OVN4NFV

Release Latest

May 09, 2019

Contents

CHAPTER 1

OVN4NFV Development

Project Ovn4Nfv, https://wiki.opnfv.org/display/PROJ/Ovn4nfv

Editors Trinath Somanchi (NXP India)

Authors Prasad Gorja (NXP India) Trinath Somanchi (NXP India) Prakash Ramchandran (Dell)

Abstract This document provides an overview of networking-ovn,its utilization into NFV and outlines OpenStack official release.

1.1 OVN information

The original OVN project announcement can be found here:

- http://networkheresy.com/2015/01/13/ovn-bringing-native-virtual-networking-to-ovs/

The OVN architecture is described here:

- http://openvswitch.org/support/dist-docs/ovn-architecture.7.html

Here are two tutorials that help with learning different aspects of OVN:

- http://blog.spinhirne.com/p/blog-series.html#introToOVN
- http://docs.openvswitch.org/en/latest/tutorials/ovn-sandbox/

There is also an in depth tutorial on using OVN with OpenStack:

- http://docs.openvswitch.org/en/latest/tutorials/ovn-openstack/

OVN DB schemas and other man pages:

- http://openvswitch.org/support/dist-docs/ovn-nb.5.html
- http://openvswitch.org/support/dist-docs/ovn-sb.5.html
- http://openvswitch.org/support/dist-docs/ovn-nbctl.8.html
- http://openvswitch.org/support/dist-docs/ovn-sbctl.8.html

- http://openvswitch.org/support/dist-docs/ovn-northd.8.html
- http://openvswitch.org/support/dist-docs/ovn-controller.8.html
- http://openvswitch.org/support/dist-docs/ovn-controller-vtep.8.html

or find a full list of OVS and OVN man pages here:

- http://docs.openvswitch.org/en/latest/ref/

The openvswitch web page includes a list of presentations, some of which are about OVN:

- http://openvswitch.org/support/

Here are some direct links to past OVN presentations:

- OVN talk at OpenStack Summit in Boston, Spring 2017
- OVN talk at OpenStack Summit in Barcelona, Fall 2016
- OVN talk at OpenStack Summit in Austin, Spring 2016
- OVN Project Update at the OpenStack Summit in Tokyo, Fall 2015 - Slides - Video
- OVN at OpenStack Summit in Vancouver, Sping 2015 - Slides - Video
- OVS Conference 2015

These blog resources may also help with testing and understanding OVN:

- http://networkop.co.uk/blog/2016/11/27/ovn-part1/
- http://networkop.co.uk/blog/2016/12/10/ovn-part2/
- https://blog.russellbryant.net/2016/12/19/comparing-openstack-neutron-ml2ovs-and-ovn-control-plane/
- https://blog.russellbryant.net/2016/11/11/ovn-logical-flows-and-ovn-trace/
- https://blog.russellbryant.net/2016/09/29/ovs-2-6-and-the-first-release-of-ovn/
- http://galsagie.github.io/2015/11/23/ovn-l3-deepdive/
- http://blog.russellbryant.net/2015/10/22/openstack-security-groups-using-ovn-acls/
- http://galsagie.github.io/sdn/openstack/ovs/2015/05/30/ovn-deep-dive/
- http://blog.russellbryant.net/2015/05/14/an-ez-bake-ovn-for-openstack/
- http://galsagie.github.io/sdn/openstack/ovs/2015/04/26/ovn-containers/
- http://blog.russellbryant.net/2015/04/21/ovn-and-openstack-status-2015-04-21/
- http://blog.russellbryant.net/2015/04/08/ovn-and-openstack-integration-development-update/

1.1.1 Install & Configuration

The `networking-ovn` repository includes integration with DevStack that enables creation of a simple Open Virtual Network (OVN) development and test environment. This document discusses what is required for manual installation or integration into a production OpenStack deployment tool of conventional architectures that include the following types of nodes:

- Controller - Runs OpenStack control plane services such as REST APIs and databases.
- Network - Runs the layer-2, layer-3 (routing), DHCP, and metadata agents for the Networking service. Some agents optional. Usually provides connectivity between provider (public) and project (private) networks via NAT and floating IP addresses.

Note: Some tools deploy these services on controller nodes.

- Compute - Runs the hypervisor and layer-2 agent for the Networking service.

Packaging

Open vSwitch (OVS) includes OVN beginning with version 2.5 and considers it experimental. The Networking service integration for OVN uses an independent package, typically `networking-ovn`.

Building OVS from source automatically installs OVN. For deployment tools using distribution packages, the `openvswitch-ovn` package for RHEL/CentOS and compatible distributions automatically installs `openvswitch` as a dependency. Ubuntu/Debian includes `ovn-central`, `ovn-host`, `ovn-docker`, and `ovn-common` packages that pull in the appropriate Open vSwitch dependencies as needed.

A `python-networking-ovn` RPM may be obtained for Fedora or CentOS from the RDO project. A package based on the `master` branch of `networking-ovn` can be found at https://trunk.rdoproject.org/.

Fedora and CentOS RPM builds of OVS and OVN from the `master` branch of `ovs` can be found in this COPR repository: https://copr.fedorainfracloud.org/coprs/leifmadsen/ovs-master/.

Controller nodes

Each controller node runs the OVS service (including dependent services such as `ovsdb-server`) and the `ovn-northd` service. However, only a single instance of the `ovsdb-server` and `ovn-northd` services can operate in a deployment. However, deployment tools can implement active/passive high-availability using a management tool that monitors service health and automatically starts these services on another node after failure of the primary node. See the faq for more information.

1. Install the `openvswitch-ovn` and `networking-ovn` packages.
2. Start the OVS service. The central OVS service starts the `ovsdb-server` service that manages OVN databases.

   Using the *systemd* unit:

   ```
   # systemctl start openvswitch
   ```

   Using the `ovs-ctl` script:

   ```
   # /usr/share/openvswitch/scripts/ovs-ctl start  --system-id="random"
   ```

3. Configure the `ovsdb-server` component. By default, the `ovsdb-server` service only permits local access to databases via Unix socket. However, OVN services on compute nodes require access to these databases.
   - Permit remote database access.

     ```
     # ovs-appctl -t ovsdb-server ovsdb-server/add-remote ptcp:6640:IP_ADDRESS
     ```

     Replace `IP_ADDRESS` with the IP address of the management network interface on the controller node.

     Note: Permit remote access to TCP port 6640 on any host firewall.

4. Start the `ovn-northd` service.

   Using the *systemd* unit:

```
# systemctl start ovn-northd
```

Using the `ovn-ctl` script:

```
# /usr/share/openvswitch/scripts/ovn-ctl start_northd
```

Options for *start_northd*:

```
# /usr/share/openvswitch/scripts/ovn-ctl start_northd --help
# ...
# DB_NB_SOCK="/usr/local/etc/openvswitch/nb_db.sock"
# DB_NB_PID="/usr/local/etc/openvswitch/ovnnb_db.pid"
# DB_SB_SOCK="usr/local/etc/openvswitch/sb_db.sock"
# DB_SB_PID="/usr/local/etc/openvswitch/ovnsb_db.pid"
# ...
```

5. Configure the Networking server component. The Networking service implements OVN as an ML2 driver. Edit the `/etc/neutron/neutron.conf` file:

   - Enable the ML2 core plug-in.

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

   - Enable the OVN layer-3 service.

   ```
   [DEFAULT]
   ...
   service_plugins = networking_ovn.l3.l3_ovn.OVNL3RouterPlugin
   ```

6. Configure the ML2 plug-in. Edit the `/etc/neutron/plugins/ml2/ml2_conf.ini` file:

   - Configure the OVN mechanism driver, network type drivers, self-service (tenant) network types, and enable the port security extension.

   ```
   [ml2]
   ...
   mechanism_drivers = ovn
   type_drivers = local,flat,vlan,geneve
   tenant_network_types = geneve
   extension_drivers = port_security
   overlay_ip_version = 4
   ```

   Note: To enable VLAN self-service networks, add `vlan` to the `tenant_network_types` option. The first network type in the list becomes the default self-service network type.

