x86_64/iso/Fedora-Server-DVD-x86_64-21.iso)
2. Burn the ISO file to a DVD and create an installation disk.
Fedora 21 OS Installation

Use the DVD to install Fedora 21. During the installation, click **Software selection**, then choose the following:

1. C Development Tool and Libraries
2. Development Tools
3. Virtualization

Create a user named “stack” and check the box **Make this user administrator** during the installation. The user “stack” is also used in the OpenStack installation. Reboot the system after completing the installation.

Proxy Configuration

If your infrastructure requires you to configure the proxy server, follow the instructions in **Appendix B: Configuring the Proxy.**

Additional Packages Installation

Some packages are not installed with the standard Fedora 21 installation, but are required by Intel® Open Network Platform for Server (ONPS) components. These packages should be installed by the user:

```
$yum -y install git ntp patch socat python-passlib libxlsxlt-devel libffi-devel fuse-devel gluster python-cliff git
```

Fedora 21 Kernel Upgrade

ONPS supports Fedora kernel 3.18.8-201, which is newer than native Fedora 21 kernel 3.17.4. To upgrade from 3.17.4 to 3.18.8-201, follow these steps:

**Note:** If the Linux real-time kernel is preferred, you can skip this section and go to the section **Real Time Kernel Compute Node Enablement.**

1. Download the kernel packages:

   ```
   # wget https://kojipkgs.fedoraproject.org/packages/kernel/3.18.8/201.fc21/x86_64/kernel-core-3.18.8-201.fc21.x86_64.rpm
   # wget https://kojipkgs.fedoraproject.org/packages/kernel/3.18.8/201.fc21/x86_64/kernel-modules-3.18.8-201.fc21.x86_64.rpm
   # wget https://kojipkgs.fedoraproject.org/packages/kernel/3.18.8/201.fc21/x86_64/kernel-3.18.8-201.fc21.x86_64.rpm
   # wget https://kojipkgs.fedoraproject.org/packages/kernel/3.18.8/201.fc21/x86_64/kernel-devel-3.18.8-201.fc21.x86_64.rpm
   # wget https://kojipkgs.fedoraproject.org/packages/kernel/3.18.8/201.fc21/x86_64/kernel-modules-extra-3.18.8-201.fc21.x86_64.rpm
   ```
2. Install the kernel packages:
   
   ```
   # rpm -i kernel-core-3.18.8-201.fc21.x86_64.rpm
   # rpm -i kernel-modules-3.18.8-201.fc21.x86_64.rpm
   # rpm -i kernel-3.18.8-201.fc21.x86_64.rpm
   # rpm -i kernel-devel-3.18.8-201.fc21.x86_64.rpm
   # rpm -i kernel-modules-extra-3.18.8-201.fc21.x86_64.rpm
   ```

3. Reboot system to allow booting into 3.18.8 kernel.

**Note:** ONPS depends on libraries provided by your Linux distribution. As such, it is recommended that you regularly update your Linux distribution with the latest bug fixes and security patches to reduce the risk of security vulnerabilities in your systems.

The following command upgrades to the latest kernel that Fedora supports. In order to maintain the kernel version (3.18.8), the yum configuration file needs to be modified with this command prior to running the yum update:

   ```
   # echo "exclude=kernel*" >> /etc/yum.conf
   ```

After installing the required kernel packages, the operating system should be updated with the following command:

   ```
   # yum update -y
   ```

Reboot the system once the update is done.

**Real-Time Kernel Compute Node Enablement**

In real-time use cases, like the Telco environment, applications like media are sensitive to low latency and jitter, it makes sense to install the Linux Real-Time stable kernel on a compute node, instead of the standard Fedora kernel. This section provides the procedure for doing this.

**Note:** The option of enabling the real-time kernel on the compute node is provided in the scripts used for automated installation.

In case automated installation scripts are not used, the procedure for manually enabling the real-time kernel on the compute node is as follows:

1. Install Fedora 21 OS by selecting the following software:
   a. Virtualization
   b. C Development Tools
   c. Development Tools

2. Reboot the system after the installation is done.

3. The inbox kernel is 3.17.4-301.fc21.x86_64. Prepare the system to define the network interface by disabling the firewall and Network Manager (refer to the "Disable and Enable Services" section below for details), and then perform the following steps:
   a. Run `yum update -y` to update the system with the latest libraries.
   b. Reboot the system.

4. Install the real-time stable kernel:
   a. Get the real-time kernel sources:
      ```
      # cd /usr/src/kernels
      ```
# git clone https://www.kernel.org/pub/scm/linux/kernel/git/rt/linux-stable-rt.git

**Note:** It may take a while to complete the download.

b. Find the latest rt version from the git tag and then check out the version.

**Note:** v3.14.36-rt34 is the latest version when this document was written.

```
# cd linux-stable-rt
# git tag
# git checkout v3.14.36-rt34
```

c. Compile the RT kernel:

```
# NOTE: Reference this link:
https://rt.wiki.kernel.org/index.php/RT_PREEMPT_HOWTO
```

d. Install the necessary package:

```
# yum install ncurses-devel
```

e. Copy the kernel configuration file to the kernel source:

```
# cp /usr/src/kernel/3.17.4-301.f21.x86_64/.config /usr/src/kernel/linux-stable-rt/
# cd /usr/src/kernel/linux-stable-rt
# make menuconfig
```

The resulting configuration interface is shown below.
f. Select the following:
   i. Enable high resolution timer:
      
      General Setup > Timers Subsystem > High Resolution Timer Support
      (This option is selected by default.)

   ii. Enable max number SMP:
       
       Processor type and features > Enable Maximum Number of SMP Processor and NUMA Nodes

   iii. Enable Preempt RT:
        
        Processor type and features > Preemption Model > Fully Pre-emptible Kernel (RT)

   iv. Set high timer frequency:
        
        Processor type and features > Timer frequency > 1000 HZ
        (This option is selected by default.)

   v. Exit and save.

   vi. Compile the kernel:
        
        # make -j `grep -n processor /proc/cpuinfo` && make modules_install && make install

   vii. Make changes to the boot sequence:
        
        i. To show all menu entry
           
           # grep ^menuentry /boot/grub2/grub.cfg

        ii. To set the default menu entry:
            
            # grub2-set-default "the desired default menu entry"

        iii. To verify:
            
            # grub2-editenv list

   iv. Reboot and log to the new real-time kernel version: v3.14.36-rt34.

**Disable and Enable Services**

For OpenStack, a set of services needs to be enabled, and a few sets of services need to be disabled. The following services need to be disabled: SELinux, firewall, and NetworkManager.

Run the following commands:

```
# sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
# systemctl disable firewalld.service
# systemctl disable NetworkManager.service
```

The following services should be enabled: ntp, sshd, and network. Run the following commands:

```
# systemctl enable ntpd.service
# systemctl enable ntpdate.service
```
It is necessary to keep the timing synchronized between all nodes and to use a known Network Time Protocol (NTP) server for each of them. You can use either the standard NTP server provided by Fedora or edit /etc/ntp.conf to add a new server and remove the default ones.

The following example replaces a default NTP server with a local NTP server 10.166.45.16 and comments out other default servers:

```bash
$ sed -i 's/server 0.fedora.pool.ntp.org iburst/server 10.166.45.16/g' /etc/ntp.conf
$ sed -i 's/server 1.fedora.pool.ntp.org iburst/# server 1.fedora.pool.ntp.org iburst / g' /etc/ntp.conf
$ sed -i 's/server 2.fedora.pool.ntp.org iburst/# server 2.fedora.pool.ntp.org iburst / g' /etc/ntp.conf
$ sed -i 's/server 3.fedora.pool.ntp.org iburst/# server 3.fedora.pool.ntp.org iburst / g' /etc/ntp.conf
```

Now that the base OS is installed and the right services are configured, you can proceed to the individual node setup for either controller or compute node.

### 5.2.2 Controller Node Setup

This section describes the controller node setup. It is assumed that the user successfully followed the operating system installation and configuration sections.

#### 5.2.2.1 OpenStack (Kilo)

This section documents the configurations that need to be made and the installation of OpenStack on the controller node.

**Network Requirements**

If your infrastructure requires you to configure the proxy server, follow the instructions in Appendix B: Configuring the Proxy.

**General**

At least two networks are required to build the OpenStack infrastructure in a lab environment. One network is used to connect all nodes for OpenStack management (management network) and the other one is a private network exclusively for an OpenStack internal connection (tenant network) between instances (or VMs).

One additional network is required for Internet connectivity, because installing OpenStack requires pulling packages from various sources/repositories on the Internet.

Some users might want to have Internet and/or external connectivity for OpenStack instances (VMs). In this case, an optional network can be used.

The assumption is that the targeting OpenStack infrastructure contains multiple nodes: one is a controller node and one or more are compute nodes.
Network Configuration Example

The following is an example of how to configure networks for the OpenStack infrastructure. The example uses four network interfaces:

- **ens2f1**: For Internet network — Used to pull all necessary packages/patches from repositories on the Internet, configured to obtain a Dynamic Host Configuration Protocol (DHCP) address.

- **ens2f0**: For Management network — Used to connect all nodes for OpenStack management, configured to use network 10.11.0.0/16.

- **p1p1**: For Tenant network — Used for OpenStack internal connections for virtual machines, configured with no IP address.

- **p1p2**: For Optional External network — Used for virtual machine Internet/external connectivity, configured with no IP address. This interface is only in the controller node, if the external network is configured. For the compute node, this interface is not required.

**Note:** Among these interfaces, the interface for the virtual network (in this example, p1p1) may be an 82599 port (Niantic) or XL710 port (Fortville), because it is used for DPDK and OVS with DPDK-netdev. Also note that a static IP address should be used for the interface of the management network.

In Fedora, the network configuration files are located at: `/etc/sysconfig/network-scripts/`.

To configure a network on the host system, edit the following network configuration files:

```
ifcfg-ens2f1
DEVICE=ens2f1
TYPE=Ethernet
ONBOOT=yes
BOOTPROTO= dhcp

ifcfg-ens2f0
DEVICE=ens2f0
TYPE=Ethernet
ONBOOT=yes
BOOTPROTO=static
IPADDR=10.11.12.11
NETMASK=255.255.0.0

ifcfg-p1p1
DEVICE=p1p1
TYPE=Ethernet
ONBOOT=yes
BOOTPROTO=None

ifcfg-p1p2
DEVICE=p1p2
TYPE=Ethernet
ONBOOT=yes
BOOTPROTO=None
```
Notes: 1. Do not configure the IP address for p1p1 (10 Gb/s interface); otherwise, DPDK will not work when binding the driver during the OpenStack Neutron installation.

