Setting Up a Service VM as an IPv6 vRouter

Release draft (7c6658f)

OPNFV

August 22, 2016

Project IPv6, http://wiki.opnfv.org/ipv6_opnfv_project

Editors Bin Hu (AT&T), Sridhar Gaddam (RedHat)

Authors Sridhar Gaddam (RedHat), Bin Hu (AT&T)

Abstract

This document provides the users with installation guidelines to create a Service VM as an IPv6 vRouter in OPNFV environment, i.e. integrated OpenStack with Open Daylight environment. There are three scenarios.

- Scenario 1 is pre-OPNFV environment, i.e. a native OpenStack environment without Open Daylight Controller.
- Scenario 2 is an OPNFV environment where OpenStack is integrated with Open Daylight Official Lithium Release. In this setup we use ODL for "Layer 2 connectivity" and Neutron L3 agent for "Layer 3 routing". Because of a bug, which got fixed recently and is not part of ODL SR3, we will have to manually execute certain commands to simulate an external IPv6 Router in this setup.
- Scenario 3 is similar to Scenario 2. However, we use an Open Daylight Lithium controller which is built from the latest stable/Lithium branch which includes the fix. In this scenario, we can fully automate the setup similar to Scenario 1.

Please **NOTE** that the instructions in this document assume the deployment model of single controller node. In case of HA (High Availability) deployment model where multiple controller nodes are used, the setup procedure is the same. In particular:

- There is **No Impact** on Scenario 1 and Scenario 3.
- For Scenario 2, when `ipv6-router` is created in step **SETUP-SVM-11**, it could be created in any of the controller node. Thus you need to identify in which controller node `ipv6-router` is created in order to manually spawn `radvd` daemon inside the `ipv6-router` namespace in steps **SETUP-SVM-24** through **SETUP-SVM-30**.

CHAPTER ONE

ARCHITECTURAL DESIGN

The architectural design of using a service VM as an IPv6 vRouter is shown as follows in Fig. 1.1:

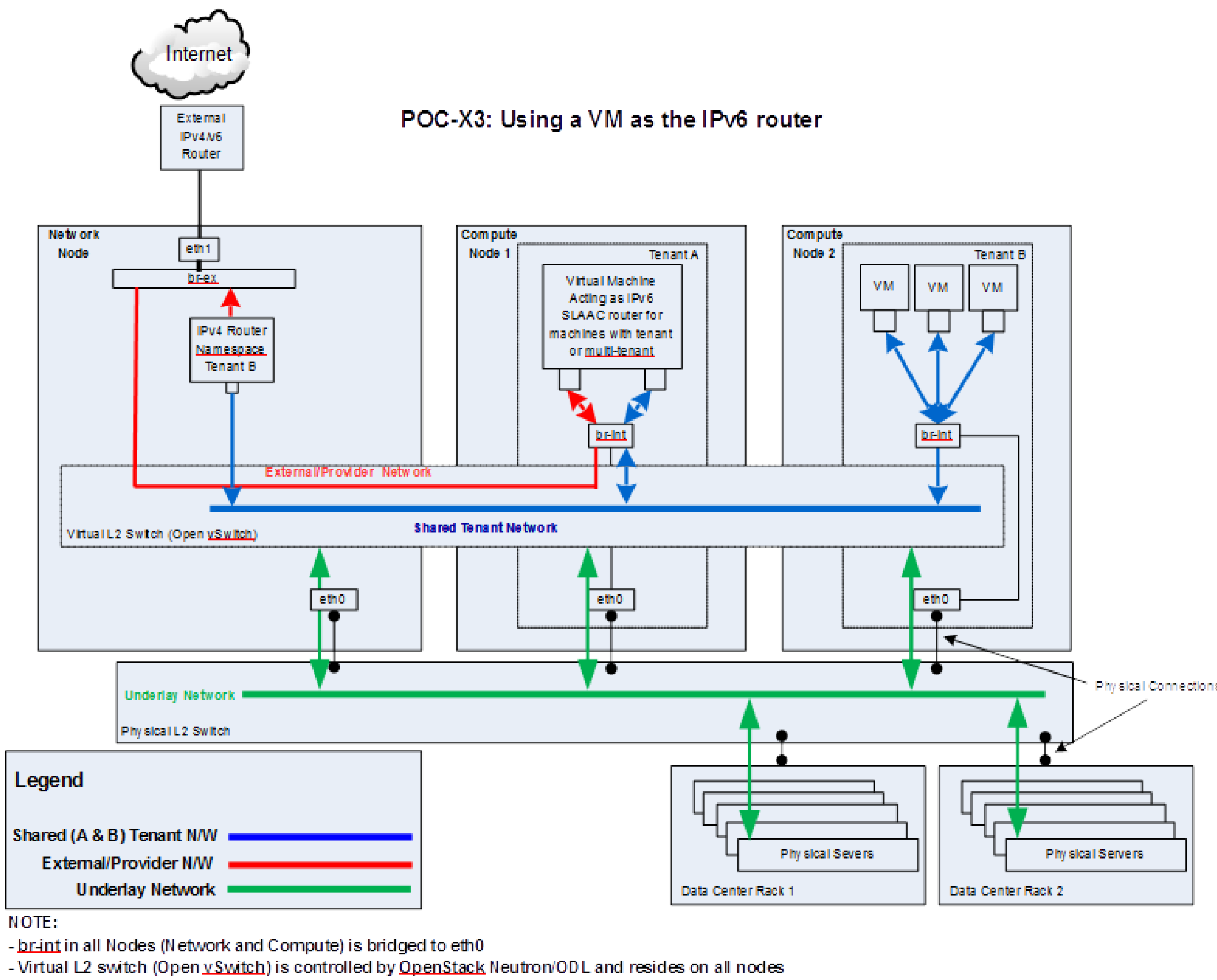

Fig. 1.1: Architectural Design of Using a VM as an IPv6 vRouter

This design applies to deployment model of single controller node as well as HA (High Availability) deployment model of multiple controller nodes.

CHAPTER TWO

SCENARIO 1 - NATIVE OPENSTACK ENVIRONMENT

Scenario 1 is the native OpenStack environment. Although the instructions are based on Liberty, they can be applied to Kilo in the same way. Because the anti-spoofing rules of Security Group feature in OpenStack prevents a VM from forwarding packets, we need to disable Security Group feature in the native OpenStack environment.

For exemplary purpose, we assume:

- A two-node setup of OpenStack environment is used as shown in Fig. 2.1
- The hostname of OpenStack Controller+Network+Compute Node is `opnfv-os-controller`, and the host IP address is `192.168.0.10`
- The hostname of OpenStack Compute Node is `opnfv-os-compute`, and the host IP address is `192.168.0.20`
- Ubuntu 14.04 or Fedora 21 is installed
- We use `opnfv` as username to login.
- We use `devstack` to install OpenStack Liberty. Please note that OpenStack Kilo can be used as well.

Please NOTE that:

- **The IP address shown in Fig. 2.1 are for exemplary purpose. You need to configure your public IP address connecting to Internet according to your actual network infrastructure. And you need to make sure the private IP address are not conflicting with other subnets**.
- **Although the deployment model of single controller node is assumed, in case of HA (High Availability) deployment model where multiple controller nodes are used, there is no impact and the setup procedure is the same.**

2.1 Prerequisite

OS-NATIVE-0: Clone the following GitHub repository to get the configuration and metadata files

```
git clone https://github.com/sridhargaddam/opnfv_os_ipv6_poc.git /opt/stack/opnfv_os_ipv6_poc
```

2.2 Set up OpenStack Controller Node

We assume the hostname is `opnfv-os-controller`, and the host IP address is `192.168.0.10`.

OS-NATIVE-N-1: Clone `stable/liberty` `devstack` code base.

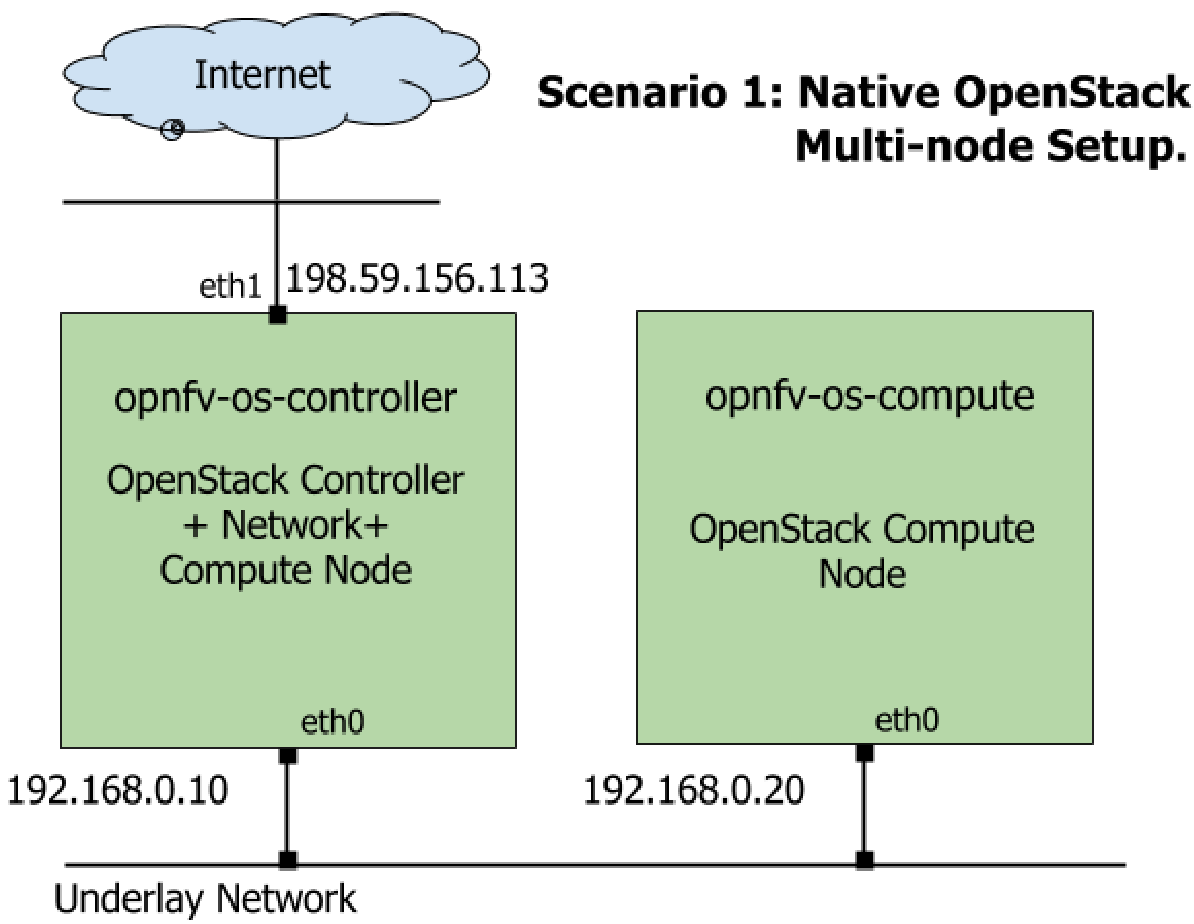

Fig. 2.1: Underlay Network Topology - Scenario 1

```
git clone https://github.com/openstack-dev/devstack.git -b stable/liberty
```

OS-NATIVE-N-2: Copy `local.conf.controller` to `devstack` as `local.conf`

```
cp /opt/stack/opnfv_os_ipv6_poc/local.conf.controller ~/devstack/local.conf
```

OS-NATIVE-N-3: If you want to modify any `devstack` configuration, update `local.conf` now.