   To use IPv6 for all overlay (tunnel) network endpoints, set the `overlay_ip_version` option to `6`.

   - Configure the Geneve ID range and maximum header size. The IP version overhead (20 bytes for IPv4 (default) or 40 bytes for IPv6) is added to the maximum header size based on the ML2 `overlay_ip_version` option.

   ```
   [ml2_type_geneve]
   ...
   ```

(continues on next page)

(continued from previous page)

```
vni_ranges = 1:65536
max_header_size = 38
```

Note: The Networking service uses the `vni_ranges` option to allocate network segments. However, OVN ignores the actual values. Thus, the ID range only determines the quantity of Geneve networks in the environment. For example, a range of `5001:6000` defines a maximum of 1000 Geneve networks.

- Optionally, enable support for VLAN provider and self-service networks on one or more physical networks. If you specify only the physical network, only administrative (privileged) users can manage VLAN networks. Additionally specifying a VLAN ID range for a physical network enables regular (non-privileged) users to manage VLAN networks. The Networking service allocates the VLAN ID for each self-service network using the VLAN ID range for the physical network.

```
[ml2_type_vlan]
...
network_vlan_ranges = PHYSICAL_NETWORK:MIN_VLAN_ID:MAX_VLAN_ID
```

Replace `PHYSICAL_NETWORK` with the physical network name and optionally define the minimum and maximum VLAN IDs. Use a comma to separate each physical network.

For example, to enable support for administrative VLAN networks on the `physnet1` network and self-service VLAN networks on the `physnet2` network using VLAN IDs 1001 to 2000:

```
network_vlan_ranges = physnet1,physnet2:1001:2000
```

- Enable security groups.

```
[securitygroup]
...
enable_security_group = true
```

Note: The `firewall_driver` option under `[securitygroup]` is ignored since the OVN ML2 driver itself handles security groups.

- Configure OVS database access and L3 scheduler

```
[ovn]
...
ovn_nb_connection = tcp:IP_ADDRESS:6641
ovn_sb_connection = tcp:IP_ADDRESS:6642
ovn_l3_scheduler = OVN_L3_SCHEDULER
```

Note: Replace `IP_ADDRESS` with the IP address of the controller node that runs the `ovsdb-server` service. Replace `OVN_L3_SCHEDULER` with `leastloaded` if you want the scheduler to select a compute node with the least number of gateway ports or `chance` if you want the scheduler to randomly select a compute node from the available list of compute nodes.

7. Start the `neutron-server` service.

Network nodes

Deployments using OVN native layer-3 and DHCP services do not require conventional network nodes because connectivity to external networks (including VTEP gateways) and routing occurs on compute nodes.

Compute nodes

Each compute node runs the OVS and `ovn-controller` services. The `ovn-controller` service replaces the conventional OVS layer-2 agent.

1. Install the `openvswitch-ovn` and `networking-ovn` packages.
2. Start the OVS service.

   Using the *systemd* unit:

   ```
   # systemctl start openvswitch
   ```

   Using the `ovs-ctl` script:

   ```
   # /usr/share/openvswitch/scripts/ovs-ctl start --system-id="random"
   ```

3. Configure the OVS service.
   - Use OVS databases on the controller node.

     ```
     # ovs-vsctl set open . external-ids:ovn-remote=tcp:IP_ADDRESS:6642
     ```

     Replace `IP_ADDRESS` with the IP address of the controller node that runs the `ovsdb-server` service.
   - Enable one or more overlay network protocols. At a minimum, OVN requires enabling the `geneve` protocol. Deployments using VTEP gateways should also enable the `vxlan` protocol.

     ```
     # ovs-vsctl set open . external-ids:ovn-encap-type=geneve,vxlan
     ```

     Note: Deployments without VTEP gateways can safely enable both protocols.
   - Configure the overlay network local endpoint IP address.

     ```
     # ovs-vsctl set open . external-ids:ovn-encap-ip=IP_ADDRESS
     ```

     Replace `IP_ADDRESS` with the IP address of the overlay network interface on the compute node.
4. Start the `ovn-controller` service.

   Using the *systemd* unit:

   ```
   # systemctl start ovn-controller
   ```

   Using the `ovn-ctl` script:

   ```
   # /usr/share/openvswitch/scripts/ovn-ctl start_controller
   ```

Verify operation

1. Each compute node should contain an `ovn-controller` instance.

```
# ovn-sbctl show
  <output>
```

1.2 OVN4NFV Project

OVN complements the existing capabilities of OVS to add native support for virtual network abstractions such as virtual L2 & L3 overlays, L3 routing and security groups. Instead of treating ovsdb and Open Flow actions separately, OVN provides simpler interface for managing virtual networks.

Besides the simpler interface, OVN takes care of transforming simple flow rules of virtual network to complex Open Flow rules on the Open vSwitches involved. The Openstack project networking-ovn implements the neutron api using OVN. As part of ovn4nfv project we would like to enable OVN along with the openstack neutron plugin networking-ovn as a deployable network control component in the OPNFV build. This would make it easier to manage virtual networks and push more of network intelligence to the edge onto the compute nodes. Since OVN has inherent support for containers, this would allow OPNFV to orchestrate container VNFs.

Further this will make the controller architecture much more simpler and scalable by placing the controller (ovn-controller) next to the Open vSwitch.

OVN for OPNFV at OPNFV Design Summit, Nov, 2015

- https://wiki.opnfv.org/_media/events/ovn-opnfv-summit2015.pdf

1.3 OVN-SFC POC Details

1.3.1 Purpose

The purpose of this Proof-of-concept is to showcase Service Function Chaining with OVN.

1.3.2 Scope

The Scope of this document is to describe SFC using OVN and discuss installation and configuration of OVN to instantiate a forwarding path.

1.3.3 Steps

1. Install CentOS7 minimal install:

- Make sure to enable network interface.
- Just create a root password. Don't create any users

2. Create user:

Below are the instructions to create user - stack, for use with Devstack.

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

3. Install git

- $ sudo yum install git -y

4. clone Devstack and Networking-ovn

- $ git clone http://git.openstack.org/openstack-dev/devstack.git
- $ git clone http://git.openstack.org/openstack/networking-ovn.git
- $ cd devstack
- $ cp ../networking-ovn/devstack/local.conf.sample local.conf

5. Edit the local.conf file:

- Add (uncomment and edited)
  - OVN_REPO=https://github.com/doonhammer/ovs
  - OVN_BRANCH=sfc.v30
- Uncomment the below line
  - OVN_BUILD_MODULES=False

We use forked/modifed OVS for SFC usecase from John McDowall.

6. Devstack Preliminaries:

- $ ./stack.sh
- $. ~/devstack/openrc admin
- $ openstack keypair create demo > ~/id_rsa_demo
- $ chmod 600 ~/id_rsa_demo
- **$ for group in $(openstack security group list -f value -c ID);**

  do openstack security group rule create –ingress –ethertype IPv4 –dst-port 22 –protocol tcp $group;
  openstack security group rule create –ingress –ethertype IPv4 – protocol ICMP $group;

  done
- $ IMAGE_ID=$(openstack image list -f value -c ID)

10. Create Neutron network and subnet

- $ openstack network create –project admin –provider-network-type geneve n1
- $ openstack subnet create –subnet-range 10.1.1.0/24 –network n1 n1subnet

10. Spawn VMs

- Create 5 VMs, 3 VMs to act as communication end-points (a,b, and c) and two VMs to act as VNFs (vnf1 & vnf2).
- The 2 VNF VMs are created with two NICs to act as ingress and egress ports (Optional)
- Created two SFCs: - SFC1: any traffic from VM a to VM b will go through vnf1 - SFC1: any traffic from VM a to VM c will go through vnf2 then vnf1

A. SFC with OVN - Scenario 1:

1. create VMs and VNFs:

- $ openstack server create –nic net-id=n1,v4-fixed-ip=10.1.1.5 –flavor m1.nano –image $IMAGE_ID –key-name demo a
- $ openstack server create –nic net-id=n1,v4-fixed-ip=10.1.1.6 –flavor m1.nano –image $IMAGE_ID –key-name demo b
- $ openstack server create –nic net-id=n1,v4-fixed-ip=10.1.1.10 –nic net-id=n1,v4-fixed-ip=10.1.1.11 –flavor m1.nano –image $IMAGE_ID –key-name demo vnf1
- $ openstack port set –name ap $(openstack port list –server a -f value -c ID)
- $ openstack port set –name bp $(openstack port list –server b -f value -c ID)
- $ AP_MAC=$(openstack port show -f value -c mac_address ap)
- $ BP_MAC=$(openstack port show -f value -c mac_address bp)
- $ openstack port set –name vnf1-pin $(openstack port list –server vnf1 –mac-address fa:16:3e:a0:e9:70 -f value -c ID)
- $ openstack port set –name vnf1-pout $(openstack port list –server vnf1 –mac-address fa:16:3e:ae:0c:36 -f value -c ID)
- $ f1_pin_MAC=$(openstack port show -f value -c mac_address vnf1-pin)
- $ f1_pout_MAC=$(openstack port show -f value -c mac_address vnf1-pout)

2. Create port-pairs, groups and chains

The switch and ports UUIDs below will different in each environment.