2. 10.11.12.11 and 255.255.0.0 are static IP address and net mask to the management network. It is necessary to have a static IP address on this subnet. (The IP address 10.11.12.11 is used here only as an example.)

5.2.2.2 OpenStack Installation Procedures

General
DevStack provides and maintains tools for the installation of OpenStack from upstream sources. Its main purpose is for OpenStack development and testing of involved components. Due to its evolving nature, DevStack does not provide production-level stability. The Intel ONP server chooses DevStack for quick deployment of OpenStack in order to use its latest features, including, at times, its experimental ones. Given this, volatility may be introduced.

The following procedure uses actual examples of an OpenStack (DevStack) installation performed in an Intel test lab. It consists of one controller node (controller) and one compute node (compute).

Controller Node Installation Procedures
The following example uses a host for the controller node installation with the following:

- Hostname: sdnlab-k01
- Internet network IP address: Obtained from the DHCP server
- OpenStack Management port and IP address: ens2f0, 10.11.12.11
- Data plane network port: p1p1
- User/password: stack/stack

Root User Actions
Log in as root user and follow these steps:

1. Add stack user to sudoers list if not already:
   
   ```
   #echo "stack ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
   ```

2. Edit `/etc/libvirt/qemu.conf`.
3. Restart libvirt service and make sure libvird is active.

Stack User Actions
1. Log in as stack user.
2. Configure the appropriate proxies (yum, http, https, and git) for the package installation, and make sure these proxies are functional. Note that on controller node, localhost, and its IP address should be included in the no_proxy setup (e.g., `export no_proxy=localhost, 10.11.12.11`). For detailed instructions how to set up your proxy, refer to Appendix B: Configuring the proxy.
3. Download the DevStack source:
   
   ```
   git clone https://github.com/openstack-dev/devstack.
   ```
4. Use OpenStack with stable/kilo tag:
   
   ```
   $cd /home/stack/devstack/
   $git checkout stable/kilo
   ```

5. Create the `local.conf` file in `/home/stack/devstack/`.

6. Pay attention to the following in the `local.conf` file:
   a. Use Rabbit for messaging services (Rabbit is on by default). In the past, Fedora only supported QPID for OpenStack. Now it only supports Rabbit.
   
   ```
   disable_service n-cpu
   ```

   b. Explicitly disable the Nova compute service on the controller. By default the service is enabled.
   
   ```
   explicitly disable the Nova compute service on the controller. By default the service is enabled.
   ```

   c. To use OVS, specify in the configuration for the ML2 plug-in:
   
   ```
   Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch
   ```

   d. Explicitly disable tenant tunneling and enable tenant VLAN. This is because by default tunneling is enabled:
   
   ```
   ENABLE_TENANT_TUNNELS=False
   ENABLE_TENANT_VLANS=True
   ```

   e. Explicitly enable VNC and related services. These services were previously enabled by default, but are now disabled by default:
   
   ```
   enable_service n-novnc
   enable_service n-xvnc
   enable_service n-crt
   enable_service n-cauth
   ```

   f. A sample `local.conf` file for the controller node is as follows:
   
   ```
   # Controller node
   #
   [[[local][localrc]]]
   FORCE=yes
   HOST_NAME=$(hostname)
   HOST_IP=10.11.12.11
   HOST_IP_IFACE=ens2f0
   PUBLIC_INTERFACE=p1p2
   VLAN_INTERFACE=p1p1
   FLAT_INTERFACE=p1p1
   ADMIN_PASSWORD=password
   MYSQL_PASSWORD=password
   DATABASE_PASSWORD=password
   SERVICE_PASSWORD=password
   SERVICE_TOKEN=no-token-password
   HORIZON_PASSWORD=password
   RABBIT_PASSWORD=password
   ```
enable_service q-meta
enable_service neutron
enable_service n-novnc
enable_service n-xvnc
enable_service n-crt
enable_service n-cauth
LOGFILE=$DEST/stack.sh.log
SCREEN_LOGDIR=$DEST/screen
SYSLOG=True
LOGDAYS=1
Q_AGENT=openvswitch
Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch
Q_ML2_PLUGIN_TYPE_DRIVERS=flat,local
Q_ML2_TENANT_NETWORK_TYPE=flat
ENABLE_TENANT_TUNNELS=False
ENABLE_TENANT_VLANS=True
PHYSICAL_NETWORK=physnet1
ML2_VLAN_RANGES=physnet1:1000:1010
OVS_PHYSICAL_BRIDGE=br-p1p1
MULTI_HOST=True
[[post-config|$NOVA_CONF]]
#disable nova security groups
[DEFAULT]
firewall_driver=nova.virt.firewall.NoopFirewallDriver
novncproxy_host=0.0.0.0
novncproxy_port=6080

7. **Install DevStack:**

   $ cd /home/stack/devstack/
   $ ./stack.sh

8. **The following appears at the end of the screen output to indicate a successful installation:**

   “This is your host ip: 10.11.12.11”
   “Horizon is now available at http://10.11.12.11.”

9. **Add physical port(s) to the bridge(s) created by the DevStack installation.**

   The following example can be used to configure the two bridges: br-p1lp1 (for the virtual network) and br-ex (for the external network):

   ```bash
   sudo ovs-vsctl add-port br-p1lp1 p1lp1
   sudo ovs-vsctl add-port br-ex p1p2
   ```

10. **Make sure the proper VLANS are created in the switch connecting physical port p1p1. For example, the previous local.conf specifies VLAN range of 1000 to 1010; therefore, matching VLANS 1000 to 1010 should also be configured in the switch.**

### 5.2.3 Compute Node Setup

This section describes how to complete the setup of the compute nodes. It is assumed that the user has successfully completed the BIOS settings and the operating system installation and configuration sections.
5.2.3.1 Host Configuration

Using DevStack to Deploy vSwitch and OpenStack Components

General

Deploying OpenStack and OpenvSwitch with DPDK-netdev using DevStack on a compute node follows the same procedures as on the controller node. Differences include:

- Required services are nova compute and neutron agent
- OpenvSwitch with DPDK-netdev, used in place of OpenvSwitch for neutron agent.

Compute Node Installation Example

The following example uses a host for the compute node installation with the following:

- **Hostname**: sdnlab-k02
- **Internet network IP address**: Obtained from DHCP server
- **OpenStack Management IP address**: ens2f0, 10.11.12.12
- **Data plane network port**: p1p1
- **User/password**: stack/stack

Note the following:

- **No_proxy setup**: Localhost and its IP address should be included in the no_proxy setup. The hostname and IP address of the controller node should also be included. For example:
  ```
  export no_proxy=localhost,10.11.12.12,sdnlab-k01,10.11.12.11
  ```

  Refer to Appendix B: Configuring the Proxy for more details about setting up the proxy.

- **Differences in the local.conf file**:
  
  - The service host is the controller, as well as other OpenStack services, such as database, messaging, authentication, and image. The service host should be spelled out in the local.conf of compute node. Using the controller node example in the previous section, the service host and its IP address should be:
    ```
    SERVICE_HOST_NAME=sdnlab-k01
    SERVICE_HOST=10.11.12.11
    ```

  - The only OpenStack services required in compute nodes are nova compute and neutron agent, so the local.conf might look like:
    ```
    disable_all_services
enable_service n-cpu
enable_service q-agt
    ```

  - Openvswitch is used for the neutron agent and neutron ML2 plugin driver:
    ```
    Q_AGENT=openvswitch
    Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch
    ```

  - Binding the physical port to the bridge is through the following line in local.conf. For example, to bind port p1p1 to bridge br-p1p1, use:
    ```
    OVS_PHYSICAL_BRIDGE=br-p1p1
    ```
For the OVS with DPDK-netdev setting, OVS-DPDK repository for OpenStack, DPDK version, OVS git tag, huge pages setting should be added:

```bash
enable_plugin networking-ovs-dpdk https://github.com/stackforge/networking-ovs-dpdk
OVS_DPDK_RTE_LIBRTE_VHOST=n
Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch,ovsdpdk
OVS_NUM_HUGE_PAGES=8192
OVS_DPDK_MEM_SEGMENTS=8192
OVS_HUGEPAGE_MOUNT_PAGESIZE=2M
OVS_DPDK_GIT_TAG=v1.8.0
OVS_GIT_TAG=b8e57534ece5f620af7d7fa0278c8e9746dd719c
```

A sample local.conf file for compute node with OVS with DPDK-netdev agent is as follows:

```bash
# Compute node
# OVS_TYPE=ovs-dpdk
[[local|localrc]]
FORCE=yes
MULTI_HOST=True
HOST_NAME=$(hostname)
HOST_IP=10.11.12.12
HOST_IP_IFACE=ens2f0
SERVICE_HOST_NAME=10.11.12.11
SERVICE_HOST=10.11.12.11
MYSQL_HOST=$SERVICE_HOST
RABBIT_HOST=$SERVICE_HOST
GLANCE_HOST=$SERVICE_HOST
GLANCE_HOSTPORT=$SERVICE_HOST:9292
KEYSTONE_AUTH_HOST=$SERVICE_HOST
KEYSTONE_SERVICE_HOST=10.11.12.11
ADMIN_PASSWORD=password
MYSQL_PASSWORD=password
DATABASE_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=no-token-password
HORIZON_PASSWORD=password
RABBIT_PASSWORD=password
disable_all_services
enable_service n-cpu
enable_service q-agt
DEST=/opt/stack
logfile=$DEST/stack.sh.log
SCREEN_LOGDIR=$DEST/screen
SYSLOG=True
LOGDAYS=1
OVS_AGENT_TYPE=openvswitch
Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch,ovsdpdk
Q_ML2_PLUGIN_TYPE_DRIVERS=vlan
enable_plugin networking-ovs-dpdk https://github.com/stackforge/networking-ovs-dpdk
OVS_DPDK_RTE_LIBRTE_VHOST=n
```
OVS_GIT_TAG=b8e57534ece5f620af7d7fa0278c8e9746dd719c
OVS_DPDK_GIT_TAG=v1.8.0
OVS_DATAPATH_TYPE=netdev
OVS_NUM_HUGE_PAGES=8192
OVS_DPDK_MEM_SEGMENTS=8192
OVS_HUGE_PAGE_MOUNT_PAGESIZE=2M
ENABLE_TENANT_TUNNELS=False
ENABLE_TENANT_VLANS=True
Q_ML2_TENANT_NETWORK_TYPE=vlan
ML2_VLAN_RANGES=physnet1:1000:1010
PHYSICAL_NETWORK=physnet1
OVS_PHYSICAL_BRIDGE=br-ppl

