IPv6 User Guide

Release draft (aa5a274)

OPNFV

August 12, 2016

Abstract

This IPv6 User Guide document provides the users with:

1. Instructions to set up a service VM as an IPv6 vRouter using OPNFV Colorado Release
2. Top-down gap analysis regarding IPv6 feature requirements with OpenStack Mitaka Official Release and Open Daylight Boron Official Release.

CHAPTER

ONE

IPV6 CONFIGURATION - SETTING UP A SERVICE VM AS AN IPV6 VROUTER

This section provides instructions to set up a service VM as an IPv6 vRouter using OPNFV Colorado Release installers. The environment may be pure OpenStack option or Open Daylight L2-only option. The deployment model may be HA or non-HA. The infrastructure may be bare metal or virtual environment.

For complete instructions and documentations of setting up service VM as an IPv6 vRouter using ANY method, please refer to:

1. IPv6 Configuration Guide (HTML): http://artifacts.opnfv.org/ipv6/docs/setupservicevm/index.html
2. IPv6 User Guide (HTML): http://artifacts.opnfv.org/ipv6/docs/gapanalysis/index.html

1.1 Pre-configuration Activities

The configuration will work in 2 environments:

1. OpenStack-only environment
2. OpenStack with Open Daylight L2-only environment

Depending on which installer will be used to deploy OPNFV, each environment may be deployed on bare metal or virtualized infrastructure. Each deployment may be HA or non-HA.

Refer to the previous installer configuration chapters, installations guide and release notes.

1.2 Setup Manual in OpenStack-Only Environment

If you intend to set up a service VM as an IPv6 vRouter in OpenStack-only environment of OPNFV Colorado Release, please **NOTE** that:

- 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 OpenStack-only environment.
- The hostnames, IP addresses, and username are for exemplary purpose in instructions. Please change as needed to fit your environment.
- The instructions apply to both deployment model of single controller node and HA (High Availability) deployment model where multiple controller nodes are used.

1.2.1 Install OPNFV and Preparation

OPNFV-NATIVE-INSTALL-1: To install OpenStack-only environment of OPNFV Colorado Release:

Apex Installer:

```
# HA, Virtual deployment in OpenStack-only environment
./opnfv-deploy -v -d /etc/opnfv-apex/os-nosdn-nofeature-ha.yaml \
-n /etc/opnfv-apex/network_setting.yaml

# HA, Bare Metal deployment in OpenStack-only environment
./opnfv-deploy -d /etc/opnfv-apex/os-nosdn-nofeature-ha.yaml \
-i <inventory file> -n /etc/opnfv-apex/network_setting.yaml

# Non-HA, Virtual deployment in OpenStack-only environment
./opnfv-deploy -v -d /etc/opnfv-apex/os-nosdn-nofeature-noha.yaml \
-n /etc/opnfv-apex/network_setting.yaml

# Non-HA, Bare Metal deployment in OpenStack-only environment
./opnfv-deploy -d /etc/opnfv-apex/os-nosdn-nofeature-noha.yaml \
-i <inventory file> -n /etc/opnfv-apex/network_setting.yaml

# Note:
#
# 1. Parameter ""-v" is mandatory for Virtual deployment
# 2. Parameter "-i <inventory file>" is mandatory for Bare Metal deployment
# 2.1 Refer to https://git.opnfv.org/cgit/apex/tree/config/inventory for examples of inventory file
# 3. You can use "-n /etc/opnfv-apex/network_setting_v6.yaml" for deployment in IPv6-only infrastruct
```

Compass Installer:

```
# HA deployment in OpenStack-only environment
export ISO_URL=file://$BUILD_DIRECTORY/compass.iso
export OS_VERSION=${COMPASS_OS_VERSION}
export OPENSTACK_VERSION=${COMPASS_OPENSTACK_VERSION}
export CONFDIR=$WORKSPACE/deploy/conf/vm_environment
./deploy.sh --dha $CONFDIR/os-nosdn-nofeature-ha.yml \
--network $CONFDIR/$NODE_NAME/network.yml

# Non-HA deployment in OpenStack-only environment
# Non-HA deployment is currently not supported by Compass installer
```

