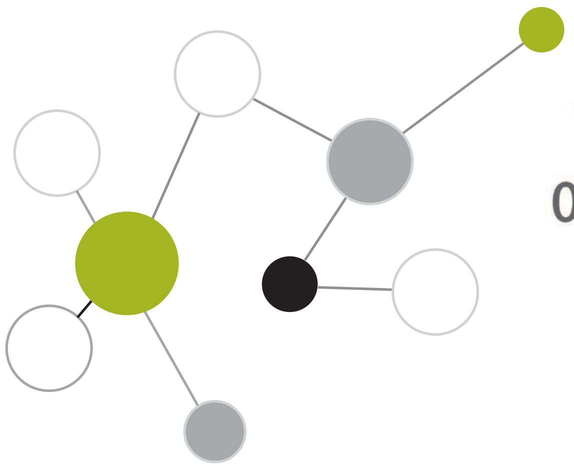
Integrating OpenDaylight VTN Manager with OpenStack



OpenDaylight is the largest open source SDN controller. The OpenDaylight virtual tenant network (VTN) is an application that provides a multi-tenant virtual network on an SDN controller. This article is a tutorial on how to integrate the OpenDaylight VTN Manager with OpenStack.

A virtual tenant network (VTN) allows users to define the network with a look and feel of the conventional L2/L3 network. Once the network is designed on VTN, it automatically gets mapped into the underlying physical network, and is then configured on the individual switch, leveraging the SDN control protocol. The definition of the logical plane makes it possible not only to hide the complexity of the underlying network but also to manage network resources better. This reduces the reconfiguration time of network services and minimises network configuration errors.

The technical introduction given above might seem complex to SDN beginners. In this article, I have tried to be as simple as I can, while teaching readers about VTN and its integration with OpenStack.

For the purpose of this article, I'm assuming readers have a basic understanding of SDN. So let me start with VTN directly.

The SDN VTN Manager helps you to aggregate multiple ports from the many underlying SDN-managed switches (both physical and virtual) to form the single isolated virtual tenant network (VTN). Each tenant network has the capability to function as an individual switch.

For example, consider that you have two physical switches (say, s1 and s2) and one virtual Open vSwitch (say, vs1) in your lab environment. Now, with the help of the VTN Manager, it is possible to group (aggregate) the three ports (say p1, p2 and p3) from switch s1 — i.e., s1p1, s1p2, s1p3; two ports from switch s2 — i.e., s2p1 and s2p2; and two ports from the virtual switch vs1 — i.e., vs1p1 and vs2p2, to form a single switch environment (say, VTN-01).

This means, virtually, the group (tenant) named VTN-01 is one switch with seven ports (s1p1, s1p2, s1p3, s2p1, s2p2, vs1p1 and vs2p2) in it. This VTN-01 will act exactly like a single isolated switch with the help of flows configured in the ports of all three switches by the OpenDaylight VTN Manager.

The above example explains the concept called port mapping in VTN, and will help beginners to understand the basic concept better. It will also help them to compare all other VTN concepts like VLAN mapping and MAC mapping.

VTN OpenStack integration

There are several ways to integrate OpenDaylight with OpenStack. This article will focus on the method that uses VTN features available on the OpenDaylight controller. During integration, the VTN Manager works as the network service provider for OpenStack.

The features of VTN Manager empower OpenStack to work in a pure OpenFlow environment, in which all the switches in the data plane are an OpenFlow switches. You could also refer to my blog on 'OpenDaylight Integration with OpenStack using OVSDB' from the link *http://www.cloudenablers.com/blog/opendaylight-integration-with-openstack/*.

The requirements are:

- OpenDaylight Controller
- OpenStack Control Node
- OpenStack Compute Node

OpenDaylight support for OpenStack network types

Till the Boron release, OpenDaylight (ODL) only supported'Local'network type in OpenStack and there was no support for VLAN. You may wonder why the developers never speak about VxLAN and GRE tunnelling network types support. You can answer that question if you recall the example I mentioned at the beginning of this article.