OS-NATIVE-N-4: Start the `devstack` installation.

```
cd ~/devstack
./stack.sh
```

OS-NATIVE-N-5: If all goes well, you should see the following output.

```
This is your host IP address: 192.168.0.10
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.0.10/
Keystone is serving at http://192.168.0.10:5000/
The default users are: admin and demo
The password: password
```

2.3 Set up OpenStack Compute Node

We assume the hostname is `opnfv-os-compute`, and the host IP address is `192.168.0.20`.

OS-NATIVE-M-1: Clone `stable/liberty` `devstack` code base.

```
git clone https://github.com/openstack-dev/devstack.git -b stable/liberty
```

OS-NATIVE-M-2: Copy `local.conf.compute` to `devstack` as `local.conf`

```
cp /opt/stack/opnfv_os_ipv6_poc/local.conf.compute ~/devstack/local.conf
```

Please **note** that you need to change the IP address of `SERVICE_HOST` to point to your actual IP address of OpenStack Controller

OS-NATIVE-M-3: If you want to modify any `devstack` configuration, update `local.conf` now.

OS-NATIVE-M-4: Start the `devstack` installation.

```
cd ~/devstack
./stack.sh
```

OS-NATIVE-M-5: If all goes well, you should see the following output.

```
This is your host IP address: 192.168.0.20
This is your host IPv6 address: ::1
```

OS-NATIVE-M-6 (OPTIONAL): You can verify that OpenStack is set up correctly by showing hypervisor list

```
~/devstack$ nova hypervisor-list
+----+------------------------------------+---------+------------+
| ID | Hypervisor hostname                | State   | Status     |
+----+------------------------------------+---------+------------+
| 1  | opnfv-os-controller                | up      | enabled    |
| 2  | opnfv-os-compute                   | up      | enabled    |
+----+------------------------------------+---------+------------+
```

2.4 Note: Disable Security Groups in OpenStack ML2 Setup

Please note that Security Groups feature has been disabled automatically through `local.conf` configuration file during the setup procedure of OpenStack in both Controller Node and Compute Node.

If you are an experienced user and installing OpenStack using a different installer (i.e. not with `devstack`), please make sure that Security Groups are disabled in the setup. You can verify that your setup has the following configuration parameters.

OS-NATIVE-SEC-1: Change the settings in `/etc/neutron/plugins/ml2/ml2_conf.ini` as follows

```
# /etc/neutron/plugins/ml2/ml2_conf.ini
[securitygroup]
extension_drivers = port_security
enable_security_group = False
firewall_driver = neutron.agent.firewall.NoopFirewallDriver
```

OS-NATIVE-SEC-2: Change the settings in `/etc/nova/nova.conf` as follows

```
# /etc/nova/nova.conf
[DEFAULT]
security_group_api = nova
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

OS-NATIVE-SEC-3: After updating the settings, you will have to restart the `Neutron` and `Nova` services.

Please note that the commands of restarting `Neutron` **and** `Nova` **would vary depending on the installer. Please refer to relevant documentation of specific installers**

2.5 Set Up Service VM as IPv6 vRouter

OS-NATIVE-SETUP-1: Now we assume that OpenStack multi-node setup is up and running. We have to source the tenant credentials in this step. The following commands should be executed in `devstack`:

```
# source the tenant credentials in devstack
cd ~/devstack
source openrc admin demo
```

Please **NOTE** that the method of sourcing tenant credentials may vary depending on installers. **Please refer to relevant documentation of installers if you encounter any issue**.

OS-NATIVE-SETUP-2: Download `fedora22` image which would be used for `vRouter`

```
wget https://download.fedoraproject.org/pub/fedora/linux/releases/22/Cloud/x86_64/Images/Fedora-Clou
```

OS-NATIVE-SETUP-3: Import Fedora22 image to `glance`

```
glance image-create --name 'Fedora22' --disk-format qcow2 --container-format bare --file ./Fedora-Clo
```

OS-NATIVE-SETUP-4: Now we have to move the physical interface (i.e. the public network interface) to `br-ex`, including moving the public IP address and setting up default route. **Please note that this step may already have been done when you use a different installer to deploy OpenStack because that installer may have already moved the physical interface to** `br-ex` **during deployment**.

Because our `opnfv-os-controller` node has two interfaces `eth0` and `eth1`, and `eth1` is used for external connectivity, move the IP address of `eth1` to `br-ex`.

Please note that the IP address `198.59.156.113` and related subnet and gateway addressed in the command below are for exemplary purpose. **Please replace them with the IP addresses of your actual network**.

```
sudo ip addr del 198.59.156.113/24 dev eth1
sudo ovs-vsctl add-port br-ex eth1
sudo ifconfig eth1 up
sudo ip addr add 198.59.156.113/24 dev br-ex
sudo ifconfig br-ex up
sudo ip route add default via 198.59.156.1 dev br-ex
```

OS-NATIVE-SETUP-5: Verify that `br-ex` now has the original external IP address, and that the default route is on `br-ex`

```
opnfv@opnfv-os-controller:~/devstack$ ip a s br-ex
38: br-ex: <BROADCAST,UP,LOWER_UP> mtu 1430 qdisc noqueue state UNKNOWN group default
    link/ether 00:50:56:82:42:d1 brd ff:ff:ff:ff:ff:ff
    inet 198.59.156.113/24 brd 198.59.156.255 scope global br-ex
       valid_lft forever preferred_lft forever
    inet6 fe80::543e:28ff:fe70:4426/64 scope link
       valid_lft forever preferred_lft forever
opnfv@opnfv-os-controller:~/devstack$
opnfv@opnfv-os-controller:~/devstack$ ip route
default via 198.59.156.1 dev br-ex
192.168.0.0/24 dev eth0  proto kernel  scope link  src 192.168.0.10
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1
198.59.156.0/24 dev br-ex  proto kernel  scope link  src 198.59.156.113
```

Please note that the IP addresses above are exemplary purpose.

OS-NATIVE-SETUP-6: Create Neutron routers `ipv4-router` and `ipv6-router` which need to provide external connectivity.

```
neutron router-create ipv4-router
neutron router-create ipv6-router
```

OS-NATIVE-SETUP-7: Create an external network/subnet `ext-net` using the appropriate values based on the data-center physical network setup.

Please **NOTE** that if you use a different installer, i.e. NOT `devstack`, your installer may have already created an external network during installation. Under this circumstance, you may only need to create the subnet of `ext-net`. When you create the subnet, you must use the same name of external network that your installer creates.

Please refer to the documentation of your installer if there is any issue

```
# If you use a different installer and it has already created an external work,
# Please skip this command "net-create"
neutron net-create --router:external ext-net

# If you use a different installer and it has already created an external work,
# Change the name "ext-net" to match the name of external network that your installer has created
neutron subnet-create --disable-dhcp --allocation-pool start=198.59.156.251,end=198.59.156.254 --gate
```

OS-NATIVE-SETUP-8: Create Neutron networks `ipv4-int-network1` and `ipv6-int-network2` with port_security disabled

```
neutron net-create --port_security_enabled=False ipv4-int-network1
neutron net-create --port_security_enabled=False ipv6-int-network2
```

OS-NATIVE-SETUP-9: Create IPv4 subnet `ipv4-int-subnet1` in the internal network `ipv4-int-network1`, and associate it to `ipv4-router`.

```
neutron subnet-create --name ipv4-int-subnet1 --dns-nameserver 8.8.8.8 ipv4-int-network1 20.0.0.0/24
neutron router-interface-add ipv4-router ipv4-int-subnet1
```

OS-NATIVE-SETUP-10: Associate the `ext-net` to the Neutron routers `ipv4-router` and `ipv6-router`.

```
# If you use a different installer and it has already created an external work,
# Change the name "ext-net" to match the name of external network that your installer has created
neutron router-gateway-set ipv4-router ext-net
neutron router-gateway-set ipv6-router ext-net
```

OS-NATIVE-SETUP-11: Create two subnets, one IPv4 subnet `ipv4-int-subnet2` and one IPv6 subnet `ipv6-int-subnet2` in `ipv6-int-network2`, and associate both subnets to `ipv6-router`

```
neutron subnet-create --name ipv4-int-subnet2 --dns-nameserver 8.8.8.8 ipv6-int-network2 10.0.0.0/24
neutron subnet-create --name ipv6-int-subnet2 --ip-version 6 --ipv6-ra-mode slaac --ipv6-address-mode
neutron router-interface-add ipv6-router ipv4-int-subnet2
neutron router-interface-add ipv6-router ipv6-int-subnet2
```

OS-NATIVE-SETUP-12: Create a keypair

```
nova keypair-add vRouterKey > ~/vRouterKey
```

OS-NATIVE-SETUP-13: Create ports for vRouter (with some specific MAC address - basically for automation - to know the IPv6 addresses that would be assigned to the port).

```
neutron port-create --name eth0-vRouter --mac-address fa:16:3e:11:11:11 ipv6-int-network2
neutron port-create --name eth1-vRouter --mac-address fa:16:3e:22:22:22 ipv4-int-network1
```

OS-NATIVE-SETUP-14: Create ports for VM1 and VM2.

```
neutron port-create --name eth0-VM1 --mac-address fa:16:3e:33:33:33 ipv4-int-network1
neutron port-create --name eth0-VM2 --mac-address fa:16:3e:44:44:44 ipv4-int-network1
```

OS-NATIVE-SETUP-15: Update `ipv6-router` with routing information to subnet `2001:db8:0:2::/64`

```
neutron router-update ipv6-router --routes type=dict list=true destination=2001:db8:0:2::/64,nexthop=
```

OS-NATIVE-SETUP-16: Boot Service VM (`vRouter`), VM1 and VM2

```
nova boot --image Fedora22 --flavor m1.small --user-data /opt/stack/opnfv_os_ipv6_poc/metadata.txt --
nova list
nova console-log vRouter #Please wait for some 10 to 15 minutes so that necessary packages (like radv
nova boot --image cirros-0.3.4-x86_64-uec --flavor m1.tiny --nic port-id=$(neutron port-list | grep -
nova boot --image cirros-0.3.4-x86_64-uec --flavor m1.tiny --nic port-id=$(neutron port-list | grep -
nova list # Verify that all the VMs are in ACTIVE state.
```

OS-NATIVE-SETUP-17: If all goes well, the IPv6 addresses assigned to the VMs would be as shown as follows:

```
vRouter eth0 interface would have the following IPv6 address: 2001:db8:0:1:f816:3eff:fe11:1111/64
vRouter eth1 interface would have the following IPv6 address: 2001:db8:0:2::1/64
VM1 would have the following IPv6 address: 2001:db8:0:2:f816:3eff:fe33:3333/64
VM2 would have the following IPv6 address: 2001:db8:0:2:f816:3eff:fe44:4444/64
```

OS-NATIVE-SETUP-18: Now we can `SSH` to VMs. You can execute the following command.