[[post-config|$NOVA_CONF]]
[DEFAULT]
firewall_driver=nova.virt.firewall.NoopFirewallDriver
vnc_enabled=True
vncserver_listen=0.0.0.0
vncserver_proxyclient_address=10.11.12.12
[libvirt]
cpu_mode=host-model

— A sample local.conf file for compute node with Regular OVS agent follows:

# Compute node
# OVS_TYPE=ovs
[[local|localrc]]
FORCE=yes
MULTI_HOST=True
HOST_NAME=$(hostname)
HOST_IP=10.11.12.12
HOST_IP_IFACE=ens2f0
SERVICE_HOST_NAME=10.11.12.11
SERVICE_HOST=10.11.12.11
MYSQL_HOST=$SERVICE_HOST
RABBIT_HOST=$SERVICE_HOST
GLANCE_HOST=$SERVICE_HOST
GLANCE_HOSTPORT=$SERVICE_HOST:9292
KEYSTONE_AUTH_HOST=$SERVICE_HOST
KEYSTONE_SERVICE_HOST=10.11.12.11
ADMIN_PASSWORD=password
MYSQL_PASSWORD=password
DATABASE_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=no-token-password
HORIZON_PASSWORD=password
RABBIT_PASSWORD=password
disable_all_services
enable_service n-cpu
enable_service q-agt
DEST=/opt/stack
LOGFILE=$DEST/stack.sh.log
SCREEN_LOGDIR=$DEST/screen
SYSLOG=True
LOGDAYS=1
Q_AGENT=openvswitch
Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch
Q_ML2_PLUGIN_TYPE_DRIVERS=_vlan
ENABLE_TENANT_TUNNELS=False
ENABLE_TENANT_VLANs=True
Q_ML2_TENANT_NETWORK_TYPE=vlan
ML2_VLAN_RANGES=physnet1:1000:1010
PHYSICAL_NETWORK=physnet1
OVS_PHYSICAL_BRIDGE=br-p1p1

[[post-config|$NOVA_CONF]]
[DEFAULT]
firewall_driver=nova.virt.firewall.NoopFirewallDriver
vnc_enabled=True
vncserver_listen=0.0.0.0
vncserver_proxyclient_address=10.11.12.12
[libvirt]
cpu_mode=host-model

— For regular OVS compute node, add the physical port(s) to the bridge(s) created by the DevStack installation. The following example can be used to configure the bridges: br-p1pl for the virtual network:

$sudo ovs-vsctl add-port br-p1p1 p1p1
6.0 Virtual Machine Deployment Using OpenStack

This section describes how to bring up the VMs in the OpenStack environment, deploy various advanced features of OpenStack, and integrate this with the network controller, OpenDaylight.

**Note:** It is assumed that commands that require root privileges are listed beginning with `#` and those that require only user privileges are listed beginning with the `$` sign.

6.1 Preparation with OpenStack

6.1.1 Deploying Virtual Machines

6.1.1.1 Default Settings

OpenStack comes with the following default settings:

- Tenant (Project): admin and demo
- Network:
  - Private network (virtual network): 10.0.0.0/24
  - Public network (external network): 172.24.4.0/24
- Image: cirros-0.3.1-x86_64
- Flavor: nano, micro, tiny, small, medium, large, and xlarge

To deploy new instances (VMs) with different setups (e.g., different VM image, flavor, or network) users must create custom configurations. OpenStack provides a command-line interface and graphic interface (Horizon) for this purpose. In this section, you will be shown how to use OpenStack commands to create VMs with custom settings. For the same functionalities using a graphic interface, refer to Appendix C: Configuring Horizon UI to Deploy Virtual Machines.

To access the OpenStack dashboard, use a web browser (Firefox, Internet Explorer, or others) and controller's IP address (management network), for example, `http://10.11.12.11/`.

Login information is defined in the `local.conf` file. In the examples that follow, `password` is the password for both admin and demo users.
6.1.1.2 Manual Deployment with Custom Settings

The following examples describe how to create a VM image with custom options like flavor, and aggregate/availability zone using OpenStack commands. The examples assume the IP address of the controller is 10.11.12.11.

1. Log in as stack user.

**Notes:**

1. Some OpenStack commands (e.g., keystone and aggregate-related) can only be used by the admin user; while others (e.g., glance, nova, and those that are neutron-related) can be used by other users, but with limited visibility.

2. DevStack provides a built-in tool, openrc, located at /home/stack/devstack/, to source an OpenStack user credential to the shell environment.

2. Source the admin credential:

   ```sh
   $ source /home/stack/devstack/openrc admin
   ``

   **Note:** OpenStack commands will thereafter use the admin credential.

3. Create an OpenStack glance image. (Glance is the OpenStack component that manages VM images.) A VM image file should be ready in a location accessible by OpenStack. The following command is from an existing image file and is a template for creating new VMs:

   ```sh
   $glance image-create --name <image-name-to-create> --is-public=true --container-format=bare --disk-format=<format> --file=<image-file-path-name>
   ``

   The following example shows the image file, fedora20-x86_64-basic.qcow2, and is located in a NFS share and mounted at /mnt/nfs/openstack/images/ to the controller host. The command creates a glance image named “fedora-basic” with a qcow2 format for public use that any tenant can use:

   ```sh
   $glance image-create --name fedora-basic --is-public=true --container-format=bare --disk-format=qcow2 --file=/mnt/nfs/openstack/images/fedora20-x86_64-basic.qcow2
   ``

4. Create the host aggregate and availability zone:

   First find out the available hypervisors, and then use this information for creating an aggregate/availability zone:

   ```sh
   $nova hypervisor-list
   $nova aggregate-create <aggregate-name> <zone-name>
   $nova aggregate-add-host <aggregate-name> <hypervisor-name>
   ``

   The following example creates an aggregate named aggr-g06 with one availability zone named zone-g06, and the aggregate contains one hypervisor named sdnlab-g06:

   ```sh
   $nova aggregate-create aggr-g06 zone-g06
   $nova aggregate-add-host aggr-g06 sdnlab-g06
   ``

5. Create flavor. Flavor is a virtual hardware configuration for the VMs; it defines the number of virtual CPUs, size of the virtual memory, and disk space, etc.

   The following command creates a flavor named “onps-flavor” with an ID of 1001, 1024 Mb virtual memory, 4 Gb virtual disk space, and 1 virtual CPU:

   ```sh
   $nova flavor-create onps-flavor 1001 1024 4 1
   ``

6. Source the demo user credential. Note that OpenStack commands will continue to use this demo credential:

   ```sh
   $source /home/stack/devstack/openrc demo
   ```
7. Create a network for the tenant demo. Perform the following steps:
   a. Create a tenant network
      
      $neutron net-create <network-name>
      
      The following creates a network with a name of net-demo for tenant demo (because demo credential is used):
      
      $neutron net-create net-demo
   
      b. Create a subnet:
      
      $neutron subnet-create --name <subnet_name> <network-name> <net-ip-range>
      
      The following creates a subnet with a name of sub-demo and CIDR address 192.168.2.0/24 for network net-demo:
      
      $neutron subnet-create --name sub-demo net-demo 192.168.2.0/24

8. Create an instance (VM) for tenant demo. Take the following steps:
   a. Get the name and/or ID of the image, flavor, and availability zone to be used for creating the instance:
      
      glance image-list
      nova flavor-list
      nova availability-zone-list
      neutron net-list
      
      b. Launch the instance (VM), using information from the previous step.
      
      $nova boot --image <image-id> --flavor <flavor-id> --availability-zone <zone-name> --nic net-id=<network-id> <instance-name>
      
      The new VM should be up and running in a few minutes.

9. Log in to the OpenStack dashboard using the demo user credential; click **Instances** under Project in the left pane, the new VM should show in the right pane. Click the instance name to open the **Instance Details** view, and then click **Console** in the top menu to access the VM as show below.
6.2 Non-Uniform Memory Access (NUMA) Placement and SR-IOV Passthrough for OpenStack

NUMA placement was implemented as a new feature in Openstack Juno release. NUMA placement allows the Openstack administrator to pin a particular NUMA node for guest system optimization. With the SR-IOV (single-root I/O virtualization) enabled network interface card, each SR-IOV port is associated with a virtual function (VF). OpenStack SR-IOV passthrough allows guests to access VF directly.

Both the controller and OVS-based compute nodes have to be set up after a fresh install of the OS prior to starting the following use case.

**Notes:** It is recommended to reset the BIOS, configure the required settings in BIOS, have a clean installation of OS, and then run the ONP server scripts to test the NUMA placement and SR-IOV passthrough for OpenStack.

6.2.1 Create Flavor for NUMA Placement

To use the NUMA placement feature in Openstack, the user needs to first create a flavor, and then modify it for NUMA placement. This procedure has to be performed on a standard OVS-based compute node.