- n1 = f1de57df-04e3-456b-85c0-64fd869507ad
- vnf1-pin = 6ec5aa3d-8440-44c9-acf3-a18914ca9b0d
- vnf1-pout = 3f558a9d-295e-4417-9646-d46b59be97d8
- ap = 0438495b-7de4-4bbb-b787-dff82615b541
- bp = 1f004846-3f38-450d-8f4a-e5ed0f7228e6

- cp = 9a72cc76-4d8d-494c-a959-8d672149c0ea
- vnf2-pin = 6a32edc7-23d4-42ed-9cf8-c6e0009da01d
- vnf2-pout = 8553b6d2-1433-4ab4-ab69-704d318b09af

1. Configure the port pair vnf1-PP1

- $ ovn-nbctl lsp-pair-add n1 vnf1-pin vnf1-pout vnf1-PP1 (didn't work with names)
- $ ovn-nbctl lsp-pair-add f1de57df-04e3-456b-85c0-64fd869507ad 6ec5aa3d-8440-44c9-acf3-a18914ca9b0d 3f558a9d-295e-4417-9646-d46b59be97d8 vnf1-PP1

2. Configure the port chain PC1

- $ ovn-nbctl lsp-chain-add n1 PC1
- $ ovn-nbctl lsp-chain-add f1de57df-04e3-456b-85c0-64fd869507ad PC1

3. Configure the port pair group PG1 and add to port chain

- $ ovn-nbctl lsp-pair-group-add PC1 PG1

4. Add port pair to port chain

- $ ovn-nbctl lsp-pair-group-add-port-pair PG1 vnf1-PP1

5. Add port chain to port classifier PCC1

- $ lsp-chain-classifier-add SWITCH CHAIN PORT DIRECTION PATH [NAME] [MATCH]
- $ ovn-nbctl lsp-chain-classifier-add n1 PC1 bp 'entry-lport' 'bi-directional' PCC1 '';
- $ ovn-nbctl lsp-chain-classifier-add f1de57df-04e3-456b-85c0-64fd869507ad PC1 1f004846-3f38-450d-8f4a-e5ed0f7228e6 'entry-lport' 'bi-directional' PCC1 ''

3. Validating SFC

- $ ovn-trace n1 'inport == "ap" && eth.src == "$AP_MAC" && eth.dst == "$BP_MAC"'

B. SFC with OVN - Scenario 2:

1. Create VMs

- $ openstack server create –nic net-id=n1,v4-fixed-ip=10.1.1.7 –flavor m1.nano –image $IMAGE_ID –key-name demo c
- $ openstack server create –nic net-id=n1,v4-fixed-ip=10.1.1.20 –nic net-id=n1,v4-fixed-ip=10.1.1.21 –flavor m1.nano –image $IMAGE_ID –key-name demo vnf2
- $ openstack port set –name cp $(openstack port list –server c -f value -c ID)
- $ CP_MAC=$(openstack port show -f value -c mac_address cp)
- $ openstack port set –name vnf2-pin $(openstack port list –server vnf2 –mac-address fa:16:3e:ff:e5:76 -f value -c ID)
- $ openstack port set –name vnf2-pout $(openstack port list –server vnf2 –mac-address fa:16:3e:4c:a3:58 -f value -c ID)
- $ f2_pin_MAC=$(openstack port show -f value -c mac_address vnf2-pin)
- $ f2_pout_MAC=$(openstack port show -f value -c mac_address vnf2-pout)

2. Configure SFC

1. Configure the port pair vnf2-PP1

- $ ovn-nbctl lsp-pair-add n1 vnf2-pin vnf2-pout vnf2-PP1 (Didn't work with names)
- $ ovn-nbctl lsp-pair-add f1de57df-04e3-456b-85c0-64fd869507ad 6a32edc7-23d4-42ed-9cf8-c6e0009da01d 8553b6d2-1433-4ab4-ab69-704d318b09af vnf2-PP1

2. Configure the port chain PC2

- $ ovn-nbctl lsp-chain-add n1 PC2
- $ ovn-nbctl lsp-chain-add f1de57df-04e3-456b-85c0-64fd869507ad PC2

3. Configure the port pair group PG2 and add to port chain

- $ ovn-nbctl lsp-pair-group-add PC2 PG2
- $ ovn-nbctl lsp-pair-group-add PC2 PG3

4. Add port pair to port chain

- $ ovn-nbctl lsp-pair-group-add-port-pair PG2 vnf2-PP1
- $ ovn-nbctl lsp-pair-group-add-port-pair PG3 vnf1-PP1

4. Add port chain to port classifier PCC2

- $ ovn-nbctl lsp-chain-classifier-add n1 PC2 cp "entry-lport" "bi-directional" PCC2 ""
- $ ovn-nbctl lsp-chain-classifier-add f1de57df-04e3-456b-85c0-64fd869507ad PC2 9a72cc76-4d8d-494c-a959-8d672149c0ea "entry-lport" "bi-directional" PCC2 "";

3. Validate Scenario

- $ ovn-trace n1 'inport == "ap" && eth.src == "$AP_MAC" && eth.dst == "$CP_MAC"'

References:

1. http://docs.openvswitch.org/en/latest/tutorials/ovn-openstack/
2. https://gist.github.com/voyageur/a26943eced3324b302f1ffede45252bd
3. https://github.com/doonhammer/ovs

CHAPTER 2

OVN4NFV Configuration guide

Project OVN4NFV, https://wiki.opnfv.org/display/PROJ/Ovn4nfv

Editors Trinath Somanchi (NXP)

Authors Prasad Gorja (NXP) Trinath Somanchi (NXP) Prakash Ramchandran (Dell)

Abstract OVN4NFV Configuration Guide

2.1 Devstack Configuration

2.1.1 Devstack - localrc/local.conf

Sample DevStack local.conf.

```
#
# This sample file is intended to be used for your typical DevStack environment
# that's running all of OpenStack on a single host.  This can also be used as
# the first host of a multi-host test environment.
#
# No changes to this sample configuration are required for this to work.
#
DATABASE_PASSWORD=password
RABBIT_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=password
ADMIN_PASSWORD=password

# The DevStack plugin defaults to using the ovn branch from the official ovs
# repo.  You can optionally use a different one.  For example, you may want to
# use the latest patches in blp's ovn branch:
#OVN_REPO=https://github.com/blp/ovs-reviews.git
#OVN_BRANCH=ovn

```

(continues on next page)