```
# 1. Create a floatingip and associate it with VM1, VM2 and vRouter (to the port id that is passed).
#    If you use a different installer and it has already created an external work,
#    Change the name "ext-net" to match the name of external network that your installer has created
neutron floatingip-create --port-id $(neutron port-list | grep -w eth0-VM1 | \
awk '{print $2}') ext-net
neutron floatingip-create --port-id $(neutron port-list | grep -w eth0-VM2 | \
awk '{print $2}') ext-net
neutron floatingip-create --port-id $(neutron port-list | grep -w eth1-vRouter | \
awk '{print $2}') ext-net
```

```
# 2. To know / display the floatingip associated with VM1, VM2 and vRouter.
neutron floatingip-list -F floating_ip_address -F port_id | grep $(neutron port-list | \
grep -w eth0-VM1 | awk '{print $2}') | awk '{print $2}'
neutron floatingip-list -F floating_ip_address -F port_id | grep $(neutron port-list | \
grep -w eth0-VM2 | awk '{print $2}') | awk '{print $2}'
neutron floatingip-list -F floating_ip_address -F port_id | grep $(neutron port-list | \
grep -w eth1-vRouter | awk '{print $2}') | awk '{print $2}'

# 3. To ssh to the vRouter, VM1 and VM2, user can execute the following command.
ssh -i ~/vRouterKey fedora@<floating-ip-of-vRouter>
ssh -i ~/vRouterKey cirros@<floating-ip-of-VM1>
ssh -i ~/vRouterKey cirros@<floating-ip-of-VM2>
```

CHAPTER

THREE

SCENARIO 2 - OPENSTACK + OPEN DAYLIGHT LITHIUM OFFICIAL RELEASE

Scenario 2 is the environment of OpenStack + Open Daylight Lithium SR3 Official Release. Because Lithium SR3 Official Release does not support IPv6 L3 Routing, we need to enable Neutron L3 Agent instead of Open Daylight L3 function, while we still use Open Daylight for L2 switching. Because there is a bug in net-virt provider implementation, we need to use manual configuration to simulate IPv6 external router in our setup.

Please note that although the instructions are based on OpenStack Kilo, they can be applied to Liberty in the same way.

3.1 Infrastructure Setup

In order to set up the service VM as an IPv6 vRouter, we need to prepare 3 hosts, each of which has minimum 8GB RAM and 40GB storage. One host is used as OpenStack Controller Node. The second host is used as Open Daylight Controller Node. And the third one is used as OpenStack Compute Node.

Please **NOTE** that in case of HA (High Availability) deployment model where multiple controller nodes are used, the setup procedure is the same. When `ipv6-router` is created in step **SETUP-SVM-11**, it could be created in any of the controller node. Thus you need to identify in which controller node `ipv6-router` is created in order to manually spawn `radvd` daemon inside the `ipv6-router` namespace in steps **SETUP-SVM-24** through **SETUP-SVM-30**.

For exemplary purpose, we assume:

- The hostname of OpenStack Controller+Network+Compute Node is `opnfv-os-controller`, and the host IP address is `192.168.0.10`
- The hostname of OpenStack Compute Node is `opnfv-os-compute`, and the host IP address is `192.168.0.20`
- The hostname of Open Daylight Controller Node is `opnfv-odl-controller`, and the host IP address is `192.168.0.30`
- We use `opnfv` as username to login.
- We use `devstack` to install OpenStack Kilo. Please note that OpenStack Liberty can be used as well.

The underlay network topology of those 3 hosts are shown as follows in Fig. 3.1:

Please note that the IP address shown in Fig. 3.1 are for exemplary purpose. You need to configure your public IP address connecting to Internet according to your actual network infrastructure. And you need to make sure the private IP address are not conflicting with other subnets.

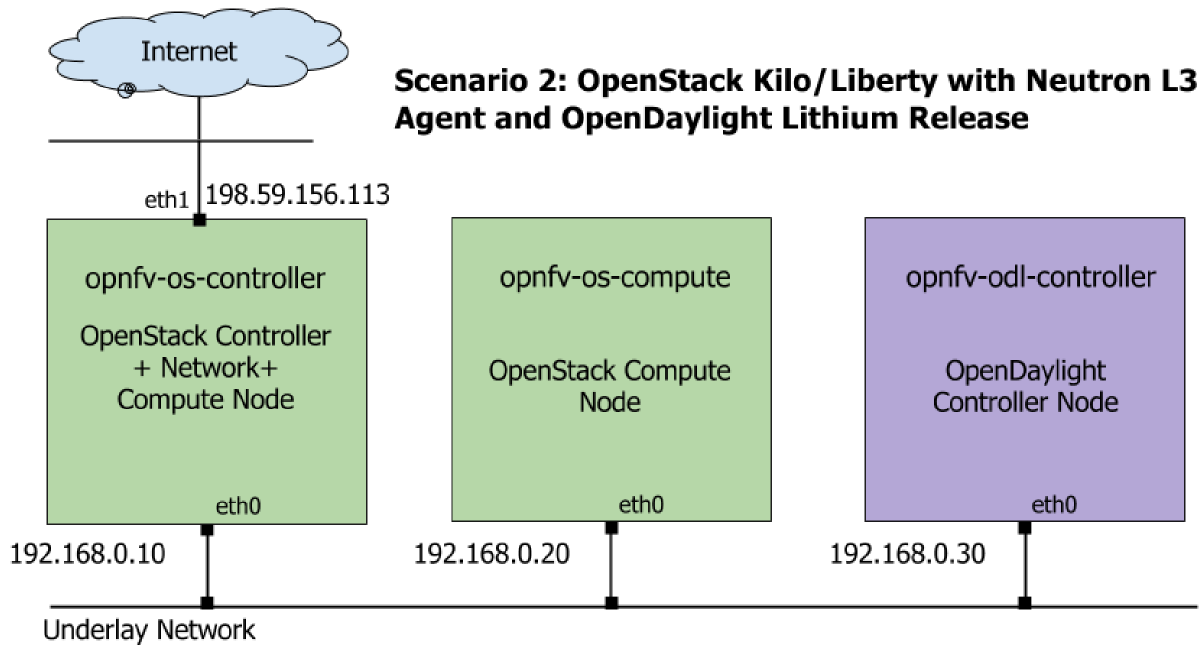

Fig. 3.1: Underlay Network Topology - Scenario 2

3.2 Setting Up Open Daylight Controller Node

For exemplary purpose, we assume:

- The hostname of Open Daylight Controller Node is `opnfv-odl-controller`, and the host IP address is `192.168.0.30`
- CentOS 7 is installed
- We use `opnfv` as username to login.
- Java 7 is installed in directory `/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.85-2.6.1.2.el7_1.x86_64/`

ODL-1: Login to Open Daylight Controller Node with username `opnfv`.

ODL-2: Download the ODL Lithium distribution from `http://www.opendaylight.org/software/downloads`

```
wget https://nexus.opendaylight.org/content/groups/public/org/opendaylight/integration/distribution-k
```

ODL-3: Extract the tar file

```
tar -zxvf distribution-karaf-0.3.3-Lithium-SR3.tar.gz
```

ODL-4: Install Java7

```
sudo yum install -y java-1.7.0-openjdk.x86_64
```

ODL-5 (OPTIONAL): We are using `iptables` instead of `firewalld` but this is optional for the OpenDaylight Controller Node. The objective is to allow all connections on the internal private network (ens160). The same objective can be achieved using firewalld as well. **If you intend to use firewalld, please skip this step and directly go to next step**:

```
sudo systemctl stop firewalld.service
sudo yum remove -y firewalld
```

```
sudo yum install -y iptables-services
sudo touch /etc/sysconfig/iptables
sudo systemctl enable iptables.service
sudo systemctl start iptables.service
sudo iptables -I INPUT 1 -i ens160 -j ACCEPT
sudo iptables -I INPUT -m state --state NEW -p tcp --dport 8181 -j ACCEPT # For ODL DLUX UI
sudo iptables-save > /etc/sysconfig/iptables
```

ODL-6: Open a screen session.

```
screen -S ODL_Controller
```

ODL-7: In the new screen session, change directory to where Open Daylight is installed. Here we use odl directory name and Lithium SR3 installation as an example.

```
cd ~/odl/distribution-karaf-0.3.3-Lithium-SR3/bin
```

ODL-8: Set the JAVA environment variables.

```
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.85-2.6.1.2.el7_1.x86_64/jre
export PATH=$PATH:/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.85-2.6.1.2.el7_1.x86_64/jre/bin
```

ODL-9: Run the karaf shell.

```
./karaf
```

ODL-10: You are now in the Karaf shell of Open Daylight. To explore the list of available features you can execute feature:list. In order to enable Open Daylight with OpenStack, you have to load the odl-ovsdb-openstack feature.

```
opendaylight-user@opnfv>feature:install odl-ovsdb-openstack
```

ODL-11: Verify that OVSDB feature is installed successfully.

```
opendaylight-user@opnfv>feature:list -i | grep ovsdb
odl-ovsdb-openstack | 1.1.1-Lithium-SR1       | x  | ovsdb-1.1.1-Lithium-SR1 | OpenDaylight :: OVSDB
odl-ovsdb-southbound-api  | 1.1.1-Lithium-SR1 | x  | odl-ovsdb-southbound-1.1.1-Lithium-SR1 | OpenDay
odl-ovsdb-southbound-impl | 1.1.1-Lithium-SR1 | x  | odl-ovsdb-southbound-1.1.1-Lithium-SR1 | OpenDay
odl-ovsdb-southbound-impl-rest|1.1.1-Lithium-SR1 | x | odl-ovsdb-southbound-1.1.1-Lithium-SR1| OpenDa
odl-ovsdb-southbound-impl-ui  | 1.1.1-Lithium-SR1| x | odl-ovsdb-southbound-1.1.1-Lithium-SR1| OpenDa
opendaylight-user@opnfv>
```

ODL-12: To view the logs, you can use the following commands (or alternately the file data/log/karaf.log).

```
opendaylight-user@opnfv>log:display
opendaylight-user@opnfv>log:tail
```

ODL-13: To enable ODL DLUX UI, install the following features. Then you can navigate to http://<opnfv-odl-controller IP address>:8181/index.html for DLUX UI. The default username and password is admin/admin.

```
opendaylight-user@opnfv>feature:install odl-dlux-core
```

ODL-14: To exit out of screen session, please use the command CTRL+a followed by d

Note: Do not kill the screen session, it will terminate the ODL controller.

At this moment, Open Daylight has been started successfully.

3.3 Setting Up OpenStack Controller Node

Please **note** that the instructions shown here are using `devstack` installer. If you are an experienced user and installs OpenStack in a different way, you can skip this step and follow the instructions of the method you are using to install OpenStack.

For exemplary purpose, we assume:

- The hostname of OpenStack Controller Node is `opnfv-os-controller`, and the host IP address is `192.168.0.10`
- Ubuntu 14.04 or Fedora 21 is installed
- We use `opnfv` as username to login.
- We use `devstack` to install OpenStack Kilo. Please note that although the instructions are based on OpenStack Kilo, they can be applied to Liberty in the same way.

OS-N-0: Login to OpenStack Controller Node with username `opnfv`

OS-N-1: Update the packages and install git

For **Ubuntu**:

```
sudo apt-get update -y
sudo apt-get install -y git
```

For **Fedora**:

```
sudo yum update -y
sudo yum install -y git
```

OS-N-2: Clone the following GitHub repository to get the configuration and metadata files

```
git clone https://github.com/sridhargaddam/opnfv_os_ipv6_poc.git /opt/stack/opnfv_os_ipv6_poc
```

OS-N-3: Download devstack and switch to stable/kilo branch

```
git clone https://github.com/openstack-dev/devstack.git -b stable/kilo
```

OS-N-4: Start a new terminal, and change directory to where OpenStack is installed.

```
cd ~/devstack
```

OS-N-5: Create a `local.conf` file from the GitHub repo we cloned at **OS-N-2**.