To enable the NUMA placement feature, *libvirtd* should be upgraded using the Fedora virt-preview repository. To do this, perform the following steps:

```
# cd /etc/yum.repos.d/
# wget http://fedorapeople.org/groups/virt/virt-preview/fedora-virt-preview.repo
# yum update
# systemctl restart libvirtd
# libvirtd --version
```

After stacking is successful on both the controller and compute nodes, create a flavor. The following example creates a numa-flavor:

```
$ nova flavor-create numa-flavor 1001 1024 4 1
```

where

- flavor name = numa-flavor
- id = 1001
- virtual memory = 1024 Mb
- virtual disk size = 4 Gb
- number of virtual CPU = 1

Then modify the numa-flavor for NUMA placement on the compute node:

```
$ nova flavor-key 1001 set numa_nodes=1 hw:numa_cpus.0=0 hw:numa_mem.0=1024
```

To show detailed information for this flavor:

```
$ nova flavor-show 1001
```
6.2.2 Prepare Compute Node for SR-IOV Passthrough

To enable SR-IOV passthrough, follow the next steps to configure the compute node:

1. For the Wildcat Pass server, make sure the BIOS version installed is SE5C610.86B.01.01.0008.031920151331 or newer by following these commands:
   a. Use this command to check the BIOS version:
      
      ```
      # dmidecode | grep "BIOS\|Version"
      ```
      (Find the BIOS version from the output.)
   b. If the BIOS version is lower, download the BIOS upgrade package from the Intel support site with the link below:
      
      ```
      ```
      To download, click "Accept terms and license agreement" at the bottom of the page.
   c. Follow these steps to upgrade the BIOS:
      i. Unzip the downloaded file to a USB key; insert the USB key in the intended system.
      ii. Reboot the system; it will take you to the UEFI shell and update the utility.
      iii. Press any key, except ‘q’ to continue.
      iv. Select 1 for updating SRD.
      v. Press any key, except ‘q’ to continue.
      vi. Type ‘exit’; the system will boot into the existing OS.
      vii. Reboot.
      viii. During boot time, press F2 to enter BIOS, and then verify the new BIOS version is correctly installed.

2. Enable Intel VT-d in the BIOS. By default, VT-d is disabled on the Wildcat Pass server. To enable VT-d, follow these steps:
   a. Enter BIOS during system boot.
   b. Select Advanced, and then "Integrated IO Configuration."
   c. Select "Intel(R) VT for Direct IO" and enable.
   d. Reboot.

3. Enable kernel IO_MMU support in grub. For Fedora, do the following:
   a. Add "intel_iommu=on" to /etc/default/grub on GRUB_CMDLINE_LINUX:
      
      ```
      # sed -i 's/rhgb quiet/rhgb quite intel_iommu=on/g' /etc/default/grub
      ```
      Reboot the system prior to proceeding to the next step.
   b. Update the grub configuration file grub.cfg:
      
      ```
      # grub2-mkconfig -o /boot/grub2/grub.cfg
      ```
   c. Reboot.
4. Verify that the system supports input/output memory management unit (IOMMU). After rebooting, run the command:

```bash
$ dmesg | grep -e DMAR -e IOMMU
```

The output should be similar to the following:

```
[    0.000000] Intel-IOMMU: enabled
[    0.130879] dmar: IOMMU 0: reg_base_addr fbffc000 ver 1:0 cap d2078c106f0466
cap f020de
[    0.130886] dmar: IOMMU 1: reg_base_addr c7ffc000 ver 1:0 cap d2078c106f0466
cap f020de
[    0.131028] IOAPIC id 10 under DRHD base 0xfbffc000 IOMMU 0
[    0.131029] IOAPIC id 8 under DRHD base 0xc7ffc000 IOMMU 1
[    0.131030] IOAPIC id 9 under DRHD base 0xc7ffc000 IOMMU 1
```

5. Upgrade libvirt using the Fedora virt-preview repository (refer to section 6.2.1, Create Flavor for NUMA Placement, for a description of the procedures).

6. Modify `/etc/libvirt/qemu.conf`, add `/dev/vfio/vfio` to the existing `cgroup_device_acl` list.

   An example is given below:

   ```
```

7. Restart libvirtd and check if the version is 1.2.13.1:

   ```bash
   # systemctl restart libvirtd
   # libvirtd --version
   ```

8. Make sure of the following:

   a. Two Fortville (or Niantic) ports are available. One of the ports is used for the tenant network (for OVS), and the other is used for SR-IOV and NUMA placement (e.g., port ens786f0 is used for OVS and port ens786f1 for SR-IOV).

   b. The NUMA node is in place for the pci device; confirm the expected output is a mixture of 0, 1, -1:

   ```bash
   # cat /sys/bus/pci/devices/*/numa_node
   ```

9. Enable the SR-IOV VF for the designated port. The following example enables 1 VF for port ens786f1:

   ```bash
   # echo 1 > /sys/class/net/ens786f1/device/sriov_numvfs
   ```

   The following command can be used to display the PCI device address, PCI vendor ID, and product ID.

   For Fortville:
   ```bash
   $ lspci -nn | grep X710
   ```

   For Niantic:
   ```bash
   $ lspci -nn | grep 82599
   ```

   The example below shows that one VF was created:

   ```bash
   $ lspci -nn | grep X710
   ```
06:00.0 Ethernet controller [0200]: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ [8086:1572] (rev 01)
06:00.1 Ethernet controller [0200]: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ [8086:1572] (rev 01)
06:00.2 Ethernet controller [0200]: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ [8086:1572] (rev 01)
06:00.3 Ethernet controller [0200]: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ [8086:1572] (rev 01)
06:06.0 Ethernet controller [0200]: Intel Corporation XL710/X710 Virtual Function [8086:154c] (rev 01)

6.2.3 DevStack Configurations

In the following example, the PCI device vendor ID (8086) and product ID of the Fortville can be obtained from the output of this command (1572 for the physical function and 154c for virtual function):

```
$ lspci -nn | grep X710
```

**On Controller Node:**

1. Edit an existing controller local.conf. Note that a similar local.conf file of section 5.2, Manual Installation Procedure, is used here along with adding the following:

```
Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch,sriovnicswitch
[[post-config|$NOVA_CONF]]
(DEFAULT)

scheduler_default_filters=RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter
pci_alias="name":"fortville","product_id":"154c","vendor_id":"8086"
```

```
[[post-config|/$Q_PLUGIN_CONF_FILE]]
[ml2_sriov]
supported_pci_vendor_devs = 8086:1572,8086:154c
agent_required=False
[sriov_nic]
physical_device_mappings=physnet1:ens786f1
```

**Notes:**

1. ens786f1 is an example of a SR-IOV port.
2. Use double escape \ in front of each quotation marks in pci_alias.
3. Run ./stack.sh.

**On Compute Node:**

1. Edit the compute local.conf. Note that the same local.conf file of section 5.2 is used here.
2. Add the following:

```
[[post-config|$NOVA_CONF]]
(DEFAULT)

pci_passthrough_whitelist="address":"0000:06:00.1","vendor_id":"
```
Intel® ONP Reference Architecture
Solutions Guide

8086", "physical_network": "physnet1"
pci_passthrough_whitelist=[{"address": "0000:06:06.0", "vendor_id": "8086", "physical_network": "physnet1"}]

Notes:
1. This specifies a Fortville port VF at PCI address 06:06.0 binding to PF at 06.00.1 for passthrough.
2. Use one port for SR-IOV and another for the tenant network (OVS); using the same port for both SR-IOV and OVS does not work.
3. Use double escape \ in front of each quotation marks in the pci_passthrough_whitelist.

3. Run ./stack.sh for compute nodes to complete the DevStack installation.

Sample local.conf files for the both controller and compute nodes are listed in Appendix A2: Sample Local.conf Files for SR-IOV Configurations.

6.2.4 Create VM with NUMA Placement and SR-IOV

1. After stacking is successful on both the controller and compute nodes, verify the PCI pass-through device(s) are in the OpenStack database. In the example below, the IP address of the controller is 10.11.12.11:
   # mysql -uroot -ppassword -h 10.11.12.11 nova -e "select * from pci_devices"

   The output should show entry(ies) of PCI device(s) similar to the following:
   | 1 | 1 |
   | 2015-04-27 17:01:14 | NULL | NULL |       0 |
   | 2 | 1 |
   | 0000:06:06.0 | 154c | 8086 | type-VF | pci_0000_06_0 | label 8086_154c | available | {"phys_function": "0000:06:00.1"} | NULL | NULL |       0 |

2. The following example uses a “demo” user credential to create a network, sriovNet, bind SR-IOV port to this network. The example also shows screen outputs (in different fonts) for each command used:
   # neutron net-create --provider:physical_network=physnet1 --provider:
   network_type=vlan sriovNet

   Created a new network:

   +---------------------------------------------+-------------------+
   | Field                     | Value                                |
   +---------------------------------------------+-------------------+
   | admin_state_up            | True                                 |
   | id                        | ba81ddd4-4a78-4e6b-b93c-995bb3080dbf |
   | mtu                       | 0                                    |
   | name                      | sriovNet                             |
   | provider:network_type     | vlan                                 |
   | provider:physical_network | physnet1                             |
   | provider:segmentation_id  | 1002                                 |
   | router:external           | False                                |
   | shared                    | False                                |
   | status                    | ACTIVE                               |
   | subnets                   |                                     |
# neutron subnet-create --name sriovSubnet sriovNet 192.168.1.0/24

Created a new subnet:

+---------------------------------+-------------------+
| Field                          | Value             |
+---------------------------------+-------------------+
| allocation_pools               | "start": "192.168.1.2", "end": "192.168.1.254" |
| cidr                           | 192.168.1.0/24    |
| dns_nameservers                |                   |
| enable_dhcp                    | True              |
| gateway_ip                     | 192.168.1.1       |
| host_routes                    |                   |
| id                             | 2cef692b-5618-4236-83f1-2960316a8740               |
| ip_version                     | 4                 |
| ipv6_address_mode              |                   |
| ipv6_ra_mode                   |                   |
| name                           | sriovSubnet       |
| network_id                     | ba81ddd4-ea78-4e6b-b93c-995bb3080dbf                |
| subnetpool_id                  |                   |
| tenant_id                      | fba8e98a2f4f4b75a28ba2ca7d0acdd9                   |
+---------------------------------+-------------------+
# neutron port-create sriovNet --binding:vnic-type direct --name sriovPort1

Created a new port:

+---------------------------------+-------------------+
| Field                          | Value             |
+---------------------------------+-------------------+
<p>| admin_state_up                 | True              |
| allowed_address_pairs          |                   |
| binding:vnic_type              | direct            |
| device_id                      |                   |
| device_owner                   |                   |
| fixed_ips                      |                   |
| id                             | 5c3b49d2-fb40-49ad-a126-819ec1233e00                  |
| mac_address                    | fa:16:3e:67:1d:7f |
| name                           | sriovPort1        |
| network_id                     | ba81ddd4-ea78-4e6b-b93c-995bb3080dbf                |</p>
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>security_groups</td>
<td>5a2b7fbf-6fc6-4509-8dab-f85db9abd50c</td>
</tr>
<tr>
<td>status</td>
<td>DOWN</td>
</tr>
<tr>
<td>tenant_id</td>
<td>fba8e98a2f4f4b75a28ba2ca7d0acdd9</td>
</tr>
</tbody>
</table>