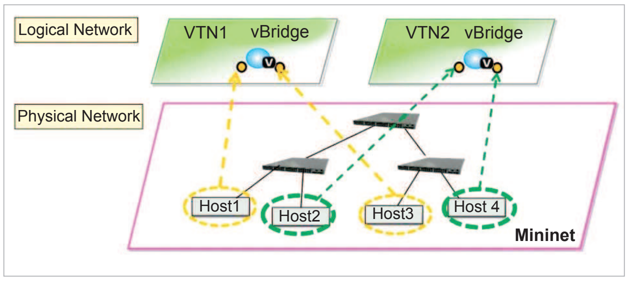
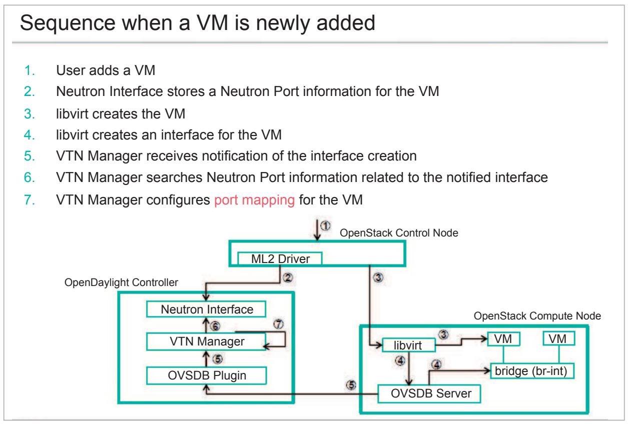


Figure 1: Virtual Tenant Network


Sequence when a VM is newly added

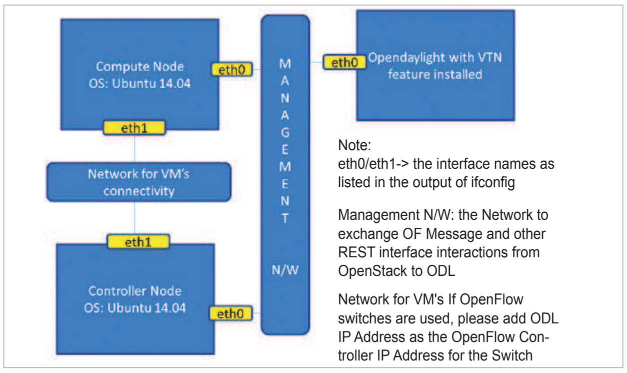
1. User adds a VM
2. Neutron Interface stores a Neutron Port information for the VM
3. libvirt creates the VM
4. libvirt creates an interface for the VM
5. VTN Manager receives notification of the interface creation
6. VTN Manager searches Neutron Port information related to the notified interface
7. VTN Manager configures port mapping for the VM

Figure 2: Request flow

To recap, I said that with the help of the VTN Manager, the user can group multiple ports from multiple switches in the infrastructure to form a single isolated network.

Let's compare this with our OpenStack environment, which has two Open vSwitches installed in the controller and compute node.

1. Whenever a new network is created in OpenStack, VTN Manager creates a new VTN in ODL.
2. Whenever a new sub-network is created, VTN Manager handles it and creates a vBridge under the VTN. vBridge is nothing but the virtual switch.
3. When a new VM is created in OpenStack, the addition of a new port in the Open vSwitch of the compute node is captured by VTN Manager, and it creates a vBridge interface in the newly created vBridge and maps that Open vSwitch port with the particular vBridge port.
4. In this case, the port (say, vs1p1) of the DHCP agent in the Open vSwitch of the controller node and the port (vs2p1) of the VM in the compute node are isolated from the actual Open vSwitch, using the flow entries from the OpenDaylight VTN Manager, to form a new virtual switch environment called the virtual tenant network.
5. When the packet sent from the DHCP agent reaches the OpenStack controller's Open vSwitch port vs1p1, then flow entries will tell the port vs1p1 to forward the packet to the compute node's Open vSwitch port vs2p1 using the underlying physical network. This packet will be sent as a regular TCP packet with a source and destination MAC address, which means that the traffic created in one network can be sent as a regular packet across the controller and compute node without any tunnelling protocol.
6. This explains why support for VxLAN and GRE network types is not required.