```
cp /opt/stack/opnfv_os_ipv6_poc/scenario2/local.conf.odl.controller ~/devstack/local.conf
```

Please **note** that you need to change the IP address of `ODL_MGR_IP` to point to your actual IP address of Open Daylight Controller.

OS-N-6: Initiate Openstack setup by invoking `stack.sh`

```
./stack.sh
```

OS-N-7: If the setup is successful you would see the following logs on the console. Please note that the IP addresses are all for the purpose of example. Your IP addresses will match the ones of your actual network interfaces.

```
This is your host IP address: 192.168.0.10
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.0.10/
Keystone is serving at http://192.168.0.10:5000/
```

```
The default users are: admin and demo
The password: password
```

Please **note** that The IP addresses above are exemplary purpose. It will show you the actual IP address of your host.

OS-N-8: Assuming that all goes well, you can set `OFFLINE=True` and `RECLONE=no` in `local.conf` to lock the codebase. Devstack uses these configuration parameters to determine if it has to run with the existing codebase or update to the latest copy.

OS-N-9: Source the credentials.

```
opnfv@opnfv-os-controller:~/devstack$ source openrc admin demo
```

Please **NOTE** that the method of sourcing tenant credentials may vary depending on installers. **Please refer to relevant documentation of installers if you encounter any issue**.

OS-N-10: Verify some commands to check if setup is working fine.

```
opnfv@opnfv-os-controller:~/devstack$ nova flavor-list
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| ID | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| 1  | m1.tiny   | 512       | 1    | 0         |      | 1     | 1.0         | True      |
| 2  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True      |
| 3  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 4  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 5  | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
```

Now you can start the Compute node setup.

3.4 Setting Up OpenStack Compute Node

Please **note** that the instructions shown here are using `devstack` installer. If you are an experienced user and installs OpenStack in a different way, you can skip this step and follow the instructions of the method you are using to install OpenStack.

For exemplary purpose, we assume:

- The hostname of OpenStack Compute Node is `opnfv-os-compute`, and the host IP address is `192.168.0.20`
- Ubuntu 14.04 or Fedora 21 is installed
- We use `opnfv` as username to login.
- We use `devstack` to install OpenStack Kilo. Please note that although the instructions are based on OpenStack Kilo, they can be applied to Liberty in the same way.

OS-M-0: Login to OpenStack Compute Node with username `opnfv`

OS-M-1: Update the packages and install git

For **Ubuntu**:

```
sudo apt-get update -y
sudo apt-get install -y git
```

For **Fedora**:

```
sudo yum update -y
sudo yum install -y git
```

OS-M-2: Clone the following GitHub repository to get the configuration and metadata files

```
git clone https://github.com/sridhargaddam/opnfv_os_ipv6_poc.git /opt/stack/opnfv_os_ipv6_poc
```

OS-M-3: Download devstack and switch to stable/kilo branch

```
git clone https://github.com/openstack-dev/devstack.git -b stable/kilo
```

OS-M-4: Start a new terminal, and change directory to where OpenStack is installed.

```
cd ~/devstack
```

OS-M-5: Create a `local.conf` file from the GitHub repo we cloned at **OS-M-2**.

```
cp /opt/stack/opnfv_os_ipv6_poc/scenario2/local.conf.odl.compute ~/devstack/local.conf
```

Please Note:

- Note 1: you need to change the IP address of `SERVICE_HOST` to point to your actual IP address of OpenStack Controller.
- Note 2: you need to change the IP address of `ODL_MGR_IP` to point to your actual IP address of Open Daylight Controller.

OS-M-6: Initiate Openstack setup by invoking `stack.sh`

```
./stack.sh
```

OS-M-7: Assuming that all goes well, you should see the following output.

```
This is your host IP address: 192.168.0.20
This is your host IPv6 address: ::1
```

Please **note** that The IP addresses above are exemplary purpose. It will show you the actual IP address of your host.

You can set `OFFLINE=True` and `RECLONE=no` in `local.conf` to lock the codebase. Devstack uses these configuration parameters to determine if it has to run with the existing codebase or update to the latest copy.

OS-M-8: Source the credentials.

```
opnfv@opnfv-os-compute:~/devstack$ source openrc admin demo
```

Please **NOTE** that the method of sourcing tenant credentials may vary depending on installers. **Please refer to relevant documentation of installers if you encounter any issue**.

OS-M-9: You can verify that OpenStack is set up correctly by showing hypervisor list

```
opnfv@opnfv-os-compute:~/devstack$ nova hypervisor-list
+----+------------------------------------+---------+------------+
| ID | Hypervisor hostname                | State   | Status     |
+----+------------------------------------+---------+------------+
| 1  | opnfv-os-controller                | up      | enabled    |
| 2  | opnfv-os-compute                   | up      | enabled    |
+----+------------------------------------+---------+------------+
```

Now you can start to set up the service VM as an IPv6 vRouter in the environment of OpenStack and Open Daylight.

3.5 Setting Up a Service VM as an IPv6 vRouter

Now we can start to set up a service VM as an IPv6 vRouter. For exemplary purpose, we assume:

- The hostname of Open Daylight Controller Node is `opnfv-odl-controller`, and the host IP address is `192.168.0.30`
- The hostname of OpenStack Controller Node is `opnfv-os-controller`, and the host IP address is `192.168.0.10`
- The hostname of OpenStack Compute Node is `opnfv-os-compute`, and the host IP address is `192.168.0.20`
- We use `opnfv` as username to login.
- We use `devstack` to install OpenStack Kilo, and the directory is `~/devstack`
- Note: all IP addresses as shown below are for exemplary purpose.

3.5.1 Note: Disable Security Groups in OpenStack ML2 Setup

Please note that Security Groups feature has been disabled automatically through `local.conf` configuration file during the setup procedure of OpenStack in both Controller Node and Compute Node using `devstack`.

If you are installing OpenStack using a different installer (i.e. not with `devstack`), please make sure that Security Groups are disabled in the setup.

Please refer to here **for the notes in** `Section 2.4`, **steps** `OS-NATIVE-SEC-1` **through** `OS-NATIVE-SEC-3`.

3.5.2 Source the Credentials in OpenStack Controller Node

SETUP-SVM-1: Login with username `opnfv` in OpenStack Controller Node `opnfv-os-controller`. Start a new terminal, and change directory to where OpenStack is installed.

```
cd ~/devstack
```

SETUP-SVM-2: Source the credentials.

```
# source the tenant credentials in devstack
opnfv@opnfv-os-controller:~/devstack$ source openrc admin demo
```

Please **NOTE** that the method of sourcing tenant credentials may vary depending on installers. **Please refer to relevant documentation of installers if you encounter any issue**.

3.5.3 Add External Connectivity to `br-ex`

Because we need to manually create networks/subnets to achieve the IPv6 vRouter, we have used the flag `NEUTRON_CREATE_INITIAL_NETWORKS=False` in `local.conf` file. When this flag is set to False, `devstack` does not create any networks/subnets during the setup phase.

Now we have to move the physical interface (i.e. the public network interface) to `br-ex`, including moving the public IP address and setting up default route. **Please note that this step may already have been done when you use a different installer to deploy OpenStack because that installer may have already moved the physical interface to** `br-ex` **during deployment**.

In OpenStack Controller Node `opnfv-os-controller`, `eth1` is configured to provide external/public connectivity for both IPv4 and IPv6 (optional). So let us add this interface to `br-ex` and move the IP address, including the default route from `eth1` to `br-ex`.

SETUP-SVM-3: Add `eth1` to `br-ex` and move the IP address and the default route from `eth1` to `br-ex`

```
sudo ip addr del 198.59.156.113/24 dev eth1
sudo ovs-vsctl add-port br-ex eth1
sudo ifconfig eth1 up
sudo ip addr add 198.59.156.113/24 dev br-ex
sudo ifconfig br-ex up
sudo ip route add default via 198.59.156.1 dev br-ex
```

Please note that:

- The IP address `198.59.156.113` and related subnet and gateway addressed in the command below are for exemplary purpose. **Please replace them with the IP addresses of your actual network**.
- **This can be automated in /etc/network/interfaces**.

SETUP-SVM-4: Verify that `br-ex` now has the original external IP address, and that the default route is on `br-ex`

```
opnfv@opnfv-os-controller:~/devstack$ ip a s br-ex
38: br-ex: <BROADCAST,UP,LOWER_UP> mtu 1430 qdisc noqueue state UNKNOWN group default
    link/ether 00:50:56:82:42:d1 brd ff:ff:ff:ff:ff:ff
    inet 198.59.156.113/24 brd 198.59.156.255 scope global br-ex
       valid_lft forever preferred_lft forever
    inet6 fe80::543e:28ff:fe70:4426/64 scope link
       valid_lft forever preferred_lft forever
opnfv@opnfv-os-controller:~/devstack$
opnfv@opnfv-os-controller:~/devstack$ ip route
default via 198.59.156.1 dev br-ex
192.168.0.0/24 dev eth0  proto kernel  scope link  src 192.168.0.10
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1
198.59.156.0/24 dev br-ex  proto kernel  scope link  src 198.59.156.113
```

Please note that The IP addresses above are exemplary purpose

3.5.4 Create IPv4 Subnet and Router with External Connectivity

SETUP-SVM-5: Create a Neutron router `ipv4-router` which needs to provide external connectivity.

```
neutron router-create ipv4-router
```

SETUP-SVM-6: Create an external network/subnet `ext-net` using the appropriate values based on the data-center physical network setup.

Please **NOTE** that if you use a different installer, i.e. NOT `devstack`, your installer may have already created an external network during installation. Under this circumstance, you may only need to create the subnet of `ext-net`. When you create the subnet, you must use the same name of external network that your installer creates.

```
# If you use a different installer and it has already created an external work,
# Please skip this command "net-create"
neutron net-create --router:external ext-net

# If you use a different installer and it has already created an external work,
# Change the name "ext-net" to match the name of external network that your installer has created
neutron subnet-create --disable-dhcp --allocation-pool start=198.59.156.251,end=198.59.156.254 --gate
```

Please note that the IP addresses in the command above are for exemplary purpose. **Please replace the IP addresses of your actual network**.

SETUP-SVM-7: Associate the `ext-net` to the Neutron router `ipv4-router`.

```
# If you use a different installer and it has already created an external work,
# Change the name "ext-net" to match the name of external network that your installer has created
neutron router-gateway-set ipv4-router ext-net
```

SETUP-SVM-8: Create an internal/tenant IPv4 network `ipv4-int-network1`

```
neutron net-create ipv4-int-network1
```

SETUP-SVM-9: Create an IPv4 subnet `ipv4-int-subnet1` in the internal network `ipv4-int-network1`

```
neutron subnet-create --name ipv4-int-subnet1 --dns-nameserver 8.8.8.8 ipv4-int-network1 20.0.0.0/24
```

SETUP-SVM-10: Associate the IPv4 internal subnet `ipv4-int-subnet1` to the Neutron router `ipv4-router`.

```
neutron router-interface-add ipv4-router ipv4-int-subnet1
```

3.5.5 Create IPv6 Subnet and Router with External Connectivity

Now, let us create a second neutron router where we can "manually" spawn a `radvd` daemon to simulate an external IPv6 router.

SETUP-SVM-11: Create a second Neutron router `ipv6-router` which needs to provide external connectivity

```
neutron router-create ipv6-router
```

SETUP-SVM-12: Associate the `ext-net` to the Neutron router `ipv6-router`

```
# If you use a different installer and it has already created an external work,
# Change the name "ext-net" to match the name of external network that your installer has created
neutron router-gateway-set ipv6-router ext-net
```

SETUP-SVM-13: Create a second internal/tenant IPv4 network `ipv4-int-network2`

```
neutron net-create ipv4-int-network2
```

SETUP-SVM-14: Create an IPv4 subnet `ipv4-int-subnet2` for the `ipv6-router` internal network `ipv4-int-network2`

```
neutron subnet-create --name ipv4-int-subnet2 --dns-nameserver 8.8.8.8 ipv4-int-network2 10.0.0.0/24
```

SETUP-SVM-15: Associate the IPv4 internal subnet `ipv4-int-subnet2` to the Neutron router `ipv6-router`.