# neutron port-show --name sriovPort1

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>sriovPort1</td>
</tr>
<tr>
<td>id</td>
<td>5c3b49d2-fb40-49ad-a126-819ec1233e00</td>
</tr>
<tr>
<td>mac_address</td>
<td>fa:16:3e:67:1d:7f</td>
</tr>
<tr>
<td>network_id</td>
<td>ba81ddd4-ea78-4e6b-b93c-995bb3080dbf</td>
</tr>
<tr>
<td>security_groups</td>
<td>5a2b7fbf-6fc6-4509-8dab-f85db9abd50c</td>
</tr>
<tr>
<td>status</td>
<td>DOWN</td>
</tr>
<tr>
<td>tenant_id</td>
<td>fba8e98a2f4f4b75a28ba2ca7d0acdd9</td>
</tr>
</tbody>
</table>
3. Create a VM instance, `sriovVM1`, using the SR-IOV port under the default project “demo”. Note that the example below assumes the flavor (numa-flavor), image (fedora20-i40e), and availability zone (zone-wp36) are already in place. Also note that the port, `sriovPort1`, was created with the procedures described above.

```bash
# nova boot --flavor numa-flavor --image fedora20-i40e --availability-zone zone-wp36 --nic port-id=5c3b49d2-fb40-49ad-a126-819ec1233e00 sriovVM1
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS-DCF:diskConfig</td>
<td>MANUAL</td>
</tr>
<tr>
<td>OS-EXT-AZ:availability_zone</td>
<td>nova</td>
</tr>
<tr>
<td>OS-EXT-STS:power_state</td>
<td>0</td>
</tr>
<tr>
<td>OS-EXT-STS:task_state</td>
<td>scheduling</td>
</tr>
<tr>
<td>OS-EXT-STS:vm_state</td>
<td>building</td>
</tr>
<tr>
<td>OS-SRV-USG:launched_at</td>
<td>-</td>
</tr>
<tr>
<td>OS-SRV-USG:terminated_at</td>
<td>-</td>
</tr>
<tr>
<td>accessIPv4</td>
<td></td>
</tr>
<tr>
<td>accessIPv6</td>
<td></td>
</tr>
<tr>
<td>adminPass</td>
<td>3bVys7z9qNSU</td>
</tr>
<tr>
<td>config_drive</td>
<td></td>
</tr>
<tr>
<td>created</td>
<td>2015-05-04T14:51:47Z</td>
</tr>
<tr>
<td>flavor</td>
<td>numa-flavor (1001)</td>
</tr>
<tr>
<td>hostId</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>59165974-fc4e-44a9-90f0-e073aae154cd</td>
</tr>
<tr>
<td>image</td>
<td>fedora20-i40e (e4094132-8a66-4b89-866f-5ebc7af6a681)</td>
</tr>
<tr>
<td>key_name</td>
<td>-</td>
</tr>
<tr>
<td>metadata</td>
<td>{}</td>
</tr>
<tr>
<td>name</td>
<td>sriovVM1</td>
</tr>
<tr>
<td>os-extended-volumes:volumes_attached</td>
<td>[]</td>
</tr>
</tbody>
</table>
When the VM is up and running, an Ethernet controller, Intel Corporation Device 154c, should be available. Run the command "lspci" to display it.

4. Note that the virtual machine image should include a proper network driver with VF support. If the VF driver is not installed, the SR-IOV VF device will not present itself to the OS. For example, the default Fedora 20 installation does not support Fortville; therefore, the `ifconfig` command will not display the Ethernet device 154c. In this case, the user should download and install the i40evf driver. The public download link is at:

http://sourceforge.net/projects/e1000/files/i40evf%20stable/1.0.5/

5. Install the VF driver. Copy the tar file, i40evf-1.0.5.tar.gz, to your VM. To install the i40evf driver, run the following commands in the VM:

```
# tar i40evf-1.0.5.tar.gz
# cd i40evf-1.0.5/src
# make
# make install
# modprobe i40evf
# reboot
```

6. Reboot the VM. A new network interface (the interface for SR-IOV VF) should show up with a name of `ensX`, where `X` is a numerical number (e.g., `ens4`). This interface will use the IP address of the corresponding network port linked to the SR-IOV VF (in the example above, the IP address is `192.168.1.6`).

7. To verify network connectivity through a VF, users can set up two compute hosts and create a VM on each node. After obtaining IP addresses, the VMs should communicate with each other similar to a normal network.
6.3 CPU Pinning for OpenStack

This feature was newly introduced in the OpenStack Kilo release. CPU pinning was developed to improve the libvirt driver for strictly pinning guest vCPUs to host pCPUs. This allows “dedicated CPU” guest instances.

Notes: It is recommended to have a clean installation of OS, and then run the ONP server scripts to test the CPU pinning for OpenStack.

6.3.1 Prepare Compute Node for CPU Pinning

To enable the CPU pinning feature, libvirtd should be upgraded with the Fedora virt-preview repository:

```
# cd /etc/yum.repos.d/
# wget http://fedorapeople.org/groups/virt/virt-preview/fedora-virt-preview.repo
# yum update
# systemctl restart libvirtd
# libvirtd --version
```

6.3.2 Create Flavor with CPU Pinning

To use the CPU pinning feature in OpenStack, you first need to create a flavor and then modify it for the CPU policy and CPU thread policy. Currently, the CPU threads policy has not been implemented and, therefore, only the CPU policy is available for the CPU pinning feature. Two options are available for the CPU policy:

1. Shared: default vCPU placement, vCPU free float across host pCPUs
2. Dedicated: vCPU pin to a set of host pCPUs

After stacking is successful on both controller and compute nodes, create a flavor. The following example is for a pin-flavor:

```
$ nova flavor-create pin-flavor 1001 1024 4 1
```

where

- flavor name = pin-flavor
- id = 1001
- virtual memory = 1024 Mb
- virtual disk size = 4Gb
- number of virtual CPU = 1

Then modify the flavor for the CPU pinning. The following example sets the CPU policy to “dedicated”:

```
$ nova flavor-key 1001 set hw:cpu_policy=dedicated
```

To show detailed information of the flavor:

```
$ nova flavor-show 1001
```
6.3.3 Create VM with CPU Pinning

Create a VM, pin-vm1, with the pin-flavor under the default project "demo". Note that the example below assumes that an image, fedora-basic, and an availability zone, zone-04, are already in place, and the "demo" project default network, private, is used.

$nova boot --flavor pin-flavor --image fedora-basic --availability-zone zone-k04 --nic net-id=<network-id of network private> pin-vm1

where

vm name = pin-vm1
image name = fedora-basic
flavor-name = pin-flavor
availability-zone = zone-k04
tenant network ID: ID of network private can be obtained by running command "neutron net-list"

6.4 Using OpenDaylight

This section describes how to install and use OpenDaylight Controller in Openstack environment.

Note: It is recommended to have a clean installation of OS and then run the ONP server scripts to test the CPU Pinning for OpenStack.

6.4.1 Preparing the OpenDaylight Controller

A new initiative in developing a plugin library for OpenDaylight is the ML2 Mechanism Driver to handle communication between OpenStack and OpenDaylight. With this plugin, it is not necessary to download and separately install OpenDaylight. In DevStack, OpenDaylight installation is specified in the local.conf file.

For Fedora 21, the native Java version from the yum repository is Java v1.8. OpenDaylight Helium SR3 (the current released version), however, only supports Java v1.7. It is necessary, therefore, to manually install Java 1.7 on Fedora 21 system and make it the default Java version:

1. Download the rpm file from the Oracle website:

   # curl -v -j -k -L -H "Cookie: oraclelicense=accept-securebackup-cookie" http://download.oracle.com/otn-pub/java/jdk/7u75-b13/jdk-7u75-linux-x64.rpm > jdk-7u75-linux-x64.rpm

2. Install Java v1.7:

   # rpm -ivh jdk-7u75-linux-x64.rpm

3. Set Java v1.7 as the default version and verify. Note that Java v1.7 was installed at /usr/java/.

   # alternatives --install /usr/bin/java java /usr/java/default/jre/bin/java 2
   # echo 2 | alternatives --config java
   # java -version

4. Configure the environment variable JAVA_HOME:

   a. Find Java path
# ls -l /etc/alternatives/java

**Note:** The output of the above command points to /usr/java/default/jre/bin/java.

b. Set JAVA_HOME:
   ```
   # echo "export JAVA_HOME=/usr/java/default/jre" >> /root/.bashrc
   # source /root/.bashrc
   ```
   **Note:** /jre/java should not be included in the path for JAVA_HOME.

c. Repeat above for the "stack" user.
   ```
   # echo "export JAVA_HOME=/usr/java/default/jre" >> /home/stack/.bashrc
   # source /home/stack/.bashrc
   ```

5. If your infrastructure requires a proxy server to access the Internet, follow the maven-specific instructions in Appendix B: Configuring the Proxy.

### 6.4.2 Prepare for DevStack

1. Update the existing local.conf file in order for ODL to be functional with DevStack. The example below assumes the following:
   a. The controller management IP address is 10.11.13.8.
   b. port p786p1 is used for data plane network.

2. On the controller, modify local.conf:
   **Comment out these lines**
   ```
   #enable_service q-agt
   #Q_AGENT=openvswitch
   #Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch
   #OVS_PHYSICAL_BRIDGE=br-e786p1
   ```

   **Add these lines in the middle of file, anywhere before [[post-config|$NOVA_CONF]]
   ```
   enable_plugin networking-odl http://git.openstack.org/stackforge/networking-odl 2015.1.1
   ODL_MODE=allinone
   Q_HOST=$HOST_IP
   ODL_MGR_IP=10.11.13.8
   ODL_PROVIDER_MAPPINGS=physnet1:p786p1
   Q_PLUGIN=ml2
   Q_ML2_PLUGIN_MECHANISM_DRIVERS.opendaylight
   ```

3. On compute node, modify local.conf:
   **Comment out these lines**
   ```
   #enable_service q-agt
   #Q_AGENT=openvswitch
   #Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch
   #OVS_PHYSICAL_BRIDGE=br-e786p1
   ```

   **Add these lines in the middle of the file, anywhere before [[post-config|$NOVA_CONF]]
   ```
   enable_plugin networking-odl http://git.openstack.org/stackforge/networking-odl 2015.1.1
   ```
ODL_MODE=compute
ODL_MGR_IP=10.11.13.8
ODL_PROVIDER_MAPPINGS=physnet1:p786p1
Q_PLUGIN=ml2
Q_ML2_PLUGIN_MECHANISM_DRIVERS=opendaylight
Q_HOST=$SERVICE_HOST

4. Run stack.sh to bring up Openstack. Before running stacking, read the following test notes.

Test note A: On the ODL controller node, a bug was found in /opt/stack/networking-odl/devstack/plugin.py that tries to do a yum install of java-1.7 that is unavailable in the Fedora 21 repository. (For this reason, Java 1.7 is manually installed as the first step in this solutions guide.) With this bug, stacking will fail, and the error points to the installation of Java. To work around this, you can choose one of the following two methods:

a. After the stacking failure, run the following:

   # cd /home/stack/devstack
   # ./unstack.sh
   # sed -i 's/yum_install maven java-1.7.0-openjdk/yum_install maven/g'
   /opt/stack/networking-odl/devstack/plugin.sh
   # FORCE=yes ./stack.sh

b. To avoid the stacking failure, run the following commands before stacking:

   $ mkdir /opt/stack
   $ chown -R stack:stack /opt/stack
   $ git clone http://git.openstack.org/stackforge/networking-odl
   /opt/stack/networking-odl
   $ chown -R stack:stack /opt/stack/networking-odl
   $ cd /opt/stack/networking-odl
   $ git checkout 2015.1.1
   $ sed -i 's/yum_install maven java-1.7.0-openjdk/yum_install maven/g'
   /opt/stack/networking-odl/devstack/plugin.sh

Sample local.conf files for both controller and compute nodes are listed in Appendix A1, Sample Local.conf Files for OpenDaylight Configurations.

6.4.3 Additional Configurations and Operations

After stacking is done, perform the following steps to have the port added to the bridge br-int:

1. With the OpenDaylight configuration in OpenStack, one of the bridges, br-<data port> (e.g., br-enp786f0), is no longer used. Instead, OpenDaylight directly controls the bridge br-int for data flow. In this case, the physical port of the tenant network should be manually added to the bridge br-int in all nodes. The example below adds port p786p1 to br-int:

   # ovs-vsctl add-port br-int p786p1

2. You can use the same methods to create a VM as those described in section 6.1, Preparation with OpenStack.
6.4.4 Monitor Network Flow with OpenDaylight

DLUX is a web-based graphic interface for OpenDaylight. After successfully installing as described above, however, users still cannot access the DLUX. This is because some features are not installed by default.

1. To install these features, do the following:

   $ cd /opt/stack/opendaylight/distribution-karaf-0.2.3-Helium-SR3/
   $./bin/client -u karaf feature:install odl-base-all odl-aaa-authn odl-restconf
   odl-nsf-all odl-adsal-northbound odl-mdsal-apidocs odl-ovsdb-openstack odl-
   ovsdb-northbound odl-dlux-core odl-dlux-all

2. Access DLUX through a web browser. The following is an example of an OpenDaylight login page from the controller running at IP address 10.11.13.17:

3. Log in. (The default login credential is admin/admin.)
4. After login, the user can browse to “Nodes” for available openflow.

5. For more details on each flow, click **Node Connectors**. The screenshot below shows the connector at a compute node with two VM ports (name starting with tap) and one physical port (enp9s0f0).
6. Click **Flows** to show the Rx and Tx statistics of each connector.
7.0 Use Cases with Virtual Network Functions

This section describes the Virtual Network Functions (VNFs) that have been used in Open Network Platform for servers. These functions assume virtual machines (VMs) that have been prepared in a way similar to compute nodes. Refer to section 6.1, Preparation with OpenStack, for the test setup.

7.1 Generic VNF Configurations

Any VNF like a virtual firewall or virtual router can be implemented on either the same compute node like other VMs or remote compute node. The following two sections provide examples of how traffic can be routed for these two different scenarios.

7.1.1 Local VNF

The following figure describes a simple network flow between two VMs (a source and sink) with a VNF between them. All three are locally configured on the same compute node.

![Figure 7-1. Local VNF](image-url)
Configuration
1. OpenStack brings up the VMs and connects them to the vSwitch.
2. IP addresses of the VMs get configured using the DHCP server. VM1 belongs to one subnet and VM3 to a different one. VM2 has ports on both subnets.
3. Flows get programmed to the vSwitch by the OpenDaylight controller (refer to section 6.4, Using OpenDaylight).

Data Path (Numbers Matching Red Circles)
1. VM1 sends a flow to VM3 through the vSwitch.
2. The vSwitch forwards the flow to the first vPort of VM2 (active IPS).
3. The IPS receives the flow, inspects it and (if not malicious) sends it out through its second vPort.
4. The VNF receives the flow and sends it out through its second vPort.

7.1.2 Remote VNF

The following figure shows a simple network flow between two VMs (source and sink) with the VNF configured on a separate compute node.
Configuration

1. OpenStack brings up the VMs and connects them to the vSwitch.
2. The IP addresses of the VMs get configured using the DHCP server.

Data Path (Numbers Matching Red Circles)

1. VM1 sends a flow to VM3 through the vSwitch inside compute node 1.
2. The vSwitch forwards the flow out of the first port to the first port of compute node 2.
3. The vSwitch of compute node 2 forwards the flow to the first port of the vHost, where the traffic gets consumed by VM1.
4. The VNF receives the flow and sends it out through its second port of the vHost into the vSwitch of compute node 2.
5. The vSwitch forwards the flow out of the second XL710 port of compute node 2 into the second port of the XL710 in compute node 1.
6. The vSwitch of compute node 1 forwards the flow into the port of the vHost of VM3 where the flow gets terminated.

7.1.3 Network Configuration with Source and Sink VM

Sink and source are two Fedora VMs that are only used to generate traffic:

1. Install iperf:
   
   # yum install -y iperf

2. Turn on IP forwarding:

   # sysctl -w net.ipv4.ip_forward=1

3. In the source, add the route to the sink:

   # route add -net 11.0.0.0/24 eth0

4. At the sink, add the route to the source:

   # route add -net 10.0.0.0/24 eth0
7.2 Installation and Configuration of vIPS

7.2.1 Setup

Controller and compute nodes have been set up with the following SRT ingredients:

1. DPDK (only for compute node)
2. Open vSwitch
3. OpenStack Kilo Release

As a prerequisite, it is assumed that the minimum components of OpenStack are installed and properly configured as described in section 6.1.

7.2.2 Local vIPS Test

1. From the OpenStack UI (<controller ip address>/project/instances), the "demo" user navigates as follows: Compute > Project > Instances > Access VMs: vm01, vm03 & vm02. It is assumed that vm01 is on network 1, vm03 is on network 2, and vm02 is on both networks; vm01 will be the source, and vm03 the sink in this test.

2. From each console, log in as root: Check that vm has loopback and the ip address is assigned by DHCP:

   $ifconfig
   eth0: inet <ip address>
   lo: inet <127.0.0.1>

3. From vm02:
   a. Test ping the other vms:
      $ping <vm01-ip>
      $ping <vm03-ip>
   b. Turn on IP forwarding, or run suricata.sh:
      
      $./suricata-conf.sh
      -or-
      $sysctl net.ipv4.ip_forward=1
      $iptables -I FORWARD -i eth0 -o eth1 -j NFQUEUE
      $iptables -I FORWARD -i eth1 -o eth0 -j NFQUEUE
      $echo 1 > /proc/sys/net/ipv4/conf/eth0/proxy_arp
      $echo 1 > /proc/sys/net/ipv4/conf/eth1/proxy_arp
      $suricata -c /etc/suricata/suricata.yaml -q 0

4. From vm01:
   a. Test ping vm02:
      $ping <vm02-net1-ip>
   b. Turn on IP forwarding or run add_router.sh:
      
      $./add_router.sh <11.0.0.0>
      -or-
$sysctl -w net.ipv4.ip_forward=1
$route add -net <11.0.0.0/24> eth0

c. Test ping vm02 and vm03:
   $ping <vm02-net2-ip>
   $ping <vm03-ip>

5. From vm03:
   a. Test ping vm02:
      $ping <vm02-net2-ip>
   b. Turn on IP forwarding or run add_router.sh:
      $./add_router.sh <10.0.0.0>
      -or-
      $sysctl -w net.ipv4.ip_forward=1
      $route add -net <10.0.0.0/24> eth0
   c. Test ping vm02 and vm03:
      $ping <vm02-net1-ip>
      $ping <vm01-ip>

6. Ensure the test pings from step 3 to step 5 are successful. Then proceed as follows:
   a. From vm03:
      $cd iperf
      $./iperf -s
   b. From vm01:
      $cd iperf
      $./iperf -c <vm03-ip> -l 1000
      $./iperf -c <vm03-ip> -l 1000 -t 60
      $./iperf -c <vm03-ip> -l 64 -t 60
      $./iperf -c <vm03-ip> -l 1500 -t 60
Figure 7–3. Local vIPS

Configuration:
1. OpenStack brings up the VMs and connects them to the vSwitch.
2. IP addresses of the VMs get configured using the DHCP server. VM1 belongs to one subnet and VM3 to a different one. VM2 has ports on both subnets.

Data Path (Numbers Matching Red Circles):
1. VM1 sends a flow to VM3 through the vSwitch.
2. The vSwitch forwards the flow to the first vPort of VM2 (active IPS).
3. The IPS receives the flow, inspects it and (if not malicious) sends it out through its second vPort.
4. The vSwitch forwards it to VM3.

7.2.3 Remote vIPS Test

1. After completing local vIPS test, delete vm02 from the previous setup:
   a. From the control node:
      i. source demo-cred
      ii. nova delete <vm02>
2. Return to step 8 of section 6.1.1.2:
   a. Create vm02 in a different availability zone (and, therefore, different aggregate and compute node) from the one vm01 and vm03 are on.
   b. Proceed with all steps as in the local vIPS test (refer to section 7.2.2, Local vIPS Test).
1. OpenStack brings up the VMs and connects them to the vSwitch.
2. The IP addresses of the VMs get configured using the DHCP server.

**Data Path (Numbers Matching Red Circles)**

1. VM1 sends a flow to VM3 through the vSwitch inside compute node 1.
2. The vSwitch forwards the flow out of the first 82599 port (used for DPDK and OVS) to the first data port of compute node 2.
3. The vSwitch of compute node 2 forwards the flow to the first port of the vHost, where the traffic gets consumed by VM2.
4. The IPS receives the flow, inspects it, and (provided it is not malicious) sends it out through its second port of the vHost into the vSwitch of compute node 2.
5. The vSwitch forwards the flow out of the second data port of compute node 2 into the second data port in compute node 1.
6. The vSwitch of compute node 1 forwards the flow into the port of the vHost of VM3 where the flow gets terminated.
7.3 Installation and Configuration of the vBNG

A virtual Border Network Gateway (vBNG) application can be configured and installed using the set of commands below. It is assumed that you have successfully set up a controller node and a compute node as described in section 6.1, Preparation with OpenStack.

1. Execute the following command in a Fedora VM with two Virtio interfaces:
   ```bash
   # yum -y update
   ``

