Using OpenStack With OpenDaylight

Dave Meyer, Brocade
Kyle Mestery, Cisco
Brent Salisbury, Red Hat
Madhu Venugopal, Red Hat
5-11-2014
What You Will Walk Away With

• An overview of how OpenStack and OpenDaylight integrate together
• A demo of bringing up a multi-node OpenStack environment
• A demo of bringing using OpenDaylight with OpenStack Neutron for virtual tenant networks
What is OpenDaylight?

OpenDaylight is an **Open Source Software** project under the **Linux Foundation** with the goal of furthering the adoption and innovation of **Software Defined Networking (SDN)** through the creation of a common industry supported platform.

<table>
<thead>
<tr>
<th>Code</th>
<th>Acceptance</th>
<th>Community</th>
</tr>
</thead>
<tbody>
<tr>
<td>To create a robust, extensible,</td>
<td>To get broad industry acceptance amongst vendors and users</td>
<td>To have a thriving and growing technical community contributing to the code base, using the code in commercial products, and adding value above, below and around.</td>
</tr>
<tr>
<td>open source code base that covers</td>
<td>• Using OpenDaylight code directly or through vendor products</td>
<td></td>
</tr>
<tr>
<td>the major common components</td>
<td>• Vendors using OpenDaylight code as part of commercial products</td>
<td></td>
</tr>
<tr>
<td>required to build an SDN solution</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What is OpenDaylight building?

OpenDaylight is an open *community* that is building:

- An evolvable SDN *platform* capable of handling diverse use cases and implementation approaches
- Common abstractions of capabilities NorthBound for people to program
- Intermediation of those capabilities to multiple Southbound implementations
- Programmable Network services
- Network Applications
- Whatever else we need to make it work
OpenDaylight NorthBound API Layer - REST APIs

OpenDaylight Neutron REST-API

OVSDB Neutron Application

API Driven SAL (ADSAL)
- Configuration Service
- Inventory Service
- Connection Service
- Flow Programmer

Model Driven SAL (MDSAL)
- Inventory Service
- Connection Service
- Flow Programmer

OVSDB South-bound Plugin
- OVSDB Protocol Library
- Bidirectional JSON-RPC Library

OpenFlow 1.0 SB Plugin
- OpenFlow 1.0 Plugin
- OpenFlow 1.0 Library
- java.nio.socket

OpenFlow 1.3 SB Plugin
- OpenFlow 1.3 Plugin
- OpenFlow 1.3 Library
- Netty.io

Netty.io

OpenFlow 1.0

OpenFlow 1.3

OVSDB Protocol

OpenVSwitch

ML2 Plug-In

Neutron
What You Will Need

• OpenDaylight Virtualization Edition with OVSDB
  • Can be in a VM or on your laptop directly

• Two or more OpenStack Nodes
  • One node running control software and optionally compute services
  • One or more compute nodes
Logistics

• The Fedora20 VM has the following information:
  • Users:
    • root/password
    • odl/odl
  • Setup for DHCP for the image itself.
Boot Your VM Images

• Boot the VM which you will run OpenDaylight inside of.
  • Optionally bring-up OpenDaylight on your laptop natively.
  • This will work in either scenario.
• Verify IP addresses on your VMs (may require reboots).
  • This should be done for all VMs.
  • This may change once you import the OVF file.
OpenDaylight Configuration and Startup

- Edit your ODL configuration:
  - opendaylight/configuration/config.ini (change of.address)

- Optionally enable OpenFlow 1.3
  - opendaylight/configuration/config.ini (uncomment this)

- Bring OpenDaylight to life:
  - Handy RUN.sh script
    - This contains options for both OpenFlow 1.0 (default) or OpenFlow 1.3 (optional)

- Post ODL fixup:
  - From OSGI console:
    - lb | grep simple
    - stop <simple forwarding ID>
OpenStack VM Setup

- Copy the VM image twice:
  - Once for control and once for compute
- On both nodes:
  - cd /etc/sysconfig/networking-scripts
  - sudo cp ifcfg-eth0 ifcfg-eth1
  - Edit ifcfg-eth1 to change interface name
- On the control node:
  - Login as odl/odl
  - Copy local.conf.control to devstack/local.conf
  - Edit devstack/local.conf and change IP addresses
- On the compute node:
  - Login as odl/odl
  - Copy local.conf.compute to devstack/local.conf
  - Edit devstack/local.conf and change IP addresses
Example of stopping Simple Forwarding


lb | grep simple
 130|Active    | 4|samples.simpleforwarding (0.4.1)
true
osgi> stop 130
osgi> lb | grep simple
 130|Resolved  | 4|samples.simpleforwarding (0.4.1)
true
osgi>
Browse to your ODL Window over HTTP
Boot Up Your OpenStack Instances

• Control Node:
  • cd devstack
  • ./stack.sh

• Compute Node:
  • cd devstack
  • ./stack.sh
Your devstack will look like this
Login to Horizon (go to the IP of your control node)
Login as (admin/admin) to see the Horizon Dashboard
Spinup a VM
Spinup a VM (cont.)
Instance is now booted
Repeat process for a second VM
Checkout OpenDaylight Web GUI
Ping test between VMs

```
$ ifconfig eth0
eth0  Link encap:Ethernet  HWaddr FA:16:3E:FA:63:3A
     inet addr:10.0.0.2  Bcast:10.0.0.255  Mask:255.255.255.0
     inet6 addr: fe80::f016:7eff:fe3a:6f0 Scope:Link
     UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
     RX packets:35  errors:0  dropped:3  overruns:0  frame:0
     TX packets:34  errors:0  dropped:0  overruns:0  carrier:0
     collisions:0  txqueuelen:1000
     RX bytes:5404 (5.2 KiB) TX bytes:2922 (2.8 KiB)

$ ping -c 3 10.0.0.4
PING 10.0.0.4 (10.0.0.4): 56 data bytes
64 bytes from 10.0.0.4: seq=0 ttl=64 time=1.283 ms
64 bytes from 10.0.0.4: seq=1 ttl=64 time=0.999 ms
64 bytes from 10.0.0.4: seq=2 ttl=64 time=0.846 ms
--- 10.0.0.4 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.846/1.042/1.283 ms
```

Thank You!