```
neutron router-interface-add ipv6-router ipv4-int-subnet2
```

3.5.6 Prepare Image, Metadata and Keypair for Service VM

SETUP-SVM-16: Download `fedora22` image which would be used as `vRouter`

```
wget https://download.fedoraproject.org/pub/fedora/linux/releases/22/Cloud/x86_64/Images/Fedora-Clou

glance image-create --name 'Fedora22' --disk-format qcow2 --container-format bare --file ./Fedora-Cl
```

SETUP-SVM-17: Create a keypair

```
nova keypair-add vRouterKey > ~/vRouterKey
```

SETUP-SVM-18: Create ports for `vRouter` and both the VMs with some specific MAC addresses.

```
neutron port-create --name eth0-vRouter --mac-address fa:16:3e:11:11:11 ipv4-int-network2
neutron port-create --name eth1-vRouter --mac-address fa:16:3e:22:22:22 ipv4-int-network1
neutron port-create --name eth0-VM1 --mac-address fa:16:3e:33:33:33 ipv4-int-network1
neutron port-create --name eth0-VM2 --mac-address fa:16:3e:44:44:44 ipv4-int-network1
```

3.5.7 Boot Service VM (`vRouter`) with `eth0` on `ipv4-int-network2` and `eth1` on `ipv4-int-network1`

Let us boot the service VM (`vRouter`) with `eth0` interface on `ipv4-int-network2` connecting to `ipv6-router`, and `eth1` interface on `ipv4-int-network1` connecting to `ipv4-router`.

SETUP-SVM-19: Boot the `vRouter` using `Fedora22` image on the OpenStack Compute Node with hostname `opnfv-os-compute`

```
nova boot --image Fedora22 --flavor m1.small --user-data /opt/stack/opnfv_os_ipv6_poc/metadata.txt --
```

Please **note** that `/opt/stack/opnfv_os_ipv6_poc/metadata.txt` is used to enable the `vRouter` to automatically spawn a `radvd`, and

- Act as an IPv6 vRouter which advertises the RA (Router Advertisements) with prefix `2001:db8:0:2::/64` on its internal interface (`eth1`).
- Forward IPv6 traffic from internal interface (`eth1`)

SETUP-SVM-20: Verify that `Fedora22` image boots up successfully and vRouter has `ssh` keys properly injected

```
nova list
nova console-log vRouter
```

Please note that **it may take a few minutes** for the necessary packages to get installed and `ssh` keys to be injected.

```
# Sample Output
[  762.884523] cloud-init[871]: ec2: ############################################################
[  762.909634] cloud-init[871]: ec2: -----BEGIN SSH HOST KEY FINGERPRINTS-----
[  762.931626] cloud-init[871]: ec2: 2048 e3:dc:3d:4a:bc:b6:b0:77:75:a1:70:a3:d0:2a:47:a9   (RSA)
[  762.957380] cloud-init[871]: ec2: -----END SSH HOST KEY FINGERPRINTS-----
[  762.979554] cloud-init[871]: ec2: ############################################################
```

3.5.8 Boot Two Other VMs in `ipv4-int-network1`

In order to verify that the setup is working, let us create two cirros VMs with `eth1` interface on the `ipv4-int-network1`, i.e., connecting to `vRouter` `eth1` interface for internal network.

We will have to configure appropriate `mtu` on the VMs' interface by taking into account the tunneling overhead and any physical switch requirements. If so, push the `mtu` to the VM either using `dhcp` options or via `meta-data`.

SETUP-SVM-21: Create VM1 on OpenStack Controller Node with hostname `opnfv-os-controller`

```
nova boot --image cirros-0.3.4-x86_64-uec --flavor m1.tiny --nic port-id=$(neutron port-list | grep -
```

SETUP-SVM-22: Create VM2 on OpenStack Compute Node with hostname `opnfv-os-compute`

```
nova boot --image cirros-0.3.4-x86_64-uec --flavor m1.tiny --nic port-id=$(neutron port-list | grep -
```

SETUP-SVM-23: Confirm that both the VMs are successfully booted.

```
nova list
nova console-log VM1
nova console-log VM2
```

3.5.9 Spawn RADVD in ipv6-router

Let us manually spawn a `radvd` daemon inside `ipv6-router` namespace to simulate an external router. First of all, we will have to identify the `ipv6-router` namespace and move to the namespace.

Please **NOTE** that in case of HA (High Availability) deployment model where multiple controller nodes are used, `ipv6-router` created in step **SETUP-SVM-11** could be in any of the controller node. Thus you need to identify in which controller node `ipv6-router` is created in order to manually spawn `radvd` daemon inside the `ipv6-router` namespace in steps **SETUP-SVM-24** through **SETUP-SVM-30**. The following command in Neutron will display the controller on which the `ipv6-router` is spawned.

```
neutron l3-agent-list-hosting-router ipv6-router
```

Then you login to that controller and execute steps **SETUP-SVM-24** through **SETUP-SVM-30**

SETUP-SVM-24: identify the `ipv6-router` namespace and move to the namespace

```
sudo ip netns exec qrouter-$(neutron router-list | grep -w ipv6-router | awk '{print $2}') bash
```

SETUP-SVM-25: Upon successful execution of the above command, you will be in the router namespace. Now let us configure the IPv6 address on the <qr-xxx> interface.

```
export router_interface=$(ip a s | grep -w "global qr-*" | awk '{print $7}')
ip -6 addr add 2001:db8:0:1::1 dev $router_interface
```

SETUP-SVM-26: Update the sample file `/opt/stack/opnfv_os_ipv6_poc/scenario2/radvd.conf` with `$router_interface`.

```
cp /opt/stack/opnfv_os_ipv6_poc/scenario2/radvd.conf /tmp/radvd.$router_interface.conf
sed -i 's/$router_interface/'$router_interface'/g' /tmp/radvd.$router_interface.conf
```

SETUP-SVM-27: Spawn a `radvd` daemon to simulate an external router. This `radvd` daemon advertises an IPv6 subnet prefix of `2001:db8:0:1::/64` using RA (Router Advertisement) on its $router_interface so that `eth0` interface of `vRouter` automatically configures an IPv6 SLAAC address.

```
$radvd -C /tmp/radvd.$router_interface.conf -p /tmp/br-ex.pid.radvd -m syslog
```

SETUP-SVM-28: Add an IPv6 downstream route pointing to the `eth0` interface of vRouter.

```
ip -6 route add 2001:db8:0:2::/64 via 2001:db8:0:1:f816:3eff:fe11:1111
```

SETUP-SVM-29: The routing table should now look similar to something shown below.

```
ip -6 route show
2001:db8:0:1::1 dev qr-42968b9e-62 proto kernel metric 256
2001:db8:0:1::/64 dev qr-42968b9e-62 proto kernel metric 256 expires 86384sec
2001:db8:0:2::/64 via 2001:db8:0:1:f816:3eff:fe11:1111 dev qr-42968b9e-62 proto ra metric 1024 expire
fe80::/64 dev qg-3736e0c7-7c proto kernel metric 256
fe80::/64 dev qr-42968b9e-62 proto kernel metric 256
```

SETUP-SVM-30: If all goes well, the IPv6 addresses assigned to the VMs would be as shown as follows:

```
vRouter eth0 interface would have the following IPv6 address: 2001:db8:0:1:f816:3eff:fe11:1111/64
vRouter eth1 interface would have the following IPv6 address: 2001:db8:0:2::1/64
VM1 would have the following IPv6 address: 2001:db8:0:2:f816:3eff:fe33:3333/64
VM2 would have the following IPv6 address: 2001:db8:0:2:f816:3eff:fe44:4444/64
```

3.5.10 Testing to Verify Setup Complete

Now, let us `SSH` to those VMs, e.g. VM1 and / or VM2 and / or vRouter, to confirm that it has successfully configured the IPv6 address using `SLAAC` with prefix `2001:db8:0:2::/64` from `vRouter`.

We use `floatingip` mechanism to achieve `SSH`.

SETUP-SVM-31: Now we can `SSH` to VMs. You can execute the following command.

```
# 1. Create a floatingip and associate it with VM1, VM2 and vRouter (to the port id that is passed).
#    If you use a different installer and it has already created an external work,
#    Change the name "ext-net" to match the name of external network that your installer has created
neutron floatingip-create --port-id $(neutron port-list | grep -w eth0-VM1 | \
awk '{print $2}') ext-net
neutron floatingip-create --port-id $(neutron port-list | grep -w eth0-VM2 | \
awk '{print $2}') ext-net
neutron floatingip-create --port-id $(neutron port-list | grep -w eth1-vRouter | \
awk '{print $2}') ext-net

# 2. To know / display the floatingip associated with VM1, VM2 and vRouter.
neutron floatingip-list -F floating_ip_address -F port_id | grep $(neutron port-list | \
grep -w eth0-VM1 | awk '{print $2}') | awk '{print $2}'
neutron floatingip-list -F floating_ip_address -F port_id | grep $(neutron port-list | \
grep -w eth0-VM2 | awk '{print $2}') | awk '{print $2}'
neutron floatingip-list -F floating_ip_address -F port_id | grep $(neutron port-list | \
grep -w eth1-vRouter | awk '{print $2}') | awk '{print $2}'

# 3. To ssh to the vRouter, VM1 and VM2, user can execute the following command.
ssh -i ~/vRouterKey fedora@<floating-ip-of-vRouter>
ssh -i ~/vRouterKey cirros@<floating-ip-of-VM1>
ssh -i ~/vRouterKey cirros@<floating-ip-of-VM2>
```

If everything goes well, `ssh` will be successful and you will be logged into those VMs. Run some commands to verify that IPv6 addresses are configured on `eth0` interface.

SETUP-SVM-32: Show an IPv6 address with a prefix of `2001:db8:0:2::/64`

```
ip address show
```

SETUP-SVM-33: ping some external IPv6 address, e.g. `ipv6-router`