2. Disable SELINUX:
   ```bash
   # setenforce 0
   # vi /etc/selinux/config
   
   And change so SELINUX=disabled
   ``

3. Disable the firewall:
   ```bash
   # systemctl disable firewalld.service
   
   # reboot
   ``

4. Edit the grub default configuration:
   ```bash
   # vi /etc/default/grub
   
   Add hugepages to it:
   ```bash
   
   ... noirqbalance intel_idle.max_cstate=0 processor.max_cstate=0 ipv6.disable=1
   default_hugepagesz=1G hugepages=4 2
   isolcpus=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19"
   ```

5. Rebuild grub config and reboot the system:
   ```bash
   # grub2-mkconfig -o /boot/grub2/grub.cfg
   
   # reboot
   ``

6. Verify that hugepages are available in the VM:
   ```bash
   # cat /proc/meminfo
   
   HugePages_Total:4
   HugePages_Free:4
   Hugepagesize:1048576 kB
   ...
   ``

7. Add the following to the end of ~/.bashrc file:
   ```bash
   # ----------------------------------------------
   export RTE_SDK=/root/dpdk
   export RTE_TARGET=x86_64-native-linuxapp-gcc
   export OVS_DIR=/root/ovs
   export RTE_UNBIND=$RTE_SDK/tools/dpdk_nic_bind.py
   export DPDK_DIR=$RTE_SDK;
   export DPDK_BUILD=$DPDK_DIR/$RTE_TARGET
   # ----------------------------------------------
   
   # Re-log in or source that file:
   . .bashrc
9. Install DPDK:
   cd /root
   git clone http://dpdk.org/git/dpdk
cd dpdk
git checkout v1.8.0
make install T=$RTE_TARGET
modprobe uio
insmod $RTE_SDK/$RTE_TARGET/kmod/igb_uio.ko

10. Check the PCI addresses of the vVirtio vNICs:
    lspci | grep Ethernet
    00:03.0 Ethernet controller: Red Hat, Inc Virtio network device
    00:04.0 Ethernet controller: Red Hat, Inc Virtio network device

11. Use the DPDK binding scripts to bind the interfaces to DPDK instead of the kernel:
    # $RTE_SDK/tools/dpdk_nic_bind.py
    # $RTE_SDK/tools/dpdk_nic_bind.py -b igb_uio 00:03.0
    # $RTE_SDK/tools/dpdk_nic_bind.py -b igb_uio 00:04.0

12. Download the BNG packages:

13. Extract the DPPD BNG sources:
    # unzip dppd-bng-v013.zip

14. Build the BNG DPPD application:
    # yum -y install ncurses-devel
    # cd dppd-BNG-v013
    # make

    The application starts like this:
    ./build/dppd -f config/handle_none.cfg
When run under OpenStack, it should look like this.
A.1 Sample Local.conf Files for OpenDaylight Configurations

The following is a sample local.conf for the OpenDaylight controller node:

```
# Controller node
#
[[local|localrc]]
FORCE=yes

HOST_NAME=$\{hostname\}
HOST_IP=10.11.12.13
HOST_IP_IFACE=enp3s0f0

PUBLIC_INTERFACE=eth2
VLAN_INTERFACE=ens786f0
FLAT_INTERFACE=ens786f0

ADMIN_PASSWORD=password
MYSQL_PASSWORD=password
DATABASE_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=no
-HORIZON_PASSWORD=password
RABBIT_PASSWORD=password

disable_service n-net
disable_service n-cpu

enable_service q-svc
#enable_service q-agt
enable_service q-dhcp
enable_service q-l3
enable_service q-meta
enable_service neutron
enable_service n-novnc
enable_service n-xvnc
enable_service n-crt
enable_service n-cauth

LOGFILE=$\{DEST\}/stack.sh.log
SCREEN_LOGDIR=$\{DEST\}/screen
SYSLOG=True
LOGDAYS=1

#Q_AGENT=openvswitch
#Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch
Q_ML2_PLUGIN_TYPE_DRIVERS=vlan,flat,local
Q_ML2_TENANT_NETWORK_TYPE=vlan
```
ENABLE_TENANT_TUNNELS=False
ENABLE_TENANT_VLANS=True

PHYSICAL_NETWORK=physnet1
ML2_VLAN_RANGES=physnet1:1000:1010
#OVS_PHYSICAL_BRIDGE=br-ens786f0

MULTI_HOST=True

# ODL specific
enable_plugin networking-odl http://git.opendaylight.org/stackforge/networking-odl
2015.1.1
ODL_MODE=allinone
Q_HOST=$HOST_IP
ODL_MGR_IP=10.11.12.13
ODL_PROVIDER_MAPPINGS=physnet1:ens786f0
Q_PLUGIN=ml2
Q_ML2_PLUGIN_MECHANISM_DRIVERS=opendaylight

[[post-config|$NOVA_CONF]]
[DEFAULT]
firewall_driver=no
token=va.virt.firewall.NoopFirewallDriver
novncproxy_host=0.0.0.0
novncproxy_port=6080

Here is a sample local.conf for compute node:

# Compute node
# OVS_TYPE=ovs
[[local|localrc]]

FORCE=yes
MULTI_HOST=True

HOST_NAME=$(hostname)
HOST_IP=10.11.12.14
HOST_IP_IFACE=enp3s0f0
SERVICE_HOST_NAME=10.11.12.13
SERVICE_HOST=10.11.12.13

MYSQL_HOST=$SERVICE_HOST
RABBIT_HOST=$SERVICE_HOST

GLANCE_HOST=$SERVICE_HOST
GLANCE_HOSTPORT=$SERVICE_HOST:9292
KEYSTONE_AUTH_HOST=$SERVICE_HOST
KEYSTONE_SERVICE_HOST=10.11.12.13

ADMIN_PASSWORD=password
MYSQL_PASSWORD=password
DATABASE_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=no-token-password
HORIZON_PASSWORD=password
RABBIT_PASSWORD=password

disable_all_services
enable_service n-cpu
#enable_service q-agt
DEST=/opt/stack
LOGFILE=$DEST/stack.sh.log
SCREEN_LOGDIR=$DEST/screen
SYSLOG=True
LOGDAYS=1

#OVS_AGENT_TYPE=openvswitch
Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch
Q_ML2_PLUGIN_TYPE_DRIVERS=vlan

ENABLE_TENANT_TUNNELS=False
ENABLE_TENANT_VLANS=True

Q_ML2_TENANT_NETWORK_TYPE=vlan
ML2_VLAN_RANGES=physnet1:1000:1010
PHYSICAL_NETWORK=physnet1
#OVS_PHYSICAL_BRIDGE=br-ens786f0

# ODL Specific
enable_plugin networking-odl http://git.openstack.org/stackforge/networking-odl
2015.1.1
ODL_MODE=compute

Q_HOST=$SERVICE_HOST
ODL_MGR_IP=10.11.12.13
ODL_PROVIDER_MAPPINGS=physnet1:ens786f0
Q_PLUGIN=ml2
Q_ML2_PLUGIN_MECHANISM_DRIVERS=opendaylight

[[post-config|$NOVA_CONF]]
[DEFAULT]
firewall_driver=nova.virt.firewall.NoopFirewallDriver
vnc_enabled=True
vncserver_listen=0.0.0.0
vncserver_proxyclient_address=10.11.12.14

[libvirt]
cpu_mode=host-model
A.2 Sample Local.conf Files for SR-IOV Configurations

The following is a sample local.conf with SR-IOV configuration for controller node:

```bash
# Controller node
#
[[local|localrc]]
FORCE=yes

HOST_NAME=$(hostname)
HOST_IP=10.11.13.31
HOST_IP_IFACE=enp3s0f0

PUBLIC_INTERFACE=ens786f1
VLAN_INTERFACE=ens786f0
FLAT_INTERFACE=ens786f0

ADMIN_PASSWORD=password
MYSQL_PASSWORD=password
DATABASE_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=no-token-password
HORIZON_PASSWORD=password
RABBIT_PASSWORD=password

disable_service n-net
disable_service n-cpu

enable_service q-svc
enable_service q-agt
enable_service q-dhcp
enable_service q-l3
enable_service neutron
enable_service n-novnc
enable_service n-xvnc
enable_service n-crt
enable_service n-cauth

LOGFILE=$DEST/stack.sh.log
SCREEN_LOGDIR=$DEST/screen
SYSLOG=True
LOGDAYS=1

Q_AGENT=openvswitch
Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch,SR-IOVnicswitch
Q_ML2_PLUGIN_TYPE_DRIVERS=vlan,flat,local
Q_ML2_TENANT_NETWORK_TYPE=vlan

ENABLE_TENANT_TUNNELS=False
ENABLE_TENANT_VLANS=True

PHYSICAL_NETWORK=physnet1
ML2_VLAN_RANGES=physnet1:1000:1010
OVS_PHYSICAL_BRIDGE=br-ens786f0
```
MULTI_HOST=True