Fuel Installer:

```
# HA deployment in OpenStack-only environment
# Scenario Name: os-nosdn-nofeature-ha
# Scenario Configuration File: ha_heat_ceilometer_scenario.yaml
# You can use either Scenario Name or Scenario Configuration File Name in "-s" parameter
sudo ./deploy.sh -b <stack-config-uri> -l <lab-name> -p <pod-name> \
-s os-nosdn-nofeature-ha -i <iso-uri>

# Non-HA deployment in OpenStack-only environment
# Scenario Name: os-nosdn-nofeature-noha
# Scenario Configuration File: no-ha_heat_ceilometer_scenario.yaml
# You can use either Scenario Name or Scenario Configuration File Name in "-s" parameter
sudo ./deploy.sh -b <stack-config-uri> -l <lab-name> -p <pod-name> \
-s os-nosdn-nofeature-noha -i <iso-uri>

# Note:
#
```

```
# 1. Refer to http://git.opnfv.org/cgit/fuel/tree/deploy/scenario/scenario.yaml for scenarios
# 2. Refer to http://git.opnfv.org/cgit/fuel/tree/ci/README for description of
#    stack configuration directory structure
# 3. <stack-config-uri> is the base URI of stack configuration directory structure
# 3.1 Example: http://git.opnfv.org/cgit/fuel/tree/deploy/config
# 4. <lab-name> and <pod-name> must match the directory structure in stack configuration
# 4.1 Example of <lab-name>: -l devel-pipeline
# 4.2 Example of <pod-name>: -p elx
# 5. <iso-uri> could be local or remote ISO image of Fuel Installer
# 5.1 Example: http://artifacts.opnfv.org/fuel/colorado/opnfv-colorado.1.0.iso
#
# Please refer to Fuel Installer's documentation for further information and any update
```

Joid Installer:

```
# HA deployment in OpenStack-only environment
./deploy.sh -o mitaka -s nosdn -t ha -l default -f ipv6

# Non-HA deployment in OpenStack-only environment
./deploy.sh -o mitaka -s nosdn -t nonha -l default -f ipv6
```

Please **NOTE** that:

- You need to refer to **installer's documentation** for other necessary parameters applicable to your deployment.
- You need to refer to **Release Notes** and **installer's documentation** if there is any issue in installation.

OPNFV-NATIVE-INSTALL-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
```

1.2.2 Disable Security Groups in OpenStack ML2 Setup

OPNFV-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
```

OPNFV-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
```

OPNFV-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

1.2.3 Set Up Service VM as IPv6 vRouter

OPNFV-NATIVE-SETUP-1: Now we assume that OpenStack multi-node setup is up and running. We have to source the tenant credentials in OpenStack controller node in this step. Please **NOTE** that the method of sourcing

tenant credentials may vary depending on installers. For example:

Apex installer:

```
# On jump host, source the tenant credentials using /bin/opnfv-util provided by Apex installer
opnfv-util undercloud "source overcloudrc; keystone service-list"

# Alternatively, you can copy the file /home/stack/overcloudrc from the installer VM called "undercl
# to a location in controller node, for example, in the directory /opt, and do:
# source /opt/overcloudrc
```

Compass installer:

```
# source the tenant credentials using Compass installer of OPNFV
source /opt/admin-openrc.sh
```

Fuel installer:

```
# source the tenant credentials using Fuel installer of OPNFV
source /root/openrc
```

Joid installer:

```
# source the tenant credentials using Joid installer of OPNFV
source $HOME/joid_config/admin-openrc
```

devstack:

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

Please refer to relevant documentation of installers if you encounter any issue.

OPNFV-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-Cloud-Base-22-20150521.x86_64.qcow2
```

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

```
glance image-create --name 'Fedora22' --disk-format qcow2 --container-format bare \
--file ./Fedora-Cloud-Base-22-20150521.x86_64.qcow2
```

OPNFV-NATIVE-SETUP-4: This step is Informational. OPNFV Installer has taken care of this step during deployment. You may refer to this step only if there is any issue, or if you are using other installers.