```
ping6 2001:db8:0:1::1
```

If the above ping6 command succeeds, it implies that `vRouter` was able to successfully forward the IPv6 traffic to reach external `ipv6-router`.

3.5.11 Next Steps

Congratulations, you have completed the setup of using a service VM to act as an IPv6 vRouter. This setup allows further open innovation by any 3rd-party. Please refer to relevant sections in User's Guide for further value-added services on this IPv6 vRouter.

CHAPTER

FOUR

SCENARIO 3 - OPENSTACK + OPEN DAYLIGHT LITHIUM WITH PATCH OF BUG FIX

Scenario 3 is the environment of OpenStack + Open Daylight Lithium, which is similar to Scenario 2. However, we use an Open Daylight Lithium controller which is built from the latest stable/Lithium branch that includes the fix of a bug. In this scenario, we can fully automate the setup similar to Scenario 1.

4.1 Infrastructure Setup

In order to set up the service VM as an IPv6 vRouter, we need to prepare 3 hosts, each of which has minimum 8GB RAM and 40GB storage. One host is used as OpenStack Controller Node. The second host is used as Open Daylight Controller Node. And the third one is used as OpenStack Compute Node.

Please NOTE that Although the deployment model of single controller node is assumed, in case of HA (High Availability) deployment model where multiple controller nodes are used, there is no impact and the setup procedure is the same.

For exemplary purpose, we assume:

- The hostname of OpenStack Controller+Network+Compute Node is `opnfv-os-controller`, and the host IP address is `192.168.0.10`
- The hostname of OpenStack Compute Node is `opnfv-os-compute`, and the host IP address is `192.168.0.20`
- The hostname of Open Daylight Controller Node is `opnfv-odl-controller`, and the host IP address is `192.168.0.30`
- We use `opnfv` as username to login.
- We use `devstack` to install OpenStack Kilo. Please note that OpenStack Liberty can be used as well.

The underlay network topology of those 3 hosts are shown as follows in Fig. 4.1:

Please note that the IP address shown in Fig. 4.1 are for exemplary purpose. You need to configure your public IP address connecting to Internet according to your actual network infrastructure. And you need to make sure the private IP address are not conflicting with other subnets.

4.2 Setting Up Open Daylight Controller Node

For exemplary purpose, we assume:

- The hostname of Open Daylight Controller Node is `opnfv-odl-controller`, and the host IP address is `192.168.0.30`

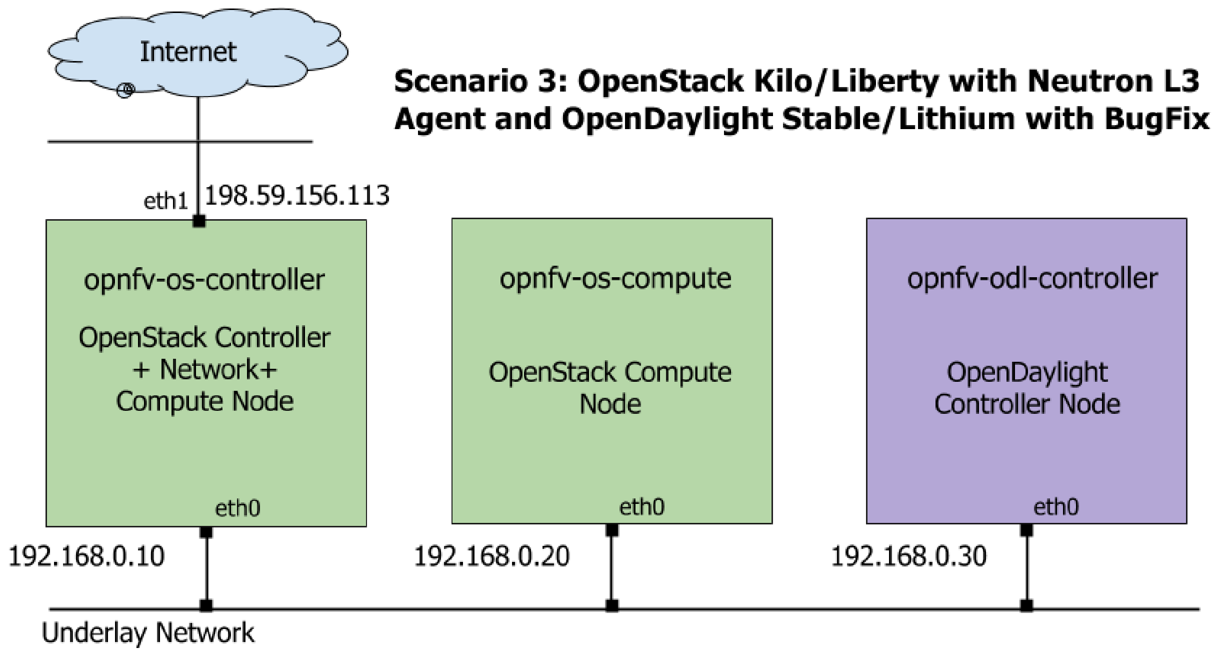

Fig. 4.1: Underlay Network Topology - Scenario 3

- CentOS 7 is installed
- We use `opnfv` as username to login.
- Java 7 is installed in directory `/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.85-2.6.1.2.el7_1.x86_64/`

Please **NOTE** that this Scenario 3 uses an Open Daylight Lithium controller which is built from the latest `stable/Lithium` branch that includes the fix of a bug, there is a **prerequisite** that you are able to build this Open Daylight Lithium Controller from the the latest `stable/Lithium` branch. Please refer to relevant documentation from Open Daylight.

ODL-1: **Prerequisite** - build Open Daylight Lithium Controller from the the latest `stable/Lithium` branch, and make it available for step **ODL-3**.

ODL-2: Login to Open Daylight Controller Node with username `opnfv`.

ODL-3: Extract the tar file of your custom build of Open Daylight Lithium Controller from step **ODL-1**.

```
tar -zxvf <filename-of-your-custom-build>.tar.gz
```

ODL-4: Install Java7

```
sudo yum install -y java-1.7.0-openjdk.x86_64
```

ODL-5 (OPTIONAL): We are using `iptables` instead of `firewalld` but this is optional for the OpenDaylight Controller Node. The objective is to allow all connections on the internal private network (ens160). The same objective can be achieved using firewalld as well. **If you intend to use firewalld, please skip this step and directly go to next step**:

```
sudo systemctl stop firewalld.service
sudo yum remove -y firewalld
sudo yum install -y iptables-services
sudo touch /etc/sysconfig/iptables
sudo systemctl enable iptables.service
sudo systemctl start iptables.service
```

```
sudo iptables -I INPUT 1 -i ens160 -j ACCEPT
sudo iptables -I INPUT -m state --state NEW -p tcp --dport 8181 -j ACCEPT # For ODL DLUX UI
sudo iptables-save > /etc/sysconfig/iptables
```

ODL-6: Open a screen session.

```
screen -S ODL_Controller
```

ODL-7: In the new screen session, change directory to where Open Daylight is installed. Here we use `odl` directory name and `Lithium SR3` installation as an example.

```
cd ~/odl/distribution-karaf-0.3.3-Lithium-SR3/bin
```

ODL-8: Set the JAVA environment variables.

```
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.85-2.6.1.2.el7_1.x86_64/jre
export PATH=$PATH:/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.85-2.6.1.2.el7_1.x86_64/jre/bin
```

ODL-9: Run the `karaf` shell.

```
./karaf
```

ODL-10: You are now in the Karaf shell of Open Daylight. To explore the list of available features you can execute `feature:list`. In order to enable Open Daylight with OpenStack, you have to load the `odl-ovsdb-openstack` feature.

```
opendaylight-user@opnfv>feature:install odl-ovsdb-openstack
```

ODL-11: Verify that OVSDB feature is installed successfully.

```
opendaylight-user@opnfv>feature:list -i | grep ovsdb
odl-ovsdb-openstack | 1.1.1-Lithium-SR1       | x   | ovsdb-1.1.1-Lithium-SR1 | OpenDaylight :: OVSDB
odl-ovsdb-southbound-api   | 1.1.1-Lithium-SR1 | x   | odl-ovsdb-southbound-1.1.1-Lithium-SR1 | OpenDay
odl-ovsdb-southbound-impl  | 1.1.1-Lithium-SR1 | x   | odl-ovsdb-southbound-1.1.1-Lithium-SR1 | OpenDay
odl-ovsdb-southbound-impl-rest|1.1.1-Lithium-SR1 | x | odl-ovsdb-southbound-1.1.1-Lithium-SR1| OpenDa
odl-ovsdb-southbound-impl-ui   | 1.1.1-Lithium-SR1| x | odl-ovsdb-southbound-1.1.1-Lithium-SR1| OpenDa
opendaylight-user@opnfv>
```

ODL-12: To view the logs, you can use the following commands (or alternately the file data/log/karaf.log).

```
opendaylight-user@opnfv>log:display
opendaylight-user@opnfv>log:tail
```

ODL-13: To enable ODL DLUX UI, install the following features. Then you can navigate to `http://<opnfv-odl-controller IP address>:8181/index.html` for DLUX UI. The default username and password is `admin/admin`.

```
opendaylight-user@opnfv>feature:install odl-dlux-core
```

ODL-14: To exit out of screen session, please use the command `CTRL+a` followed by `d`

Note: Do not kill the screen session, it will terminate the ODL controller.

At this moment, Open Daylight has been started successfully.

4.3 Setting Up OpenStack Controller Node

Please **note** that the instructions shown here are using `devstack` installer. If you are an experienced user and installs OpenStack in a different way, you can skip this step and follow the instructions of the method you are using to install

OpenStack.

For exemplary purpose, we assume:

- The hostname of OpenStack Controller Node is `opnfv-os-controller`, and the host IP address is `192.168.0.10`
- Ubuntu 14.04 or Fedora 21 is installed
- We use `opnfv` as username to login.
- We use `devstack` to install OpenStack Kilo. Please note that although the instructions are based on OpenStack Kilo, they can be applied to Liberty in the same way.

OS-N-0: Login to OpenStack Controller Node with username `opnfv`

OS-N-1: Update the packages and install git

For **Ubuntu**:

```
sudo apt-get update -y
sudo apt-get install -y git
```

For **Fedora**:

```
sudo yum update -y
sudo yum install -y git
```

OS-N-2: Clone the following GitHub repository to get the configuration and metadata files

```
git clone https://github.com/sridhargaddam/opnfv_os_ipv6_poc.git /opt/stack/opnfv_os_ipv6_poc
```

OS-N-3: Download devstack and switch to stable/kilo branch

```
git clone https://github.com/openstack-dev/devstack.git -b stable/kilo
```

OS-N-4: Start a new terminal, and change directory to where OpenStack is installed.

```
cd ~/devstack
```

OS-N-5: Create a `local.conf` file from the GitHub repo we cloned at **OS-N-2**.

```
cp /opt/stack/opnfv_os_ipv6_poc/scenario2/local.conf.odl.controller ~/devstack/local.conf
```

Please **note** that:

- Note 1: Because Scenario 3 and Scenario 2 are essentially the same, and their only difference is using different build of Open Daylight, they share the same `local.conf` file of OpenStack.
- Note 2: You need to change the IP address of `ODL_MGR_IP` to point to your actual IP address of Open Daylight Controller.

OS-N-6: Initiate Openstack setup by invoking `stack.sh`

```
./stack.sh
```

OS-N-7: If the setup is successful you would see the following logs on the console. Please note that the IP addresses are all for the purpose of example. Your IP addresses will match the ones of your actual network interfaces.

```
This is your host IP address: 192.168.0.10
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.0.10/
Keystone is serving at http://192.168.0.10:5000/
The default users are: admin and demo
The password: password
```

Please **note** that The IP addresses above are exemplary purpose. It will show you the actual IP address of your host.

OS-N-8: Assuming that all goes well, you can set `OFFLINE=True` and `RECLONE=no` in `local.conf` to lock the codebase. Devstack uses these configuration parameters to determine if it has to run with the existing codebase or update to the latest copy.

OS-N-9: Source the credentials.

```
opnfv@opnfv-os-controller:~/devstack$ source openrc admin demo
```

Please **NOTE** that the method of sourcing tenant credentials may vary depending on installers. **Please refer to relevant documentation of installers if you encounter any issue**.

OS-N-10: Verify some commands to check if setup is working fine.