Note:
eth0/eth1-> the interface names as listed in the output of ifconfig

Management N/W: the Network to exchange OF Message and other REST interface interactions from OpenStack to ODL

Network for VM's If OpenFlow switches are used, please add ODL IP Address as the OpenFlow Controller IP Address for the Switch

Figure 3: LAB layout

LAB set-up layout

The VTN features support multiple OpenStack nodes. Hence, you can deploy multiple OpenStack compute nodes.

In the management plane, OpenDaylight controller, OpenStack nodes and OpenFlow switches (optional) should communicate with each other.

In the data plane, Open vSwitches running in OpenStack nodes should communicate with each other through physical or logical OpenFlow switches (optional). Core OpenFlow switches are not mandatory. Therefore, you can directly connect to the Open vSwitches.

You may need to disable the firewall (UFW) in all the nodes to reduce the complexity.

Installing OpenStack with the Open vSwitch configuration

Installing OpenStack is beyond the scope of this article; however, getting started with a minimal multi-node OpenStack deployment is recommended.

To help speed up the process, you could use my fully automated bash script for installing the OpenStack-Mitaka set-up at *https://github.com/CloudenablersPvtLtd/openstack-setup.*

Note: This script will install OpenStack and configure the Linux bridge for networking. But for the VTN integration to work in OpenStack, we need network configuration with Open vSwitch. So, you must uninstall the Linux bridge settings and reconfigure with Open vSwitch.

After the successful OpenStack installation, run the sanity test by performing the following operations.

Create two instances on a private subnet. Then add the floating IP address from your public network, verify that you can connect to them and that they can ping each other.

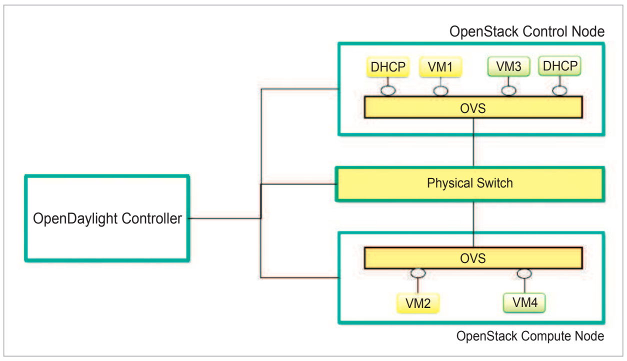


Figure 4: VTN OpenStack architecture

Installing OpenDaylight

The OpenDaylight controller runs in a JVM. The OpenDaylight-Boron release requires OpenJDK8, which you can install using the command given below:

```
$apt-get install openjdk-8-jdk
```

Download the latest OpenDaylight-Boron package from the official repo, as follows:

```
$wget https://nexus.opendaylight.org/content/repositories/
opendaylight.release/org/opendaylight/integration/
distribution-karaf/0.5.1-Boron-SR1/distribution-karaf-0.5.1-
Boron-SR1.tar.gz
```

Untar the file as the root user, and start OpenDaylight using the commands given below:

```
$ tar -xvf distribution-karaf-0.5.1-Boron.tar.gz
$ cd distribution-karaf-0.5.1-Boron.tar.gz
$ ./bin/karaf
```

Now, you should be in OpenDaylight's console. Install all the required features, as follows:

```
opendaylight-user@root> feature:install odl-vtn-manager-
neutron
opendaylight-user@root> feature:install odl-vtn-manager-rest
opendaylight-user@root> feature:install odl-mdsal-apidocs
opendaylight-user@root> feature:install odl-dlux-all
```

Feature installation may take some time. Once the installation is complete, you can check whether everything is working fine by using the following *curl* call:

```
$ curl -u admin:admin http://<ODL_IP>:8080/controller/nb/v2/
neutron/networks
```

The response should be an empty network list if OpenDaylight is working properly.

Now, you should be able to log into the DLUX interface on *http://<ODL_IP>:8181/index.html*.

The default username and password are *admin/ admin*.

Additionally, you could find useful log details at the following location:

```
$ tail -f /<directory_of_odl>/data/log/karaf.log
$ tail -f /<directory_of_odl>/logs/web_access_log_2015-12.txt
```

Now, you have a working OpenDaylight-Boron set-up. Let's get into the integration part.