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 refer to `OS-NATIVE-SETUP-4` and `OS-NATIVE-SETUP-5` in our more complete instruction.

OPNFV-NATIVE-SETUP-5: Create Neutron routers `ipv4-router` and `ipv6-router` which need to provide external connectivity.

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

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

Please **NOTE** that you may only need to create the subnet of `ext-net` because OPNFV installers should have created an external network during installation. You must use the same name of external network that installer creates when you create the subnet. For example:

- **Apex** installer: `external`
- **Compass** installer: `ext-net`
- **Fuel** installer: `admin_floating_net`
- **Joid** installer: `ext-net`

Please refer to the documentation of installers if there is any issue

```
# This is needed only if installer does not create an external work
# Otherwise, skip this command "net-create"
neutron net-create --router:external ext-net

# Note that the name "ext-net" may work for some installers such as Compass and Joid
# Change the name "ext-net" to match the name of external network that an installer creates
neutron subnet-create --disable-dhcp --allocation-pool start=198.59.156.251,\
end=198.59.156.254 --gateway 198.59.156.1 ext-net 198.59.156.0/24
```

OPNFV-NATIVE-SETUP-7: 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
```

OPNFV-NATIVE-SETUP-8: 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
```

OPNFV-NATIVE-SETUP-9: Associate the `ext-net` to the Neutron routers `ipv4-router` and `ipv6-router`.

```
# Note that the name "ext-net" may work for some installers such as Compass and Joid
# Change the name "ext-net" to match the name of external network that an installer creates
neutron router-gateway-set ipv4-router ext-net
neutron router-gateway-set ipv6-router ext-net
```

OPNFV-NATIVE-SETUP-10: 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 slaac ipv6-int-network2 2001:db8:0:1::/64

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

OPNFV-NATIVE-SETUP-11: Create a keypair

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

OPNFV-NATIVE-SETUP-12: 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
```

OPNFV-NATIVE-SETUP-13: 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
```

OPNFV-NATIVE-SETUP-14: 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=2001:db8:0:1:f816:3eff:fe11:1111
```

OPNFV-NATIVE-SETUP-15: Boot Service VM (`vRouter`), VM1 and VM2

```
nova boot --image Fedora22 --flavor m1.small \
--user-data /opt/stack/opnfv_os_ipv6_poc/metadata.txt \
--availability-zone nova:opnfv-os-compute \
--nic port-id=$(neutron port-list | grep -w eth0-vRouter | awk '{print $2}') \
--nic port-id=$(neutron port-list | grep -w eth1-vRouter | awk '{print $2}') \
--key-name vRouterKey vRouter

nova list

# Please wait for some 10 to 15 minutes so that necessary packages (like radvd)
# are installed and vRouter is up.
nova console-log vRouter

nova boot --image cirros-0.3.4-x86_64-uec --flavor m1.tiny \
--user-data /opt/stack/opnfv_os_ipv6_poc/set_mtu.sh \
--availability-zone nova:opnfv-os-controller \
--nic port-id=$(neutron port-list | grep -w eth0-VM1 | awk '{print $2}') \
--key-name vRouterKey VM1

nova boot --image cirros-0.3.4-x86_64-uec --flavor m1.tiny
--user-data /opt/stack/opnfv_os_ipv6_poc/set_mtu.sh \
--availability-zone nova:opnfv-os-compute \
--nic port-id=$(neutron port-list | grep -w eth0-VM2 | awk '{print $2}') \
--key-name vRouterKey VM2

nova list # Verify that all the VMs are in ACTIVE state.
```

OPNFV-NATIVE-SETUP-16: 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
```