[[post-config|$Q_PLUGIN_CONF_FILE]]
[ml2_sriov]
supported_pci_vendor_devs=8086:1572,8086:154c
agent_required=True

[sriov_nic]
physical_device_mappings=physnet1:ens786f1

[[post-config|$NOVA_CONF]]
[DEFAULT]
fwpwall_driver=nova.virt.firewall.NoopFirewallDriver
novncproxy_host=0.0.0.0
novncproxy_port=6080

scheduler_default_filters=RamFilter,ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter
cpci_alias={"name":"fortville","product_id":"154c","vendor_id":"8086"}

Here is a sample local.conf with the sriov configuration for compute node:

```plaintext
# Compute node
# OVS_TYPE=ovs
[[local|localrc]]

FORCE=yes
MULTI_HOST=True

HOST_NAME=$(hostname)
HOST_IP=10.11.13.36
HOST_IP_IFACE=enp3s0f0
SERVICE_HOST_NAME=10.11.13.31
SERVICE_HOST=10.11.13.31

MYSQL_HOST=$SERVICE_HOST
RABBIT_HOST=$SERVICE_HOST
GLANCE_HOST=$SERVICE_HOST
GLANCE_HOSTPORT=$SERVICE_HOST:9292
KEYSTONE_AUTH_HOST=$SERVICE_HOST
KEYSTONE_SERVICE_HOST=10.11.13.31

ADMIN_PASSWORD=password
MYSQL_PASSWORD=password
DATABASE_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=no-token-password
HORIZON_PASSWORD=password
RABBIT_PASSWORD=password

disable_all_services
enable_service n-cpu
enable_service q-agt

DEST=/opt/stack
```
LOGFILE=$DEST/stack.sh.log
SCREEN_LOGDIR=$DEST/screen
SYSLOG=True
LOGDAYS=1

OVS_AGENT_TYPE=openvswitch
Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch
Q_ML2_PLUGIN_TYPE_DRIVERS=vlan

ENABLE_TENANT_TUNNELS=False
ENABLE_TENANT_VLANS=True

Q_ML2_TENANT_NETWORK_TYPE=vlan
ML2_VLAN_RANGES=physnet1:1000:1010
PHYSICAL_NETWORK=physnet1
OVS_PHYSICAL_BRIDGE=br-ens786f0

[[post-config|$NOVA_CONF]]
[DEFAULT]
firewall_driver=nova.virt.firewall.NoopFirewallDriver
vnc_enabled=True
vncserver_listen=0.0.0.0
vncserver_proxyclient_address=10.11.13.36

pci_passthrough_whitelist=

[libvirt]
cpu_mode=host-model
Appendix B  Configuring the Proxy

This appendix describes how to configure the proxy in case the infrastructure requires it. Proxy settings are generally set as environment variables in the user’s .bashrc:

$ vi ~/.bashrc

And add:

export http_proxy=<your http proxy server>:<your http proxy port>
export https_proxy=<your https proxy server>:<your https proxy port>

Also add the no proxy settings, i.e., the hosts and/or subnets that you do not want to use the proxy server to access:

export no_proxy=192.168.122.1,<intranet subnets>

If you want to make the change across all users instead of just yours, make the above additions with /etc/profile as root:

# vi /etc/profile

This will allow most shell commands (e.g., wget or curl) to access your proxy server first.

Because yum does not read the proxy settings from your shell, you must also edit /etc/yum.conf as root and add the following line:

# vi /etc/yum.conf
proxy=http://<your http proxy server>:<your http proxy port>

In order for git to also use your proxy servers, execute the following command:

$ git config --global http.proxy <your http proxy server>:<your http proxy port>
$ git config --global https.proxy <your https proxy server>:<your https proxy port>

If you want to make the git proxy settings available to all users as root, run the following commands instead:

# git config --system http.proxy <your http proxy server>:<your http proxy port>
# git config --system https.proxy <your https proxy server>:<your https proxy port>

For OpenDaylight deployments, the proxy needs to be defined as part of the XML settings file of Maven:

1. Create settings.xml to .m2/ directory, if it does not already exist:
   $ mkdir ~/.m2

2. Edit the ~/.m2/settings.xml file:
   $ vi ~/.m2/settings.xml

3. Add the following:

   <settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
   <localRepository/>
   <interactiveMode/>
   <usePluginRegistry/>
   <offline/>
   <pluginGroups/>
<servers/>
<mirrors/>
<proxies>
  <proxy>
    <id>intel</id>
    <active>true</active>
    <protocol>http</protocol>
    <host><strong>your http proxy host</strong></host>
    <port><strong>your http proxy port no</strong></port>
    <nonProxyHosts>localhost,127.0.0.1</nonProxyHosts>
  </proxy>
</proxies>
<profiles/>
<activeProfiles/>
</settings>
Appendix C  Configuring Horizon UI to Deploy Virtual Machines

C.1  Custom VM Image and Zoning

Users can create their own custom VM image to deploy. Horizon UI provides a few choices to accomplish this. These include the choice of flavor, format, architecture, etc. The following example provides instructions on how to use a customer VM image and create a host aggregate and an available zone.

1. Use admin user to log in from the OpenStack dashboard:

   http://10.11.12.1/

2. After logging in, create a VM image. Click the Images tab under the System Panel in the left pane, and then click the Create Image tab in the upper-right corner.
3. In the “Create An Image” window, enter the image name and description, then select the image source (the source should be accessible by OpenStack) and form from the respective drop-down boxes. Click **Create Image** at the bottom right to load the image file to the controller host.

4. Create the available zone and host aggregate. Click **Host Aggregates** under **System Panel** on the left pane, and then click **Create Host Aggregate** in the upper-right corner.

5. In the **Create Host Aggregate** window, enter the names of the aggregate and availability zone.
6. Click the **Hosts within aggregate** tab (all available hosts are listed), and then select host(s) to add into the aggregate.

7. Click **Create Host Aggregate** to finish.
C.2 Creating Additional Networks

The following example describes how to create a new network in an OpenStack environment and apply it to a VM to enable the VM to have multiple network interfaces. To do this, follow these steps:

1. Use demo user to log in.
2. Click the Network/Networks tab in the left pane.
3. Click Create Network in the upper-right corner.
   Note: Make sure the user is “demo” under the projects section on top left of the page, as highlighted in red below.

4. In the Create Network window, click the Network tab, then enter the network name. Click the Subnet tab and enter a subnet name, network address range, and gateway. Click Next to continue. Note that users can ignore DNS and the router setup, and complete creating the network.
5. Click **Finish** to complete the creation of the network.
C.3 VM Deployment

The following example describes how to use a customer VM image and aggregate to launch a VM in an OpenStack environment.

1. Log in as demo user.
2. Click the Instances tab under Project > Compute in the left pane. Click the Launch Instance tab at the upper-right corner in the new window, select the desired availability zone, instance name, flavor, and instance boot source from the respective drop-down boxes.

3. Click the Networking tab, and then select one or more networks for the instance. The two networks, private and the newly created new-net, are now available. Assign both networks when creating the instance.

4. Click Launch to finish. The instance has two network interfaces, belonging to two different networks.
Note: Make sure the user is "demo" under the projects section on top left of the page, as highlighted in red below.

5. The new VM should be up and running in a few minutes. Click the new name of the VM from the list, then click **Console** in the top menu to access the VM.
## Appendix D  Glossary

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATR</td>
<td>Application Targeted Routing</td>
</tr>
<tr>
<td>BNG</td>
<td>Broadband (or Border) Network Gateway</td>
</tr>
<tr>
<td>BRAS</td>
<td>Broadband Remote Access Server</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>DPDK</td>
<td>Data Plane Development Kit</td>
</tr>
<tr>
<td>IOMMU</td>
<td>Input/Output Memory Management Unit</td>
</tr>
<tr>
<td>KVM</td>
<td>Kernel-based Virtual Machine</td>
</tr>
<tr>
<td>NFV</td>
<td>Network Function Virtualization</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card</td>
</tr>
<tr>
<td>NTP</td>
<td>Network Time Protocol</td>
</tr>
<tr>
<td>NUMA</td>
<td>Non-Uniform Memory Access</td>
</tr>
<tr>
<td>ONP</td>
<td>Open Network Platform</td>
</tr>
<tr>
<td>OVS</td>
<td>Open vSwitch</td>
</tr>
<tr>
<td>SDN</td>
<td>Software Defined Network</td>
</tr>
<tr>
<td>SR-IOV</td>
<td>Single Root I/O Virtualization</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>VNF</td>
<td>Virtualized Network Function</td>
</tr>
</tbody>
</table>
## Appendix E  References

<table>
<thead>
<tr>
<th>Document Name</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Protocol version 4</td>
<td><a href="http://www.ietf.org/rfc/rfc791.txt">http://www.ietf.org/rfc/rfc791.txt</a></td>
</tr>
<tr>
<td>Internet Protocol version 6</td>
<td><a href="http://www.faqs.org/rfc/rfc2460.txt">http://www.faqs.org/rfc/rfc2460.txt</a></td>
</tr>
<tr>
<td>4x Intel® 10 Gigabit Fortville (FVL) XL710 Ethernet Controller</td>
<td><a href="http://ark.intel.com/products/82945/Intel-Ethernet-Controller-XL710-AM1">http://ark.intel.com/products/82945/Intel-Ethernet-Controller-XL710-AM1</a></td>
</tr>
<tr>
<td>DPDK</td>
<td><a href="http://www.intel.com/go/dpdk">http://www.intel.com/go/dpdk</a></td>
</tr>
<tr>
<td>OpenvSwitch with DPDK-netdev</td>
<td><a href="https://01.org/packet-processing">https://01.org/packet-processing</a></td>
</tr>
</tbody>
</table>
Legal Information

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below. 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.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

The products described in this document 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. Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Intel technologies may require enabled hardware, specific software, or services activation. Check with your system manufacturer or retailer. Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice. Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

No computer system can be absolutely secure. Intel does not assume any liability for lost or stolen data or systems or any damages resulting from such losses.

Intel does not control or audit third-party websites referenced in this document. You should visit the referenced website and confirm whether referenced data are accurate.

Intel Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

2015 Intel Corporation. All rights reserved. Intel, the Intel logo, Core, Xeon, and others 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.