```
opnfv@opnfv-os-controller:~/devstack$ nova flavor-list
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| ID | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| 1  | m1.tiny   | 512       | 1    | 0         |      | 1     | 1.0         | True      |
| 2  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True      |
| 3  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 4  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 5  | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
```

Now you can start the Compute node setup.

4.4 Setting Up OpenStack Compute Node

Please **note** that the instructions shown here are using `devstack` installer. If you are an experienced user and installs OpenStack in a different way, you can skip this step and follow the instructions of the method you are using to install OpenStack.

For exemplary purpose, we assume:

- The hostname of OpenStack Compute Node is `opnfv-os-compute`, and the host IP address is `192.168.0.20`
- Ubuntu 14.04 or Fedora 21 is installed
- We use `opnfv` as username to login.
- We use `devstack` to install OpenStack Kilo. Please note that although the instructions are based on OpenStack Kilo, they can be applied to Liberty in the same way.

OS-M-0: Login to OpenStack Compute Node with username `opnfv`

OS-M-1: Update the packages and install git

For **Ubuntu**:

```
sudo apt-get update -y
sudo apt-get install -y git
```

For **Fedora**:

```
sudo yum update -y
sudo yum install -y git
```

OS-M-2: Clone the following GitHub repository to get the configuration and metadata files

```
git clone https://github.com/sridhargaddam/opnfv_os_ipv6_poc.git /opt/stack/opnfv_os_ipv6_poc
```

OS-M-3: Download devstack and switch to stable/kilo branch

```
git clone https://github.com/openstack-dev/devstack.git -b stable/kilo
```

OS-M-4: Start a new terminal, and change directory to where OpenStack is installed.

```
cd ~/devstack
```

OS-M-5: Create a `local.conf` file from the GitHub repo we cloned at **OS-M-2**.

```
cp /opt/stack/opnfv_os_ipv6_poc/scenario2/local.conf.odl.compute ~/devstack/local.conf
```

Please Note:

- Note 1: Because Scenario 3 and Scenario 2 are essentially the same, and their only difference is using different build of Open Daylight, they share the same `local.conf` file of OpenStack.
- Note 2: you need to change the IP address of `SERVICE_HOST` to point to your actual IP address of OpenStack Controller.
- Note 3: you need to change the IP address of `ODL_MGR_IP` to point to your actual IP address of Open Daylight Controller.

OS-M-6: Initiate Openstack setup by invoking `stack.sh`

```
./stack.sh
```

OS-M-7: Assuming that all goes well, you should see the following output.

```
This is your host IP address: 192.168.0.20
This is your host IPv6 address: ::1
```

Please **note** that The IP addresses above are exemplary purpose. It will show you the actual IP address of your host.

You can set `OFFLINE=True` and `RECLONE=no` in `local.conf` to lock the codebase. Devstack uses these configuration parameters to determine if it has to run with the existing codebase or update to the latest copy.

OS-M-8: Source the credentials.

```
opnfv@opnfv-os-compute:~/devstack$ source openrc admin demo
```

Please **NOTE** that the method of sourcing tenant credentials may vary depending on installers. **Please refer to relevant documentation of installers if you encounter any issue**.

OS-M-9: You can verify that OpenStack is set up correctly by showing hypervisor list

```
opnfv@opnfv-os-compute:~/devstack$ nova hypervisor-list
+----+------------------------------------+---------+------------+
| ID | Hypervisor hostname                | State   | Status     |
+----+------------------------------------+---------+------------+
| 1  | opnfv-os-controller                | up      | enabled |
| 2  | opnfv-os-compute                   | up      | enabled |
+----+------------------------------------+---------+------------+
```

Now you can start to set up the service VM as an IPv6 vRouter in the environment of OpenStack and Open Daylight.

4.5 Setting Up a Service VM as an IPv6 vRouter

Now we can start to set up a service VM as an IPv6 vRouter. For exemplary purpose, we assume:

- The hostname of Open Daylight Controller Node is `opnfv-odl-controller`, and the host IP address is `192.168.0.30`
- The hostname of OpenStack Controller Node is `opnfv-os-controller`, and the host IP address is `192.168.0.10`
- The hostname of OpenStack Compute Node is `opnfv-os-compute`, and the host IP address is `192.168.0.20`
- We use `opnfv` as username to login.
- We use `devstack` to install OpenStack Kilo, and the directory is `~/devstack`
- Note: all IP addresses as shown below are for exemplary purpose.

4.5.1 Note: Disable Security Groups in OpenStack ML2 Setup

Please note that Security Groups feature has been disabled automatically through `local.conf` configuration file during the setup procedure of OpenStack in both Controller Node and Compute Node using `devstack`.

If you are installing OpenStack using a different installer (i.e. not with `devstack`), please make sure that Security Groups are disabled in the setup.

Please refer to here **for the notes in** `Section 2.4`, **steps** `OS-NATIVE-SEC-1` **through** `OS-NATIVE-SEC-3`.

4.5.2 Set Up Service VM as IPv6 vRouter

SCENARIO-3-SETUP-1: Now we assume that OpenStack multi-node setup is up and running. The following commands should be executed:

```
cd ~/devstack

# source the tenant credentials in devstack
source openrc admin demo
```

Please **NOTE** that the method of sourcing tenant credentials may vary depending on installers. **Please refer to relevant documentation of installers if you encounter any issue**.

SCENARIO-3-SETUP-2: Download `fedora22` image which would be used for `vRouter`

```
wget https://download.fedoraproject.org/pub/fedora/linux/releases/22/Cloud/x86_64/Images/Fedora-Clouc
```

SCENARIO-3-SETUP-3: Import Fedora22 image to `glance`

```
glance image-create --name 'Fedora22' --disk-format qcow2 --container-format bare --file ./Fedora-Clo
```

SCENARIO-3-SETUP-4: Now we have to move the physical interface (i.e. the public network interface) to `br-ex`, including moving the public IP address and setting up default route. **Please note that this step may already have been done when you use a different installer to deploy OpenStack because that installer may have already moved the physical interface to** `br-ex` **during deployment**.

Because our `opnfv-os-controller` node has two interfaces `eth0` and `eth1`, and `eth1` is used for external connectivity, move the IP address of `eth1` to `br-ex`.

Please note that the IP address `198.59.156.113` and related subnet and gateway addressed in the command below are for exemplary purpose. **Please replace them with the IP addresses of your actual network**.

```
sudo ip addr del 198.59.156.113/24 dev eth1
sudo ovs-vsctl add-port br-ex eth1
sudo ifconfig eth1 up
sudo ip addr add 198.59.156.113/24 dev br-ex
sudo ifconfig br-ex up
sudo ip route add default via 198.59.156.1 dev br-ex
```

SCENARIO-3-SETUP-5: Verify that `br-ex` now has the original external IP address, and that the default route is on `br-ex`

```
opnfv@opnfv-os-controller:~/devstack$ ip a s br-ex
38: br-ex: <BROADCAST,UP,LOWER_UP> mtu 1430 qdisc noqueue state UNKNOWN group default
    link/ether 00:50:56:82:42:d1 brd ff:ff:ff:ff:ff:ff
    inet 198.59.156.113/24 brd 198.59.156.255 scope global br-ex
       valid_lft forever preferred_lft forever
    inet6 fe80::543e:28ff:fe70:4426/64 scope link
       valid_lft forever preferred_lft forever
opnfv@opnfv-os-controller:~/devstack$
opnfv@opnfv-os-controller:~/devstack$ ip route
default via 198.59.156.1 dev br-ex
192.168.0.0/24 dev eth0  proto kernel  scope link  src 192.168.0.10
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1
198.59.156.0/24 dev br-ex  proto kernel  scope link  src 198.59.156.113
```

Please note that the IP addresses above are exemplary purpose.

SCENARIO-3-SETUP-6: Create Neutron routers `ipv4-router` and `ipv6-router` which need to provide external connectivity.

```
neutron router-create ipv4-router
neutron router-create ipv6-router
```

SCENARIO-3-SETUP-7: Create an external network/subnet `ext-net` using the appropriate values based on the data-center physical network setup.

Please **NOTE** that if you use a different installer, i.e. NOT `devstack`, your installer may have already created an external network during installation. Under this circumstance, you may only need to create the subnet of `ext-net`. When you create the subnet, you must use the same name of external network that your installer creates.

```
# If you use a different installer and it has already created an external work,
# Please skip this command "net-create"
neutron net-create --router:external ext-net

# If you use a different installer and it has already created an external work,
# Change the name "ext-net" to match the name of external network that your installer has created
neutron subnet-create --disable-dhcp --allocation-pool start=198.59.156.251,end=198.59.156.254 --gate
```

SCENARIO-3-SETUP-8: Create Neutron networks `ipv4-int-network1` and `ipv6-int-network2`

```
neutron net-create ipv4-int-network1
neutron net-create ipv6-int-network2
```

SCENARIO-3-SETUP-9: Create IPv4 subnet `ipv4-int-subnet1` in the internal network `ipv4-int-network1`, and associate it to `ipv4-router`.

```
neutron subnet-create --name ipv4-int-subnet1 --dns-nameserver 8.8.8.8 ipv4-int-network1 20.0.0.0/24
neutron router-interface-add ipv4-router ipv4-int-subnet1
```

SCENARIO-3-SETUP-10: Associate the `ext-net` to the Neutron routers `ipv4-router` and `ipv6-router`.

```
# If you use a different installer and it has already created an external work,
# Change the name "ext-net" to match the name of external network that your installer has created
neutron router-gateway-set ipv4-router ext-net
neutron router-gateway-set ipv6-router ext-net
```

SCENARIO-3-SETUP-11: Create two subnets, one IPv4 subnet `ipv4-int-subnet2` and one IPv6 subnet `ipv6-int-subnet2` in `ipv6-int-network2`, and associate both subnets to `ipv6-router`

```
neutron subnet-create --name ipv4-int-subnet2 --dns-nameserver 8.8.8.8 ipv6-int-network2 10.0.0.0/24
neutron subnet-create --name ipv6-int-subnet2 --ip-version 6 --ipv6-ra-mode slaac --ipv6-address-mode
neutron router-interface-add ipv6-router ipv4-int-subnet2
neutron router-interface-add ipv6-router ipv6-int-subnet2
```

SCENARIO-3-SETUP-12: Create a keypair

```
nova keypair-add vRouterKey > ~/vRouterKey
```

SCENARIO-3-SETUP-13: Create ports for vRouter (with some specific MAC address - basically for automation - to know the IPv6 addresses that would be assigned to the port).

```
neutron port-create --name eth0-vRouter --mac-address fa:16:3e:11:11:11 ipv6-int-network2
neutron port-create --name eth1-vRouter --mac-address fa:16:3e:22:22:22 ipv4-int-network1
```

SCENARIO-3-SETUP-14: Create ports for VM1 and VM2.

```
neutron port-create --name eth0-VM1 --mac-address fa:16:3e:33:33:33 ipv4-int-network1
neutron port-create --name eth0-VM2 --mac-address fa:16:3e:44:44:44 ipv4-int-network1
```

SCENARIO-3-SETUP-15: Update `ipv6-router` with routing information to subnet `2001:db8:0:2::/64`

```
neutron router-update ipv6-router --routes type=dict list=true destination=2001:db8:0:2::/64,nexthop=
```

SCENARIO-3-SETUP-16: Boot Service VM (`vRouter`), VM1 and VM2

```
nova boot --image Fedora22 --flavor m1.small --user-data /opt/stack/opnfv_os_ipv6_poc/metadata.txt --
nova list
nova console-log vRouter #Please wait for some 10 to 15 minutes so that necessary packages (like radv
nova boot --image cirros-0.3.4-x86_64-uec --flavor m1.tiny --nic port-id=$(neutron port-list | grep -
nova boot --image cirros-0.3.4-x86_64-uec --flavor m1.tiny --nic port-id=$(neutron port-list | grep -
nova list # Verify that all the VMs are in ACTIVE state.
```

SCENARIO-3-SETUP-17: If all goes well, the IPv6 addresses assigned to the VMs would be as shown as follows:

```
vRouter eth0 interface would have the following IPv6 address: 2001:db8:0:1:f816:3eff:fe11:1111/64
vRouter eth1 interface would have the following IPv6 address: 2001:db8:0:2::1/64
VM1 would have the following IPv6 address: 2001:db8:0:2:f816:3eff:fe33:3333/64
VM2 would have the following IPv6 address: 2001:db8:0:2:f816:3eff:fe44:4444/64
```

SCENARIO-3-SETUP-18: Now we can `SSH` to VMs. You can execute the following command.