(continued from previous page)

```
enable_plugin networking-ovn https://git.openstack.org/openstack/networking-ovn
enable_service ovn-northd
enable_service ovn-controller
enable_service networking-ovn-metadata-agent

# Use Neutron instead of nova-network
disable_service n-net
enable_service q-svc

# Disable Neutron agents not used with OVN.
disable_service q-agt
disable_service q-l3
disable_service q-dhcp
disable_service q-meta

# Horizon (the web UI) is enabled by default. You may want to disable
# it here to speed up DevStack a bit.
enable_service horizon
#disable_service horizon

# Cinder (OpenStack Block Storage) is disabled by default to speed up
# DevStack a bit. You may enable it here if you would like to use it.
disable_service cinder c-sch c-api c-vol
#enable_service cinder c-sch c-api c-vol

# How to connect to ovsdb-server hosting the OVN NB database.
#OVN_NB_REMOTE=tcp:$SERVICE_HOST:6641

# How to connect to ovsdb-server hosting the OVN SB database.
#OVN_SB_REMOTE=tcp:$SERVICE_HOST:6642

# A UUID to uniquely identify this system.  If one is not specified, a random
# one will be generated and saved in the file 'ovn-uuid' for re-use in future
# DevStack runs.
#OVN_UUID=

# If using the OVN native layer-3 service, choose a router scheduler to
# manage the distribution of router gateways on hypervisors/chassis.
# Default value is leastloaded.
#OVN_L3_SCHEDULER=leastloaded

# Whether or not to build custom openvswitch kernel modules from the ovs git
# tree. This is enabled by default.  This is required unless your distro kernel
# includes ovs+conntrack support.  This support was first released in Linux 4.3,
# and will likely be backported by some distros.
#OVN_BUILD_MODULES=False

# Enable services, these services depend on neutron plugin.
#enable_plugin neutron https://git.openstack.org/openstack/neutron
#enable_service q-qos
#enable_service q-trunk

# Skydive
#enable_plugin skydive https://github.com/redhat-cip/skydive.git
#enable_service skydive-analyzer
#enable_service skydive-agent
```

(continues on next page)

(continued from previous page)

```
# If you want to enable a provider network instead of the default private
# network after your DevStack environment installation, you *must* set
# the Q_USE_PROVIDER_NETWORKING to True, and also give FIXED_RANGE,
# NETWORK_GATEWAY and ALLOCATION_POOL option to the correct value that can
# be used in your environment. Specifying Q_AGENT is needed to allow devstack
# to run various "ip link set" and "ovs-vsctl" commands for the provider
# network setup.
#Q_AGENT=openvswitch
#Q_USE_PROVIDER_NETWORKING=True
#PHYSICAL_NETWORK=providernet
#PROVIDER_NETWORK_TYPE=flat
#PUBLIC_INTERFACE=<public interface>
#OVS_PHYSICAL_BRIDGE=br-provider
#PROVIDER_SUBNET_NAME=provider-subnet
# use the following for IPv4
#IP_VERSION=4
#FIXED_RANGE=<CIDR for the Provider Network>
#NETWORK_GATEWAY=<Provider Network Gateway>
#ALLOCATION_POOL=<Provider Network Allocation Pool>
# use the following for IPv4+IPv6
#IP_VERSION=4+6
#FIXED_RANGE=<CIDR for the Provider Network>
#NETWORK_GATEWAY=<Provider Network Gateway>
#ALLOCATION_POOL=<Provider Network Allocation Pool>
# IPV6_PROVIDER_FIXED_RANGE=<v6 CDIR for the Provider Network>
# IPV6_PROVIDER_NETWORK_GATEWAY=<v6 Gateway for the Provider Network>

# If you wish to use the provider network for public access to the cloud,
# set the following
#Q_USE_PROVIDERNET_FOR_PUBLIC=True
#PUBLIC_NETWORK_NAME=<Provider network name>
#PUBLIC_NETWORK_GATEWAY=<Provider network gateway>
#PUBLIC_PHYSICAL_NETWORK=<Provider network name>
#IP_VERSION=4
#PUBLIC_SUBNET_NAME=<provider subnet name>
#Q_FLOATING_ALLOCATION_POOL=<Provider Network Allocation Pool>
#FLOATING_RANGE=<CIDR for the Provider Network>

# NOTE: DO NOT MOVE THESE SECTIONS FROM THE END OF THIS FILE
# IF YOU DO, THEY WON'T WORK!!!!!
#

# Enable Nova automatic host discovery for cell every 2 seconds
# Only needed in case of multinode devstack, as otherwise there will be issues
# when the 2nd compute node goes online.
discover_hosts_in_cells_interval=2
```

2.1.2 Neutron - metadata_agent.ini

The following configuration options in /etc/neutron/metadata_agent.ini are required when OVN is enabled in OpenStack neutron.

```
...
...
[ovs]
```

(continues on next page)

(continued from previous page)

```
#
# From networking_ovn.metadata.agent
#

# The connection string for the native OVSDB backend.
# Use tcp:IP:PORT for TCP connection.
# Use unix:FILE for unix domain socket connection. (string value)
#ovsdb_connection = unix:/usr/local/var/run/openvswitch/db.sock

# Timeout in seconds for the OVSDB connection transaction (integer value)
#ovsdb_connection_timeout = 180

[ovn]
ovn_sb_connection = tcp:<controller-ip>:<port>
```

2.1.3 Neutron - neutron.conf

The following configuration changes are required in /etc/neutron/neutron.conf

```
[DEFAULT]
service_plugins = networking_ovn.l3.l3_ovn.OVNL3RouterPlugin
```

2.1.4 Neutron - ml2_conf.ini

The following configuration changes are required in /etc/neutron/plugins/ml2/ml2_conf.ini

```
[ml2]
mechanism_drivers = ovn,logger

[ovn]
ovn_metadata_enabled = True
ovn_l3_scheduler = leastloaded
neutron_sync_mode = log
ovn_sb_connection = tcp:<controller-ip>:<port>
ovn_nb_connection = tcp:<controller-ip>:<port>
```

CHAPTER 3

OVN4NFV - Fraser Release Notes

Project OVN4NFV, https://wiki.opnfv.org/display/PROJ/Ovn4nfv

Editors Trinath Somanchi (NXP India)

Authors Prasad Gorja (NXP India) Trinath Somanchi (NXP India) Prakash Ramchandran (Dell) Aakash K T (IIIT Hyderabad, INDIA)

Abstract OVN4NFV Release Notes.

3.1 Abstract

This document compiles the release notes for OPNFV release when using Apex, Joid and Fuel as a deployment tool.

3.2 Important Notes

This notes provides release information for the use of Apex, Joid and Fuel as deployment tool for the Gambia release of OPNFV.

The goal of the Gambia release and this Apex and Fuel based deployment process is to establish a lab ready platform accelerating further development of the OPNFV infrastructure.

Carefully follow the installation-instructions.

3.3 Summary

For Gambia release, OVN4NFV is supported with APEX and FUEL installers.

This Gambia artifact provides Apex and Fuel as the deployment stage tool in the OPNFV CI pipeline including:

- Documentation built by Jenkins

  - overall OPNFV documentation
  - this document (release notes)
  - installation instructions
- Automated validation of the Fraser deployment.

3.4 Release Data

Project	ovn4nfv
Repo/tag	ovn4nfv/opnfv-7.0
Release designation	Gambia 7.0
Release date	November 09 2018
Purpose of the delivery	New Scenario PoC: OVN SFC Scenario documentation, OVN-CN-VM OVN support in Fuel installer.

3.4.1 Deliverables

Software Deliverables

- Apex based installation
- Fuel based installation

Documentation Deliverables

- Installation instructions
- Release notes (This document)
- User guide and Testing notes
- Scenario Documentation (K8S-OVN)
- Scenario Documentation (OVN-SFC)

3.5 References

For more information, please see:

3.5.1 OPNFV

1) OPNFV Home Page
2) OPNFV Documentation
3) OPNFV Software Downloads
4) OVN4NFV Project

3.5.2 OpenStack

1) networking-ovn Project
2) OVN with K8S
3) OVN Architecture

CHAPTER 4

OVN4NFV Installer Scenarios

Editors Trinath Somanchi (NXP India)

Authors Prasad Gorja (NXP India) Trinath Somanchi (NXP India) Prakash Ramchandran (Dell) Aakash K T (IIIT Hyderabad INDIA)

Abstract Installer scenario documenation. Currently Apex, Fuel and Joid support OVN no-feature and feature scenarios.

4.1 Abstract

This document outlines the notes for deploying Kubernetes with OVN as the SDN and a load balancer. JOID is used as the deployment tool.