Configuring OpenStack for VTN integration

Step 1

Erase all VMs, networks, routers and ports in the controller node, since you already have a working OpenStack set-up. You might test for VM provisioning as a sanity test, but before integrating OpenStack with OpenDaylight, you must clean up all the unwanted data from the OpenStack database. When using OpenDaylight as the Neutron back-end, ODL expects to be the only source for Open vSwitch configuration. Because of this, it is necessary to remove existing OpenStack and Open vSwitch settings to give OpenDaylight a clean slate.

The following steps will guide you through the cleaning process.

- Delete instances, as follows:

```
$ nova list
$ nova delete <instance names>
```

- Remove links from subnets to routers, as follows:

```
$ neutron subnet-list
$ neutron router-list
$ neutron router-port-list <router name>
$ neutron router-interface-delete <router name> <subnet ID or name>
```

- Delete subnets, nets and routers, as follows:

```
$ neutron subnet-delete <subnet name>
$ neutron net-list
$ neutron net-delete <net name>
$ neutron router-delete <router name>
```

- Check that all ports have been cleared – at this point, this should be an empty list:

```
$ neutron port-list
```

- Stop the Neutron service, as follows:

```
$ service neutron-server stop
```

While Neutron is managing the OVS instances on the compute and control nodes, OpenDaylight and Neutron may be in conflict. To prevent issues, let’s turn off the Neutron server on the network controller and Neutron’s Open vSwitch agents on all hosts.

Step 2: Configuring Open vSwitches in the controller and compute nodes

The Neutron plugin in every node must be removed because only OpenDaylight will be controlling the Open vSwitches. So, on each host, we will erase the pre-existing Open vSwitch config and set OpenDaylight to manage the Open vSwitch:

```
$ apt-get purge neutron-plugin-openvswitch-agent
$ service openvswitch-switch stop
$ rm -rf /var/log/openvswitch/*
$ rm -rf /etc/openvswitch/conf.db
$ service openvswitch-switch start
$ ovs-vsctl show
# The above command must return the empty set except
OpenVswitch ID and it’s Version.
```

Step 3:Connecting Open vSwitch to OpenDaylight

Use the command given below to make OpenDaylight administer Open vSwitch:

```
$ ovs-vsctl set-manager tcp:<OPENDAYLIGHT MANAGEMENT IP>:6640
```

You can copy the Open vSwitch ID from the command *ovs-vsctl show*. Execute the above command in all the nodes (controller and compute nodes) to set ODL as the manager for Open vSwitch:

```
$ ovs-vsctl show
```

The above command will show that you are connected to the OpenDaylight server, which will automatically create a *br-int* bridge.

```
[root@vinoth ~]# ovs-vsctl show
9e3b34cb-fefc-4br4-828s-084b3e55rtfd
Manager “tcp:192.168.2.101:6640”
Is_connected: true
Bridge br-int
Controller “tcp:192.168.2.101:6633”
fail_mode: secure
Port br-int
Interface br-int
ovs_version: “2.1.3”
```

If you get any error messages during bridge creation, you may need to log out from the OpenDaylight Karaf console and check the *90-vtn-neutron.xml* file from the following path *distribution-karaf-0.5.0-Boron/etc/opendaylight/karaf/*.

The contents of *90-vtn-neutron.xml* should be as follows:

```
bridgename=br-int
portname=eth1
protocols=OpenFlow13
failmode=secure
```

By default, if *90-vtn-neutron.xml* is not created, VTN uses *ens33* as the port name.

After running the ODL controller, please ensure it listens to the following ports: 6633, 6653, 6640 and 8080.

Note:

- 6633/6653 are the OpenFlow ports.
- 6640 is the OVS Manager port.
- 8080 is the port for the REST API.