OPNFV-NATIVE-SETUP-17: 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).
#    Note that the name "ext-net" may work for some installers such as Compass and Joid
#    Change the name "ext-net" to match the name of external network that an installer creates
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>
```

1.3 Setup Manual in OpenStack with Open Daylight L2-Only Environment

If you intend to set up a service VM as an IPv6 vRouter in an environment of OpenStack and Open Daylight L2-only of OPNFV Colorado Release, please **NOTE** that:

- We **SHOULD** use the `odl-ovsdb-openstack` version of Open Daylight Boron in OPNFV Colorado Release. Please refer to our Gap Analysis for more information.
- The hostnames, IP addresses, and username are for exemplary purpose in instructions. Please change as needed to fit your environment.
- The instructions apply to both deployment model of single controller node and HA (High Availability) deployment model where multiple controller nodes are used.
- However, in case of HA, 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**.

1.3.1 Install OPNFV and Preparation

OPNFV-INSTALL-1: To install OpenStack with Open Daylight L2-only environment of OPNFV Colorado Release:

Apex Installer:

```
# HA, Virtual deployment in OpenStack with Open Daylight L2-only environment
./opnfv-deploy -v -d /etc/opnfv-apex/os-odl_l2-nofeature-ha.yaml \
-n /etc/opnfv-apex/network_setting.yaml

# HA, Bare Metal deployment in OpenStack with Open Daylight L2-only environment
./opnfv-deploy -d /etc/opnfv-apex/os-odl_l2-nofeature-ha.yaml \
-i <inventory file> -n /etc/opnfv-apex/network_setting.yaml

# Non-HA deployment in OpenStack with Open Daylight L2-only environment
# There is no settings file provided by default for odl_l2 non-HA deployment
# You need to copy /etc/opnfv-apex/os-odl_l2-nofeature-ha.yaml to another file
# e.g. /etc/opnfv-apex/os-odl_l2-nofeature-noha.yaml
# and change the "ha_enabled" parameter to be "false", i.e.: "ha_enabled: false", and:
```

```
# - For Non-HA, Virtual deployment
./opnfv-deploy -v -d /etc/opnfv-apex/os-odl_l2-nofeature-noha.yaml \
-n /etc/opnfv-apex/network_setting.yaml

# - For Non-HA, Bare Metal deployment
./opnfv-deploy -d /etc/opnfv-apex/os-odl_l2-nofeature-noha.yaml \
-i <inventory file> -n /etc/opnfv-apex/network_setting.yaml

# Note:
#
# 1. Parameter ""-v" is mandatory for Virtual deployment
# 2. Parameter "-i <inventory file>" is mandatory for Bare Metal deployment
# 2.1 Refer to https://git.opnfv.org/cgit/apex/tree/config/inventory for examples of inventory file
# 3. You can use "-n /etc/opnfv-apex/network_setting_v6.yaml" for deployment in IPv6-only infrastruct
```

Compass Installer:

```
# HA deployment in OpenStack with Open Daylight L2-only environment
export ISO_URL=file://$BUILD_DIRECTORY/compass.iso
export OS_VERSION=${COMPASS_OS_VERSION}
export OPENSTACK_VERSION=${COMPASS_OPENSTACK_VERSION}
export CONFDIR=$WORKSPACE/deploy/conf/vm_environment
./deploy.sh --dha $CONFDIR/os-odl_l2-nofeature-ha.yml \
--network $CONFDIR/$NODE_NAME/network.yml

# Non-HA deployment in OpenStack with Open Daylight L2-only environment
# Non-HA deployment is currently not supported by Compass installer
```

Fuel Installer:

```
# HA deployment in OpenStack with Open Daylight L2-only environment
# Scenario Name: os-odl_l2-nofeature-ha
# Scenario Configuration File: ha_odl-l2_heat_ceilometer_scenario.yaml
# You can use either Scenario Name or Scenario Configuration File Name in "-s" parameter
sudo ./deploy.sh -b <stack-config-uri> -l <lab-name> -p <pod-name> \
-s os-odl_l2-nofeature-ha -i <iso-uri>

# Non-HA deployment in OpenStack with Open Daylight L2-only environment
# Scenario Name: os-odl_l2-nofeature-noha
# Scenario Configuration File: no-ha_odl-l2_heat_ceilometer_scenario.yaml
# You can use either Scenario Name or Scenario Configuration File Name in "-s" parameter
sudo ./deploy.sh -b <stack-config-uri> -l <lab-name> -p <pod-name> \
-s os-odl_l2-nofeature-noha -i <iso-uri>

# Note:
#
# 1. Refer to http://git.opnfv.org/cgit/fuel/tree/deploy/scenario/scenario.yaml for scenarios
# 2. Refer to http://git.opnfv.org/cgit/fuel/tree/ci/README for description of
#    stack configuration directory structure
# 3. <stack-config-uri> is the base URI of stack configuration directory structure
# 3.1 Example: http://git.opnfv.org/cgit/fuel/tree/deploy/config
# 4. <lab-name> and <pod-name> must match the directory structure in stack configuration
# 4.1 Example of <lab-name>: -l devel-pipeline
# 4.2 Example of <pod-name>: -p elx
# 5. <iso-uri> could be local or remote ISO image of Fuel Installer
# 5.1 Example: http://artifacts.opnfv.org/fuel/colorado/opnfv-colorado.1.0.iso
#
# Please refer to Fuel Installer's documentation for further information and any update
```

Joid Installer:

```
# HA deployment in OpenStack with Open Daylight L2-only environment
./deploy.sh -o mitaka -s odl -t ha -l default -f ipv6

# Non-HA deployment in OpenStack with Open Daylight L2-only environment
./deploy.sh -o mitaka -s odl -t nonha -l default -f ipv6
```

Please **NOTE** that:

- You need to refer to **installer's documentation** for other necessary parameters applicable to your deployment.
- You need to refer to **Release Notes** and **installer's documentation** if there is any issue in installation.

OPNFV-INSTALL-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
```

1.3.2 Disable Security Groups in OpenStack ML2 Setup

Please **NOTE** that although Security Groups feature has been disabled automatically through `local.conf` configuration file by some installers such as `devstack`, it is very likely that other installers such as `Apex`, `Compass`, `Fuel` or `Joid` will enable Security Groups feature after installation.

Please make sure that Security Groups are disabled in the setup

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

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

OPNFV-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
```

OPNFV-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

1.3.3 Source the Credentials in OpenStack Controller Node

SETUP-SVM-1: Login in OpenStack Controller Node. Start a new terminal, and change directory to where OpenStack is installed.

SETUP-SVM-2: We have to source the tenant credentials in this step. Please **NOTE** that the method of sourcing tenant credentials may vary depending on installers. For example:

Apex installer:

```
# On jump host, source the tenant credentials using /bin/opnfv-util provided by Apex installer
opnfv-util undercloud "source overcloudrc; keystone service-list"
```

```
# Alternatively, you can copy the file /home/stack/overcloudrc from the installer VM called "undercl
# to a location in controller node, for example, in the directory /opt, and do:
# source /opt/overcloudrc
```

Compass installer:

```
# source the tenant credentials using Compass installer of OPNFV
source /opt/admin-openrc.sh
```

Fuel installer:

```
# source the tenant credentials using Fuel installer of OPNFV
source /root/openrc
```

Joid installer:

```
# source the tenant credentials using Joid installer of OPNFV
source $HOME/joid_config/admin-openrc
```

devstack:

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

Please refer to relevant documentation of installers if you encounter any issue.

1.3.4 Informational Note: Move Public Network from Physical Network Interface to br-ex

SETUP-SVM-3: Move the physical interface (i.e. the public network interface) to `br-ex`

SETUP-SVM-4: Verify setup of `br-ex`

Those 2 steps are Informational. OPNFV Installer has taken care of those 2 steps during deployment. You may refer to this step only if there is any issue, or if you are using other installers.

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 refer to `SETUP-SVM-3` and `SETUP-SVM-4` in our more complete instruction.

1.3.5 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 you may only need to create the subnet of `ext-net` because OPNFV installers should have created an external network during installation. You must use the same name of external network that installer creates when you create the subnet. For example:

- **Apex** installer: `external`
- **Compass** installer: `ext-net`
- **Fuel** installer: `admin_floating_net`

- **Joid** installer: `ext-net`

Please refer to the documentation of installers if there is any issue

```
# This is needed only if installer does not create an external work
# Otherwise, skip this command "net-create"
neutron net-create --router:external ext-net

# Note that the name "ext-net" may work for some installers such as Compass and Joid
# Change the name "ext-net" to match the name of external network that an installer creates
neutron subnet-create --disable-dhcp --allocation-pool start=198.59.156.251,\
end=198.59.156.254 --gateway 198.59.156.1 ext-net 198.59.156.0/24
```

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`.

```
# Note that the name "ext-net" may work for some installers such as Compass and Joid
# Change the name "ext-net" to match the name of external network that an installer creates
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
```

1.3.6 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`

```
# Note that the name "ext-net" may work for some installers such as Compass and Joid
# Change the name "ext-net" to match the name of external network that an installer creates
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
```

1.3.7 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-Cloud-Base-22-20150521.x86_64.qcow2

glance image-create --name 'Fedora22' --disk-format qcow2 --container-format bare \
--file ./Fedora-Cloud-Base-22-20150521.x86_64.qcow2
```

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
```

1.3.8 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 \
--availability-zone nova:opnfv-os-compute \
--nic port-id=$(neutron port-list | grep -w eth0-vRouter | awk '{print $2}') \
--nic port-id=$(neutron port-list | grep -w eth1-vRouter | awk '{print $2}') \
--key-name vRouterKey vRouter
```

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: ############################################################
```

1.3.9 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 \
--user-data /opt/stack/opnfv_os_ipv6_poc/set_mtu.sh \
--availability-zone nova:opnfv-os-controller \
--nic port-id=$(neutron port-list | grep -w eth0-VM1 | awk '{print $2}') \
--key-name vRouterKey VM1
```

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 \
--user-data /opt/stack/opnfv_os_ipv6_poc/set_mtu.sh \
--availability-zone nova:opnfv-os-compute \
--nic port-id=$(neutron port-list | grep -w eth0-VM2 | awk '{print $2}') \
--key-name vRouterKey VM2
```

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

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

1.3.10 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
```

1.3.11 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).
#    Note that the name "ext-net" may work for some installers such as Compass and Joid
#    Change the name "ext-net" to match the name of external network that an installer creates
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`.

1.4 IPv6 Post Installation Procedures

Congratulations, you have completed the setup of using a service VM to act as an IPv6 vRouter. You have validated the setup based on the instruction in previous sections. If you want to further test your setup, you can `ping6` among `VM1`, `VM2`, `vRouter` and `ipv6-router`.

This setup allows further open innovation by any 3rd-party. For more instructions and documentations, please refer to:

1. IPv6 Configuration Guide (HTML): http://artifacts.opnfv.org/ipv6/docs/setupservicevm/index.html
2. IPv6 User Guide (HTML): http://artifacts.opnfv.org/ipv6/docs/gapanalysis/index.html

1.5 Automated post installation activities

Refer to the relevant testing guides, results, and release notes of Yardstick Project.