4.2 Introduction

4.2.1 Juju OPNFV Infrastructure Deployer (JOID)

JOID as Juju OPNFV Infrastructure Deployer allows you to deploy different combinations of OpenStack release and SDN solution in HA or non-HA mode. For OpenStack, JOID supports Juno and Liberty. For SDN, it supports Openvswitch, OpenContrail, OpenDayLight, and ONOS. In addition to HA or non-HA mode, it also supports deploying from the latest development tree.

JOID heavily utilizes the technology developed in Juju and MAAS. Juju is a state-of-the-art, open source, universal model for service oriented architecture and service oriented deployments. Juju allows you to deploy, configure, manage, maintain, and scale cloud services quickly and efficiently on public clouds, as well as on physical servers, OpenStack, and containers. You can use Juju from the command line or through its powerful GUI. MAAS (Metal-As-A-Service) brings the dynamism of cloud computing to the world of physical provisioning and Ubuntu. Connect, commission and deploy physical servers in record time, re-allocate nodes between services dynamically, and keep them up to date; and in due course, retire them from use. In conjunction with the Juju service orchestration software,

MAAS will enable you to get the most out of your physical hardware and dynamically deploy complex services with ease and confidence.

For more info on Juju and MAAS, please visit https://jujucharms.com/ and http://maas.ubuntu.com.

4.3 Production grade container Orchestrator - Kubernetes (K8S)

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

This is a Kubernetes cluster that includes logging, monitoring, and operational knowledge. It is comprised of the following components and features:

- Kubernetes (automated deployment, operations, and scaling)

  TLS used for communication between nodes for security. A CNI plugin - in this scenario, we use OVN (Refer : https://github.com/openvswitch/ovn-kubernetes) A load balancer for HA kubernetes-master (Experimental) Optional Ingress Controller (on worker) Optional Dashboard addon (on master) including Heapster for cluster monitoring

- EasyRSA

  Performs the role of a certificate authority serving self signed certificates to the requesting units of the cluster.

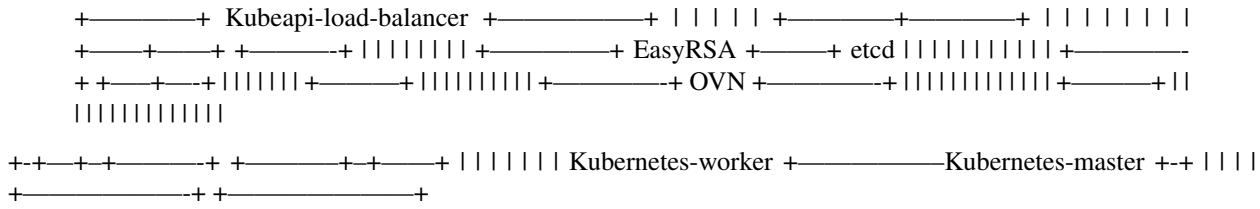
- Etcd (distributed key value store)

  Minimum Three node cluster for reliability.

- **Kubernetes deployment with Load Balancer**

```
./deploy.sh -m kubernetes -f lb -l custom -s ovn
```

4.4 Deployment Diagram



4.5 Using Kubernetes after Deployment

Once you have finished installing Kubernetes, you can use the following command to test the deployment.

To deploy 5 replicas of the microbot web application inside the Kubernetes cluster run the following command:

```
juju run-action kubernetes-worker/0 microbot replicas=5
```

This action performs the following steps:

It creates a deployment titled 'microbots' comprised of 5 replicas defined during the run of the action. It also creates a service named 'microbots' which binds an 'endpoint', using all 5 of the 'microbots' pods. Finally, it will create an ingress resource, which points at a xip.io domain to simulate a proper DNS service.

Running the packaged example

You can run a Juju action to create an example microbot web application:

```
$ juju run-action kubernetes-worker/0 microbot replicas=3
Action queued with id: db7cc72b-5f35-4a4d-877c-284c4b776eb8

$ juju show-action-output db7cc72b-5f35-4a4d-877c-284c4b776eb8
results:
  address: microbot.104.198.77.197.xip.io
  status: completed
  timing:
  completed: 2016-09-26 20:42:42 +0000 UTC
  enqueued: 2016-09-26 20:42:39 +0000 UTC
  started: 2016-09-26 20:42:41 +0000 UTC
```

Note: Your FQDN will be different and contain the address of the cloud instance. At this point, you can inspect the cluster to observe the workload coming online.

For deploying via juju, refer to : https://jujucharms.com/u/aakashkt/kubernetes-ovn/

4.6 Supported OVN scenarios

Name: joid-k8s-ovn-lb-noha Test Link: https://build.opnfv.org/ci/view/joid/job/joid-k8-ovn-lb-noha-baremetal-daily-master/

4.7 References

- *Juju Charm store <https://jujucharms.com/>*
- *Juju documents <https://jujucharms.com/docs/stable/getting-started>*
- *Canonical Distibuytion of Kubernetes <https://jujucharms.com/canonical-kubernetes/>*
- *Bare metal management (Metal-As-A-Service) <http://maas.io/get-started>*
- *MAAS API documents <http://maas.ubuntu.com/docs/>*
- *OPNFV JOID wiki <https://wiki.opnfv.org/joid>*
- *OPNFV JOID Get Started <https://wiki.opnfv.org/display/joid/JOID+Get+Started>*
- *Kubernetes Release artifacts <https://get.k8s.io/>*
- *Kubernetes documentation <https://kubernetes.io/>*

CHAPTER 5

OVN4NFV Testing Notes

Project OVN4NFV, https://wiki.opnfv.org/display/PROJ/Ovn4nfv

Editors Trinath Somanchi (NXP India)

Authors Prasad Gorja (NXP India) Trinath Somanchi (NXP India) Prakash Ramchandran (Dell)

Abstract OVN4NFV Testing Notes.

5.1 Testing with DevStack

This document describes how to test OpenStack with OVN using DevStack. We will start by describing how to test on a single host.

5.1.1 Single Node Test Environment

1. Create a test system.

It's best to use a throwaway dev system for running DevStack. Your best bet is to use either CentOS 7 or the latest Ubuntu LTS (16.04, Xenial).

2. Create the `stack` user.

```
$ git clone https://git.openstack.org/openstack-dev/devstack.git
$ sudo ./devstack/tools/create-stack-user.sh
```

3. Switch to the `stack` user and clone DevStack and networking-ovn.

```
$ sudo su - stack
$ git clone https://git.openstack.org/openstack-dev/devstack.git
$ git clone https://git.openstack.org/openstack/networking-ovn.git
```

4. Configure DevStack to use networking-ovn.

networking-ovn comes with a sample DevStack configuration file you can start with. For example, you may want to set some values for the various PASSWORD variables in that file so DevStack doesn't have to prompt you for them. Feel free to edit it if you'd like, but it should work as-is.

```
$ cd devstack
$ cp ../networking-ovn/devstack/local.conf.sample local.conf
```

5. Run DevStack.

This is going to take a while. It installs a bunch of packages, clones a bunch of git repos, and installs everything from these git repos.

```
$ ./stack.sh
```

Once DevStack completes successfully, you should see output that looks something like this:

```
This is your host IP address: 172.16.189.6
This is your host IPv6 address: ::1
Horizon is now available at http://172.16.189.6/dashboard
Keystone is serving at http://172.16.189.6/identity/
The default users are: admin and demo
The password: password
2017-03-09 15:10:54.117 | stack.sh completed in 2110 seconds.
```

5.1.2 Environment Variables

Once DevStack finishes successfully, we're ready to start interacting with OpenStack APIs. OpenStack provides a set of command line tools for interacting with these APIs. DevStack provides a file you can source to set up the right environment variables to make the OpenStack command line tools work.

```
$ . openrc
```

If you're curious what environment variables are set, they generally start with an OS prefix:

```
$ env | grep OS
OS_REGION_NAME=RegionOne
OS_IDENTITY_API_VERSION=2.0
OS_PASSWORD=password
OS_AUTH_URL=http://192.168.122.8:5000/v2.0
OS_USERNAME=demo
OS_TENANT_NAME=demo
OS_VOLUME_API_VERSION=2
OS_CACERT=/opt/stack/data/CA/int-ca/ca-chain.pem
OS_NO_CACHE=1
```