```
# 1. Create a floatingip and associate it with VM1, VM2 and vRouter (to the port id that is passed).
#    If you use a different installer and it has already created an external work,
#    Change the name "ext-net" to match the name of external network that your installer has created
neutron floatingip-create --port-id $(neutron port-list | grep -w eth0-VM1 | \
awk '{print $2}') ext-net
neutron floatingip-create --port-id $(neutron port-list | grep -w eth0-VM2 | \
awk '{print $2}') ext-net
neutron floatingip-create --port-id $(neutron port-list | grep -w eth1-vRouter | \
awk '{print $2}') ext-net
```

```
# 2. To know / display the floatingip associated with VM1, VM2 and vRouter.
neutron floatingip-list -F floating_ip_address -F port_id | grep $(neutron port-list | \
grep -w eth0-VM1 | awk '{print $2}') | awk '{print $2}'
neutron floatingip-list -F floating_ip_address -F port_id | grep $(neutron port-list | \
grep -w eth0-VM2 | awk '{print $2}') | awk '{print $2}'
neutron floatingip-list -F floating_ip_address -F port_id | grep $(neutron port-list | \
grep -w eth1-vRouter | awk '{print $2}') | awk '{print $2}'

# 3. To ssh to the vRouter, VM1 and VM2, user can execute the following command.
ssh -i ~/vRouterKey fedora@<floating-ip-of-vRouter>
ssh -i ~/vRouterKey cirros@<floating-ip-of-VM1>
ssh -i ~/vRouterKey cirros@<floating-ip-of-VM2>
```

NETWORK TOPOLOGY AFTER SETUP

5.1 Post-Install Network Topology

The network topology after setting up service VM as IPv6 vRouter is shown as follows Fig. 5.1:

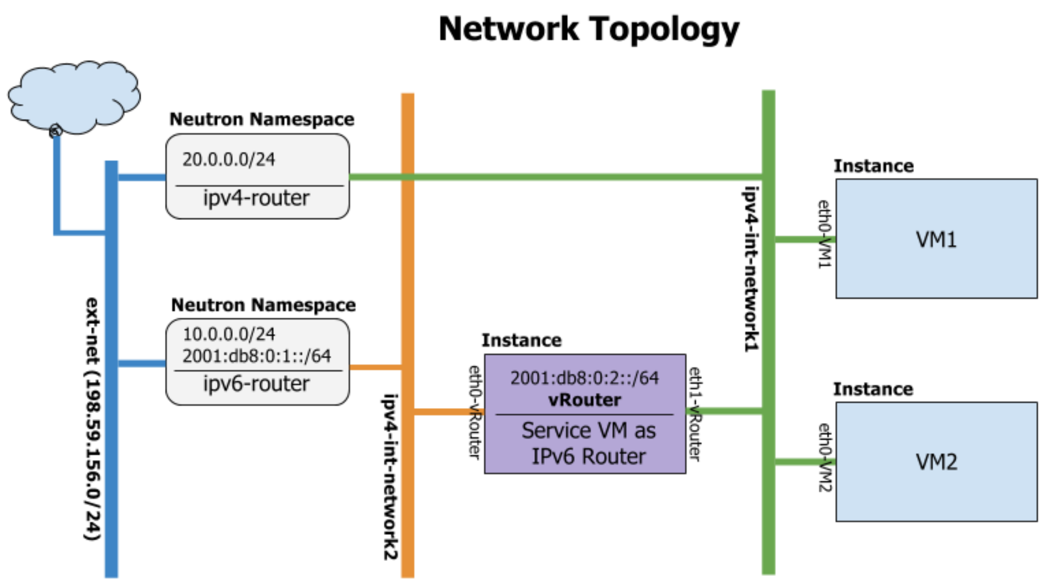

Fig. 5.1: Post-Install Network Topology

5.2 Sample Network Topology of this Setup through Horizon UI

The sample network topology of the setup will be shown in Horizon UI as follows Fig. 5.2:

5.3 Sample Network Topology of this Setup through ODL DLUX UI

If you set up either Scenario 2 or Scenario 3, the sample network topology of the setup will be shown in Open Daylight DLUX UI as follows Fig. 5.3:

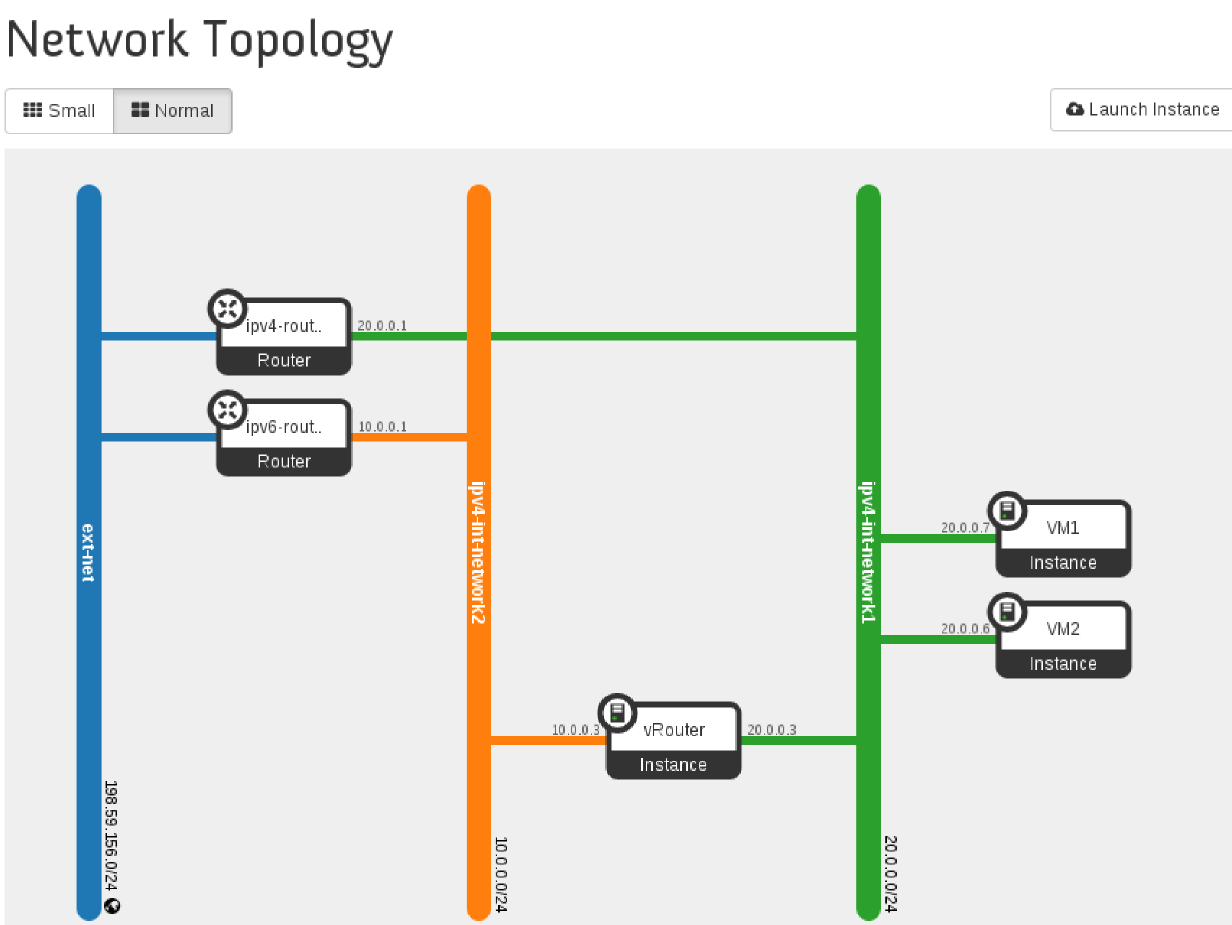

Fig. 5.2: Sample Network Topology in Horizon UI

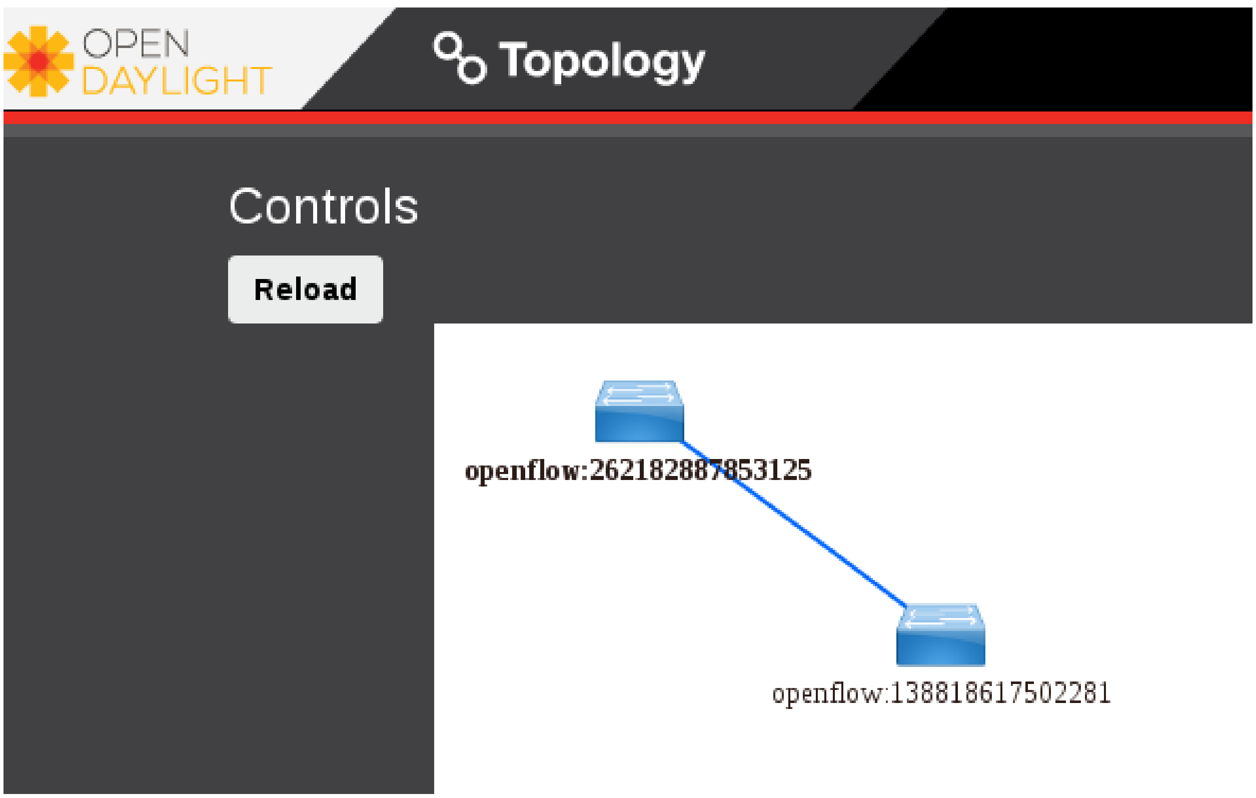

Fig. 5.3: Sample Network Topology in Open Daylight DLUX UI