Step 4: Configure* ml2_conf.ini *for the ODL driver

Edit *vi /etc/neutron/plugins/ml2/ml2_conf.ini* in all the required nodes and modify the following configuration. Leave the other configurations as they are.

```
[ml2]
type_drivers = local
tenant_network_types = local
mechanism_drivers = opendaylight
[ml2_odl]
password = admin
username = admin
url = http://<OPENDAYLIGHT SERVER’s IP>:8080/controller/nb/
v2/neutron
```

Step 5: Configure the Neutron database

Reset the Neutron database, as follows:

```
$ mysql -uroot -p
$ drop database neutron;
$ create database neutron;
$ grant all privileges on neutron.* to ‘neutron’@’localhost’
identified by ‘<YOUR NEUTRON PASSWORD>’;
$ grant all privileges on neutron.* to ‘neutron’@’%’
identified by ‘<YOUR NEUTRON PASSWORD>’;
$ exit
$ su -s /bin/sh -c "neutron-db-manage --config-file /etc/
neutron/neutron.conf --config-file /etc/neutron/plugins/ml2/
ml2_conf.ini upgrade head" neutron
```

Restart the Neutron-server, as follows:

```
$ service neutron-server start
```

Step 6: Install the* Python-networking-odl *Python module

IMPORTANT: You should get the status alert if the Neutron service fails to start by this time. Don’t worry. This is a temporary issue since you have enabled OpenDaylight as the *mechanism_driver* but not yet installed the Python module for it.

Install the *Python-networking-odl* Python module, as follows:

```
$ apt-get install python-networking-odl
```

Now, restart the Neutron server and check its status. It should be running without errors.

Step 7: Verify the integration

We have almost completed the integration of OpenStack with VTN. Now, create initial networks in OpenStack and check whether a new network is created and posted to ODL, for which VTN Manager creates a VTN.

Use the *curl* commands given below to verify the creation of the network and VTN:

```
$ curl --user "admin":"admin" -H "Content-type: application/
json" -X GET http://<ODL_IP>:8181/restconf/operational/
vtn:vtns/
$ curl -u admin:admin http://<ODL_IP>:8080/controller/nb/v2/
neutron/ networks
```

Whenever a new sub-network is created in the OpenStack Horizon, VTN Manager will handle it and create a vBridge under the VTN. When you create a new VM in OpenStack, the interface (br-int) mentioned as the integration bridge in the configuration file will be added with more interfaces, and the network is provisioned for it by the VTN Neutron bundle. The addition of the new port is captured by VTN Manager, and it creates a vBridge interface with port mapping.

When the VM starts to communicate with the other VMs that have been created, VTN Manager will install flows in the OVS and other OpenFlow switches to facilitate communication between the VMs.

Note: To access OpenDaylight RestConf API documentation, use the link *http://<ODL_IP>:8181/apidoc/explorer/index.html*, which points to your *ODL_IP*.

If everything works correctly, you will able to communicate with other VMs created in the different compute nodes.

The VTN project doesn't support the vRouter up to the Boron release, which means that the floating IP operation in OpenStack is not supported when integrating VTN Manager with OpenStack. It might support the vRouter in the Carbon or Nitrogen releases. END

References

[1] *http://www.hellovinoth.com/*
[2] *http://www.cloudenablers.com/blog/*
[3] *https://www.opendaylight.org/*
[4] *http://docs.openstack.org/*

By: Vinoth Kumar Selvaraj

The author is a DevOps engineer at Cloudenablers Inc., a cloud technology startup based in Chennai. He has also worked as a book reviewer with PackPub Publishers, for books related to OpenStack. He blogs at *http://www.hellovinoth.com.* His Twitter handle is *@vinoth6664*.

Statement about ownership and other particulars about
OPEN SOURCE FOR YOU
FORM IV (See Rule 8)

1.	Place of publication	:	New Delhi
2.	Periodicity of its publication	:	Monthly
3.	Printer's Name Nationality Address	: : :	Ramesh Chopra Indian OPEN SOURCE FOR YOU D-87/1, Okhla Industrial Area, Phase-1, New Delhi 110020
4.	Publisher's Name Nationality and address	:	Same as (3) above
5.	Names and addresses of individuals who own the newspaper & partners or shareholders holding more than 1% of the total capital	:	**EFY Enterprises Pvt Ltd** D-87/1, Okhla Industrial Area, Phase-1, New Delhi 110020

I, Ramesh Chopra, hereby declare that the particulars given above are true to the best of my knowledge and belief.

Date: 28-2-2017

Ramesh Chopra
Publisher