5.1.3 Default Network Configuration

By default, DevStack creates networks called `private` and `public`. Run the following command to see the existing networks:

```
$ openstack network list
+--------------------------------------+---------+--------------------------------------
↪----------------------------------------+
| ID                                   | Name    | Subnets                              ␣
↪                                      |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+---------+----------------------------------
↪--------------------------------------+
| 40080dad-0064-480a-b1b0-592ae51c1471 | private | 5ff81545-7939-4ae0-8365-
↪1658d45fa85c, da34f952-3bfc-45bb-b062-d2d973c1a751 |
| 7ec986dd-aae4-40b5-86cf-8668feeeab67 | public  | 60d0c146-a29b-4cd3-bd90-
↪3745603b1a4b, f010c309-09be-4af2-80d6-e6af9c78bae7 |
+--------------------------------------+---------+----------------------------------
↪--------------------------------------+
```

A Neutron network is implemented as an OVN logical switch. networking-ovn creates logical switches with a name in the format neutron-<network UUID>. We can use `ovn-nbctl` to list the configured logical switches and see that their names correlate with the output from `neutron net-list`:

```
$ ovn-nbctl ls-list
71206f5c-b0e6-49ce-b572-eb2e964b2c4e (neutron-40080dad-0064-480a-b1b0-592ae51c1471)
8d8270e7-fd51-416f-ae85-16565200b8a4 (neutron-7ec986dd-aae4-40b5-86cf-8668feeeab67)

$ ovn-nbctl get Logical_Switch neutron-40080dad-0064-480a-b1b0-592ae51c1471 external_
↪ids
{"neutron:network_name"=private}
```

5.1.4 Booting VMs

In this section we'll go through the steps to create two VMs that have a virtual NIC attached to the `private` Neutron network.

DevStack uses libvirt as the Nova backend by default. If KVM is available, it will be used. Otherwise, it will just run qemu emulated guests. This is perfectly fine for our testing, as we only need these VMs to be able to send and receive a small amount of traffic so performance is not very important.

1. Get the Network UUID.

Start by getting the UUID for the `private` network from the output of `neutron net-list` from earlier and save it off:

```
$ PRIVATE_NET_ID=40080dad-0064-480a-b1b0-592ae51c1471
```

2. Create an SSH keypair.

Next create an SSH keypair in Nova. Later, when we boot a VM, we'll ask that the public key be put in the VM so we can SSH into it.

```
$ openstack keypair create demo > id_rsa_demo
$ chmod 600 id_rsa_demo
```

3. Choose a flavor.

We need minimal resources for these test VMs, so the `m1.nano` flavor is sufficient.

```
$ openstack flavor list
+----+-----------+-------+------+-----------+-------+-----------+
| ID | Name      |   RAM | Disk | Ephemeral | VCPUs | Is Public |
+----+-----------+-------+------+-----------+-------+-----------+
| 1  | m1.tiny   |   512 |    1 |         0 |     1 | True      |
| 2  | m1.small  |  2048 |   20 |         0 |     1 | True      |
| 3  | m1.medium |  4096 |   40 |         0 |     2 | True      |
```

(continues on next page)

(continued from previous page)

```
| 4  | m1.large  |  8192 |   80 |         0 |     4 | True      |
| 42 | m1.nano   |    64 |    0 |         0 |     1 | True      |
| 5  | m1.xlarge | 16384 |  160 |         0 |     8 | True      |
| 84 | m1.micro  |   128 |    0 |         0 |     1 | True      |
| c1 | cirros256 |   256 |    0 |         0 |     1 | True      |
| d1 | ds512M    |   512 |    5 |         0 |     1 | True      |
| d2 | ds1G      |  1024 |   10 |         0 |     1 | True      |
| d3 | ds2G      |  2048 |   10 |         0 |     2 | True      |
| d4 | ds4G      |  4096 |   20 |         0 |     4 | True      |
+----+-----------+-------+------+-----------+-------+-----------+

$ FLAVOR_ID=42
```

4. Choose an image.

DevStack imports the CirrOS image by default, which is perfect for our testing. It's a very small test image.

```
$ openstack image list
+--------------------------------------+--------------------------+--------+
| ID                                   | Name                     | Status |
+--------------------------------------+--------------------------+--------+
| 849a8db2-3754-4cf6-9271-491fa4ff7195 | cirros-0.3.5-x86_64-disk | active |
+--------------------------------------+--------------------------+--------+

$ IMAGE_ID=849a8db2-3754-4cf6-9271-491fa4ff7195
```

5. Setup a security rule so that we can access the VMs we will boot up next.

By default, DevStack does not allow users to access VMs, to enable that, we will need to add a rule. We will allow both ICMP and SSH.

```
$ openstack security group rule create --ingress --ethertype IPv4 --dst-port 22 --
↪protocol tcp default
$ openstack security group rule create --ingress --ethertype IPv4 --protocol ICMP␣
↪default
$ openstack security group rule list
+--------------------------------------+-------------+-----------+------------+-------
↪-------------------------------+--------------------------------------+
| ID                                   | IP Protocol | IP Range  | Port Range |␣
↪Remote Security Group                | Security Group                       |
+--------------------------------------+-------------+-----------+------------+-------
↪-------------------------------+--------------------------------------+
...
| ade97198-db44-429e-9b30-24693d86d9b1 | tcp         | 0.0.0.0/0 | 22:22      | None ␣
↪                                | a47b14da-5607-404a-8de4-3a0f1ad3649c |
| d0861a98-f90e-4d1a-abfb-827b416bc2f6 | icmp        | 0.0.0.0/0 |            | None ␣
↪                                | a47b14da-5607-404a-8de4-3a0f1ad3649c |
...
+--------------------------------------+-------------+-----------+------------+-------
↪-------------------------------+--------------------------------------+

$ neutron security-group-rule-create --direction ingress --ethertype IPv4 --port-
↪range-min 22 --port-range-max 22 --protocol tcp default
$ neutron security-group-rule-create --direction ingress --ethertype IPv4 --protocol␣
↪ICMP default
$ neutron security-group-rule-list
+--------------------------------------+----------------+-----------+-----------+-----
↪----------+-----------------+
```

(continues on next page)

(continued from previous page)

```
| id                                   | security_group | direction | ethertype |␣
↪protocol/port | remote          |
+--------------------------------------+----------------+-----------+-----------+-----
↪----------+-----------------+
| 8b2edbe6-790e-40ef-af54-c7b64ced8240 | default        | ingress   | IPv4      | 22/
↪tcp        | any             |
| 5bee0179-807b-41d7-ab16-6de6ac051335 | default        | ingress   | IPv4      |␣
↪icmp          | any             |
...
+--------------------------------------+----------------+-----------+-----------+-----
↪----------+-----------------+
```

6. Boot some VMs.

Now we will boot two VMs. We'll name them `test1` and `test2`.

```
$ openstack server create --nic net-id=$PRIVATE_NET_ID --flavor $FLAVOR_ID --image
↪$IMAGE_ID --key-name demo test1
+------------------------------+--------------------------------------------------------
↪----------+
| Field                        | Value                                                  ␣
↪          |
+------------------------------+--------------------------------------------------------
↪----------+
| OS-DCF:diskConfig            | MANUAL                                                 ␣
↪          |
| OS-EXT-AZ:availability_zone  |                                                        ␣
↪          |
| OS-EXT-STS:power_state       | NOSTATE                                                ␣
↪          |
| OS-EXT-STS:task_state        | scheduling                                             ␣
↪          |
| OS-EXT-STS:vm_state          | building                                               ␣
↪          |
| OS-SRV-USG:launched_at       | None                                                   ␣
↪          |
| OS-SRV-USG:terminated_at     | None                                                   ␣
↪          |
| accessIPv4                   |                                                        ␣
↪          |
| accessIPv6                   |                                                        ␣
↪          |
| addresses                    |                                                        ␣
↪          |
| adminPass                    | BzAWWA6byGP6                                           ␣
↪          |
| config_drive                 |                                                        ␣
↪          |
| created                      | 2017-03-09T16:56:08Z                                   ␣
↪          |
| flavor                       | m1.nano (42)                                           ␣
↪          |
| hostId                       |                                                        ␣
↪          |
| id                           | d8b8084e-58ff-44f4-b029-a57e7ef6ba61                   ␣
↪          |
| image                        | cirros-0.3.5-x86_64-disk (849a8db2-3754-4cf6-9271-
↪491fa4ff7195) |
```

