# OpenDaylight Neutron Plugin

## Scope:

This blueprint covers creating an OpenDaylight plugin for OpenStack Neutron. OpenDaylight provides the ability to leverage advanced QoS and traffic forwarding to provide advanced virtual machine networking as part of a complete Open Source software-defined networking stack.

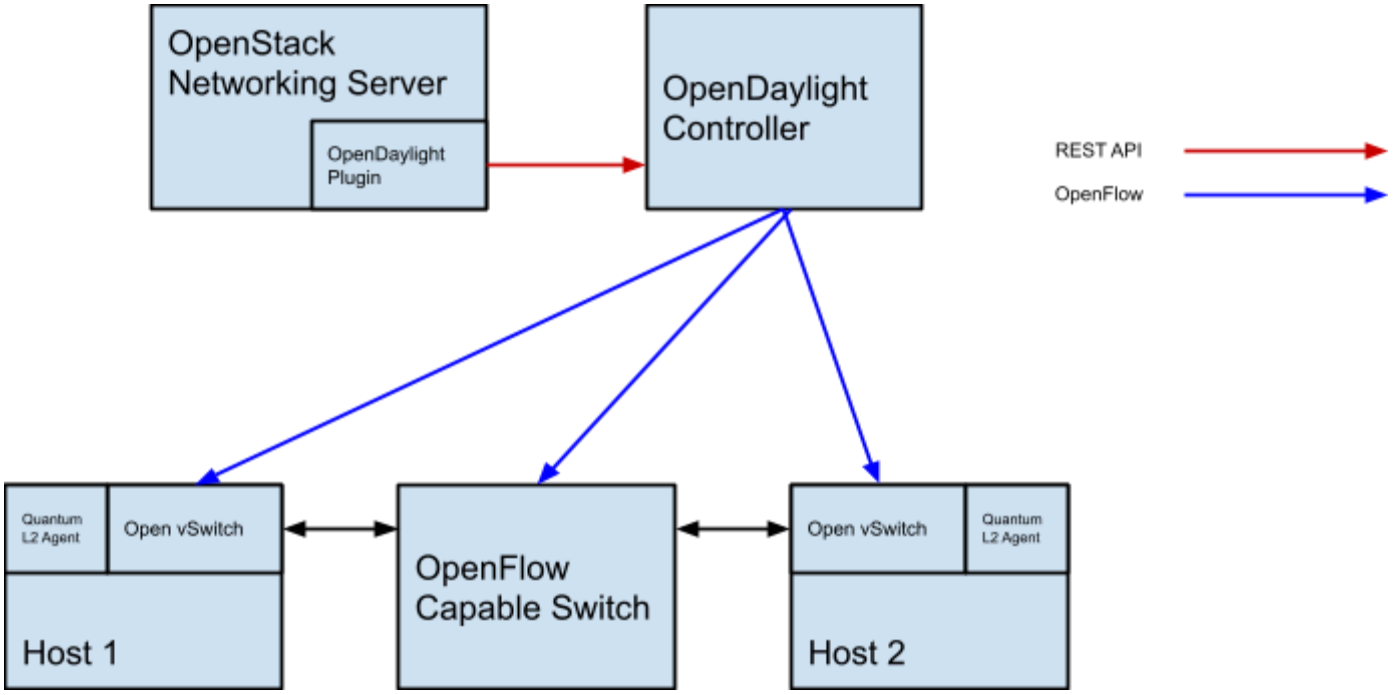
http://www.opendaylight.org/

## Use Cases:

1. Create network
2. Delete network
3. Create Port
4. Delete Port
5. L3 services (routing, load balancers)
6. Virtual machine live migration
7. QoS

## Implementation Overview:

The below diagram provides a high level overview of how the OpenDaylight Plugin will integrate into a working environment consisting of multiple hosts with virtual switches connected by a physical switch.




### Flow Setup

The following diagram shows flow setup.

## Data Model Changes:

N/A

**Configuration variables:**

- IP address and port number of the OpenDaylight controller.
- Username and password to login to the OpenDaylight controller.
- OVS Integration bridge name on the compute hosts

## Required Plugin support:

N/A

## Dependencies:

This plugin will require the OpenDaylight Controller to function. Instructions for downloading, installing, and running the OpenDaylight controller are available at the OpenDaylight website here: http://www.opendaylight.org/

## Workflow:

### Bridge Connection and creation:

When the ODL plugin is first initialized it has to perform a few steps to get the compute host ready for Instances:

1. Gather a list of hosts configured in nova.
2. Call the ODL BridgeDomain northbound to connect to OVSDB on each of the compute hosts.
3. Call the ODL BridgeDomain northbound to create the integration bridge (Name defined in the config) if not already created.

Pros:

1. Almost no used interaction required to setup bridges, hence low chance of human introduced configuration errors.

Cons:

1. Requires nova/keystone interactions.
2. Need some way to setup new compute hosts that are added to the Openstack deployment after Neutron is started.
3. This might not be desirable in some cases like OpenDOVE with the host agent.

Proposed solutions:

1. Make the auto configuration of bridges on the hosts configurable via the plugin's ini file with default turned to off.
2. We can introduce a thread or process that periodically polls nova for a list of compute hosts, checks for any new unconfigured hosts and does the bridge configuration on them (Expensive and state could be out of sync at times).
3. We can introduce an extension in Neutron that Nova can invoke via the Neutron manager v2 api to notify Neutron and the plugin of any new hosts that it sees (Requires modifications in Nova but most effective and requires minimal interactions)

### Network creation:

No foreseeable ODL configuration required for network creation. If VTN or OpenDOVE are used then network segmentation needs to be managed by ODL instead of Neutron.
(TBD)

### Port Creation:

The conventional OVS workflow for creating ports is as follows (simplified):

1. New instance booted in Nova.
2. Nova invokes a create_port call via the Neutron API v2 with device_owner set to Compute:None and device_id set to the instance id. WIth the portbindings extension we also get binding:host_id to signify what compute host the instance is being spawned on.

3. Neutron receives the create_port call and passes it on to it's plugin(s).
4. The plugin creates a DB port with the attributes specified and returns the UUID of the newly created port. The port status is administratively DOWN at this point.
5. Nova writes libvirt domain XML for the instance with the port UUID as the instance port's external_id.
6. Instance boots up, a new port appears on the bridge.
7. The OVS L2 agent picks up this new port, extracts the external_id and makes an RPC call to Neutron with the external_id (Neutron Port UUID).
8. The Neutron plugin responds with relevant information about that port back to the agent via the RPC channel with information like segmentation type, id etc.
9. The agent receives information back from Neutron, performs the required configuration using ovs-vsctl rootwrapped commands and sets port status to UP.

For the ODL ML2 Mechanism driver we need a different workflow as there is no agent involved. We can solve this in a few different ways:

1. Send all port information from the Neutron plugin to Opendaylight to be cached within ODL until it sees a new port on the bridge, extracts the external id and applies relevant configuration from it's cache to that port.
    a. Use Ryan Mott's neutron API in Opendaylight to replicate Openstack port creation in Opendaylight.
2. Poll all ports on a bridge from the Neutron plugin periodically and apply port configuration from Neutron via the bridgedomain northbound API (Expensive)
3. Opendaylight can call the Neutron northbound API to get details of a port using the external id when a new port shows up on the bridge (Requires neutron configuration in ODL).