(continues on next page)

(continued from previous page)

```
| key_name                    | demo                                                   ␣
↪          |
| name                        | test1                                                  ␣
↪          |
| progress                    | 0                                                      ␣
↪          |
| project_id                  | b6522570f7344c06b1f24303abf3c479                       ␣
↪          |
| properties                  |                                                        ␣
↪          |
| security_groups             | name='default'                                         ␣
↪          |
| status                      | BUILD                                                  ␣
↪          |
| updated                     | 2017-03-09T16:56:08Z                                   ␣
↪          |
| user_id                     | c68f77f1d85e43eb9e5176380a68ac1f                       ␣
↪          |
| volumes_attached            |                                                        ␣
↪          |
+-----------------------------+-------------------------------------------------------
↪----------+

$ openstack server create --nic net-id=$PRIVATE_NET_ID --flavor $FLAVOR_ID --image
↪$IMAGE_ID --key-name demo test2
+-----------------------------+-------------------------------------------------------
↪----------+
| Field                       | Value                                                  ␣
↪          |
+-----------------------------+-------------------------------------------------------
↪----------+
| OS-DCF:diskConfig           | MANUAL                                                 ␣
↪          |
| OS-EXT-AZ:availability_zone |                                                        ␣
↪          |
| OS-EXT-STS:power_state      | NOSTATE                                                ␣
↪          |
| OS-EXT-STS:task_state       | scheduling                                             ␣
↪          |
| OS-EXT-STS:vm_state         | building                                               ␣
↪          |
| OS-SRV-USG:launched_at      | None                                                   ␣
↪          |
| OS-SRV-USG:terminated_at    | None                                                   ␣
↪          |
| accessIPv4                  |                                                        ␣
↪          |
| accessIPv6                  |                                                        ␣
↪          |
| addresses                   |                                                        ␣
↪          |
| adminPass                   | YB8dmt5v88JV                                           ␣
↪          |
| config_drive                |                                                        ␣
↪          |
| created                     | 2017-03-09T16:56:50Z                                   ␣
↪          |
```

(continues on next page)

(continued from previous page)

```
| flavor                       | m1.nano (42)                                                 ␣
↪          |
| hostId                       |                                                              ␣
↪          |
| id                           | 170d4f37-9299-4a08-b48b-2b90fce8e09b                         ␣
↪          |
| image                        | cirros-0.3.5-x86_64-disk (849a8db2-3754-4cf6-9271-
↪491fa4ff7195) |
| key_name                     | demo                                                         ␣
↪          |
| name                         | test2                                                        ␣
↪          |
| progress                     | 0                                                            ␣
↪          |
| project_id                   | b6522570f7344c06b1f24303abf3c479                             ␣
↪          |
| properties                   |                                                              ␣
↪          |
| security_groups              | name='default'                                               ␣
↪          |
| status                       | BUILD                                                        ␣
↪          |
| updated                      | 2017-03-09T16:56:51Z                                         ␣
↪          |
| user_id                      | c68f77f1d85e43eb9e5176380a68ac1f                             ␣
↪          |
| volumes_attached             |                                                              ␣
↪          |
+------------------------------+--------------------------------------------------------
↪----------+
```

Once both VMs have been started, they will have a status of `ACTIVE`:

```
$ openstack server list
+--------------------------------------+-------+--------+-----------------------------
↪---------------------------+--------------------------+
| ID                                   | Name  | Status | Networks                    ␣
↪                           | Image Name               |
+--------------------------------------+-------+--------+-----------------------------
↪---------------------------+--------------------------+
| 170d4f37-9299-4a08-b48b-2b90fce8e09b | test2 | ACTIVE |␣
↪private=fd5d:9d1b:457c:0:f816:3eff:fe24:49df, 10.0.0.3  | cirros-0.3.5-x86_64-disk |
| d8b8084e-58ff-44f4-b029-a57e7ef6ba61 | test1 | ACTIVE |␣
↪private=fd5d:9d1b:457c:0:f816:3eff:fe3f:953d, 10.0.0.10 | cirros-0.3.5-x86_64-disk |
+--------------------------------------+-------+--------+-----------------------------
↪---------------------------+--------------------------+
```

Our two VMs have addresses of `10.0.0.3` and `10.0.0.10`. If we list Neutron ports, there are two new ports with these addresses associated with them:

```
$ openstack port list
+--------------------------------------+------+-------------------+-------------------
↪-----------------------------------------------------------------------------+-
↪-------+
| ID                                   | Name | MAC Address       | Fixed IP␣
↪Addresses                                                                    ␣
↪       | Status |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+------+-------------------+-------------------
↪-------------------------------------------------------------------------------------+-
↪-------+
...
| 97c970b0-485d-47ec-868d-783c2f7acde3 |      | fa:16:3e:3f:95:3d | ip_address='10.0.
↪0.10', subnet_id='da34f952-3bfc-45bb-b062-d2d973c1a751'                             ␣
↪| ACTIVE |
|                                      |      |                   | ip_address=
↪'fd5d:9d1b:457c:0:f816:3eff:fe3f:953d', subnet_id='5ff81545-7939-4ae0-8365-
↪1658d45fa85c' |        |
| e003044d-334a-4de3-96d9-35b2d2280454 |      | fa:16:3e:24:49:df | ip_address='10.0.
↪0.3', subnet_id='da34f952-3bfc-45bb-b062-d2d973c1a751'                              ␣
↪| ACTIVE |
|                                      |      |                   | ip_address=
↪'fd5d:9d1b:457c:0:f816:3eff:fe24:49df', subnet_id='5ff81545-7939-4ae0-8365-
↪1658d45fa85c' |        |
...
+--------------------------------------+------+-------------------+-------------------
↪-------------------------------------------------------------------------------------+-
↪-------+

$ TEST1_PORT_ID=97c970b0-485d-47ec-868d-783c2f7acde3
$ TEST2_PORT_ID=e003044d-334a-4de3-96d9-35b2d2280454
```

Now we can look at OVN using `ovn-nbctl` to see the logical switch ports that were created for these two Neutron ports. The first part of the output is the OVN logical switch port UUID. The second part in parentheses is the logical switch port name. Neutron sets the logical switch port name equal to the Neutron port ID.

```
$ ovn-nbctl lsp-list neutron-$PRIVATE_NET_ID
...
fde1744b-e03b-46b7-b181-abddcbe60bf2 (97c970b0-485d-47ec-868d-783c2f7acde3)
7ce284a8-a48a-42f5-bf84-b2bca62cd0fe (e003044d-334a-4de3-96d9-35b2d2280454)
...
```

These two ports correspond to the two VMs we created.

5.1.5 VM Connectivity

We can connect to our VMs by associating a floating IP address from the public network.

```
$ openstack floating ip create --port $TEST1_PORT_ID public
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| created_at          | 2017-03-09T18:58:12Z                 |
| description         |                                      |
| fixed_ip_address    | 10.0.0.10                            |
| floating_ip_address | 172.24.4.8                           |
| floating_network_id | 7ec986dd-aae4-40b5-86cf-8668feeeab67 |
| id                  | 24ff0799-5a72-4a5b-abc0-58b301c9aee5 |
| name                | None                                 |
| port_id             | 97c970b0-485d-47ec-868d-783c2f7acde3 |
| project_id          | b6522570f7344c06b1f24303abf3c479     |
| revision_number     | 1                                    |
| router_id           | ee51adeb-0dd8-4da0-ab6f-7ce60e00e7b0 |
```

(continues on next page)

(continued from previous page)

```
| status               | DOWN                                 |
| updated_at           | 2017-03-09T18:58:12Z                 |
+----------------------+--------------------------------------+
```

Devstack does not wire up the public network by default so we must do that before connecting to this floating IP address.

```
$ sudo ip link set br-ex up
$ sudo ip route add 172.24.4.0/24 dev br-ex
$ sudo ip addr add 172.24.4.1/24 dev br-ex
```

Now you should be able to connect to the VM via its floating IP address. First, ping the address.

```
$ ping -c 1 172.24.4.8
PING 172.24.4.8 (172.24.4.8) 56(84) bytes of data.
64 bytes from 172.24.4.8: icmp_seq=1 ttl=63 time=0.823 ms

--- 172.24.4.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.823/0.823/0.823/0.000 ms
```

Now SSH to the VM:

```
$ ssh -i id_rsa_demo cirros@172.24.4.8 hostname
test1
```

5.1.6 Adding Another Compute Node

After completing the earlier instructions for setting up devstack, you can use a second VM to emulate an additional compute node. This is important for OVN testing as it exercises the tunnels created by OVN between the hypervisors.

Just as before, create a throwaway VM but make sure that this VM has a different host name. Having same host name for both VMs will confuse Nova and will not produce two hypervisors when you query nova hypervisor list later. Once the VM is setup, create the `stack` user:

```
$ git clone https://git.openstack.org/openstack-dev/devstack.git
$ sudo ./devstack/tools/create-stack-user.sh
```

Switch to the `stack` user and clone DevStack and networking-ovn:

```
$ sudo su - stack
$ git clone https://git.openstack.org/openstack-dev/devstack.git
$ git clone https://git.openstack.org/openstack/networking-ovn.git
```

networking-ovn comes with another sample configuration file that can be used for this:

```
$ cd devstack
$ cp ../networking-ovn/devstack/computenode-local.conf.sample local.conf
```

You must set SERVICE_HOST in local.conf. The value should be the IP address of the main DevStack host. You must also set HOST_IP to the IP address of this new host. See the text in the sample configuration file for more information. Once that is complete, run DevStack:

```
$ cd devstack
$ ./stack.sh
```

This should complete in less time than before, as it's only running a single OpenStack service (nova-compute) along with OVN (ovn-controller, ovs-vswitchd, ovsdb-server). The final output will look something like this:

```
This is your host IP address: 172.16.189.30
This is your host IPv6 address: ::1
2017-03-09 18:39:27.058 | stack.sh completed in 1149 seconds.
```

Now go back to your main DevStack host. You can use admin credentials to verify that the additional hypervisor has been added to the deployment:

```
$ cd devstack
$ . openrc admin

$ openstack hypervisor list
+----+------------------------+-----------------+---------------+-------+
| ID | Hypervisor Hostname    | Hypervisor Type | Host IP       | State |
+----+------------------------+-----------------+---------------+-------+
|  1 | centos7-ovn-devstack   | QEMU            | 172.16.189.6  | up    |
|  2 | centos7-ovn-devstack-2 | QEMU            | 172.16.189.30 | up    |
+----+------------------------+-----------------+---------------+-------+
```

You can also look at OVN and OVS to see that the second host has shown up. For example, there will be a second entry in the Chassis table of the OVN_Southbound database. You can use the `ovn-sbctl` utility to list chassis, their configuration, and the ports bound to each of them:

```
$ ovn-sbctl show

Chassis "ddc8991a-d838-4758-8d15-71032da9d062"
    hostname: "centos7-ovn-devstack"
    Encap vxlan
        ip: "172.16.189.6"
        options: {csum="true"}
    Encap geneve
        ip: "172.16.189.6"
        options: {csum="true"}
    Port_Binding "97c970b0-485d-47ec-868d-783c2f7acde3"
    Port_Binding "e003044d-334a-4de3-96d9-35b2d2280454"
    Port_Binding "cr-lrp-08d1f28d-cc39-4397-b12b-7124080899a1"
Chassis "b194d07e-0733-4405-b795-63b172b722fd"
    hostname: "centos7-ovn-devstack-2.os1.phx2.redhat.com"
    Encap geneve
        ip: "172.16.189.30"
        options: {csum="true"}
    Encap vxlan
        ip: "172.16.189.30"
        options: {csum="true"}
```

You can also see a tunnel created to the other compute node:

```
$ ovs-vsctl show
...
Bridge br-int
    fail_mode: secure
    ...
    Port "ovn-b194d0-0"
        Interface "ovn-b194d0-0"
            type: geneve
            options: {csum="true", key=flow, remote_ip="172.16.189.30"}
```

(continues on next page)

(continued from previous page)

```
    ...
...
```

5.1.7 Provider Networks

Neutron has a "provider networks" API extension that lets you specify some additional attributes on a network. These attributes let you map a Neutron network to a physical network in your environment. The OVN ML2 driver is adding support for this API extension. It currently supports "flat" and "vlan" networks.

Here is how you can test it:

First you must create an OVS bridge that provides connectivity to the provider network on every host running ovn-controller. For trivial testing this could just be a dummy bridge. In a real environment, you would want to add a local network interface to the bridge, as well.

```
$ ovs-vsctl add-br br-provider
```

ovn-controller on each host must be configured with a mapping between a network name and the bridge that provides connectivity to that network. In this case we'll create a mapping from the network name "providernet" to the bridge 'br-provider".

```
$ ovs-vsctl set open . \
external-ids:ovn-bridge-mappings=providernet:br-provider
```

Now create a Neutron provider network.

```
$ neutron net-create provider --shared \
--provider:physical_network providernet \
--provider:network_type flat
```

Alternatively, you can define connectivity to a VLAN instead of a flat network:

```
$ neutron net-create provider-101 --shared \
--provider:physical_network providernet \
--provider:network_type vlan \
--provider:segmentation_id 101
```

Observe that the OVN ML2 driver created a special logical switch port of type localnet on the logical switch to model the connection to the physical network.

```
$ ovn-nbctl show
...
 switch 5bbccbbd-f5ca-411b-bad9-01095d6f1316 (neutron-729dbbee-db84-4a3d-afc3-
↪82c0b3701074)
     port provnet-729dbbee-db84-4a3d-afc3-82c0b3701074
         addresses: ["unknown"]
...

$ ovn-nbctl lsp-get-type provnet-729dbbee-db84-4a3d-afc3-82c0b3701074
localnet

$ ovn-nbctl lsp-get-options provnet-729dbbee-db84-4a3d-afc3-82c0b3701074
network_name=providernet
```

If VLAN is used, there will be a VLAN tag shown on the localnet port as well.

Finally, create a Neutron port on the provider network.

```
$ neutron port-create provider
```

or if you followed the VLAN example, it would be:

```
$ neutron port-create provider-101
```

5.1.8 Run Unit Tests

Run the unit tests in the local environment with `tox`.

```
$ tox -e py27
$ tox -e py27 networking_ovn.tests.unit.test_ovn_db_sync
$ tox -e py27 networking_ovn.tests.unit.test_ovn_db_sync.TestOvnSbSyncML2
$ tox -e py27 networking_ovn.tests.unit.test_ovn_db_sync.TestOvnSbSyncML2
  .test_ovn_sb_sync
```

5.1.9 Run Functional Tests

you can run the functional tests with `tox` in your devstack environment:

```
$ cd networking_ovn/tests/functional
$ tox -e dsvm-functional
$ tox -e dsvm-functional networking_ovn.tests.functional.test_mech_driver\
  .TestPortBinding.test_port_binding_create_port
```

If you want to run functional tests in your local clean environment, you may need a new working directory.

```
$ export BASE=/opt/stack
$ mkdir -p /opt/stack/new
$ cd /opt/stack/new
```

Next, get networking_ovn, neutron and devstack.

```
$ git clone https://git.openstack.org/openstack/networking-ovn.git
$ git clone https://git.openstack.org/openstack/neutron.git
$ git clone https://git.openstack.org/openstack-dev/devstack.git
```

Then execute the script to prepare the environment.

```
$ cd networking-ovn/
$ ./networking_ovn/tests/contrib/gate_hook.sh
```

Finally, run the functional tests with `tox`

```
$ cd networking_ovn/tests/functional
$ tox -e dsvm-functional
$ tox -e dsvm-functional networking_ovn.tests.functional.test_mech_driver\
  .TestPortBinding.test_port_binding_create_port
```