OPEN DAYLIGHT SUMMIT

# The OpenDaylight OVSDB Project as a Solution for Network Virtualization Needs in OpenStack

Sam Hague(shague@redhat.com) [irc : shague]
Flavio Fernandes (ffernand@redhat.com) [irc : flaviof]
Anil Vishnoi (avishnoi@brocade.com) [irc : vishnoianil]

# ....will talk about:

- What the OVSDB Project offers?
- Why it's the Center of Attraction?
- Brief Overview of Open vSwitch & Management Protocol
- High Level Architecture and Control Flow
- What we have accomplished in Lithium
- What are we planning for Beryllium?
- Let’s ./stack!
- Looking to contribute?

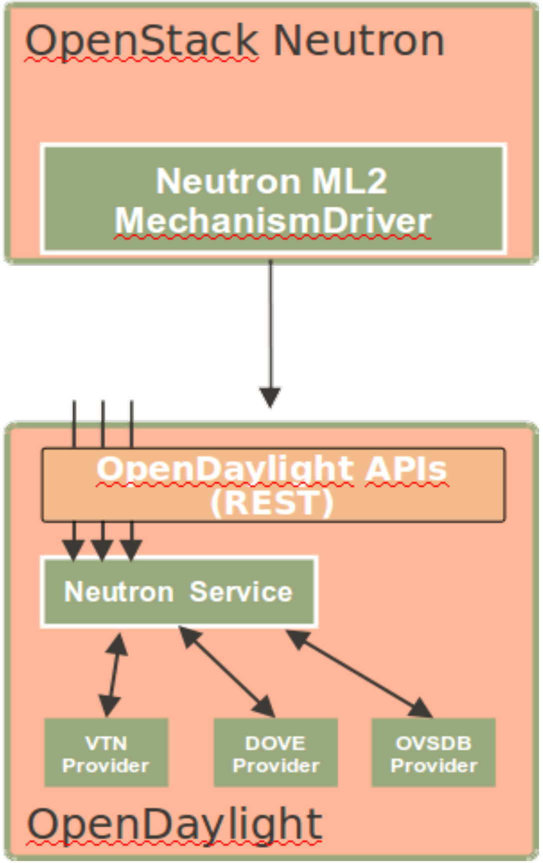
# What the OVSDB Project offers?

- … network virtualization solution for Openstack
- … southbound plugin to configure Open vSwitch
- … library to encode/decode OVSDB protocol
- … rest & restconf interface to configure Open vSwitch
- … challenging Software Defined Networking problems to solve
- … challenging work items, if you want to contribute :)

# ....will talk about:

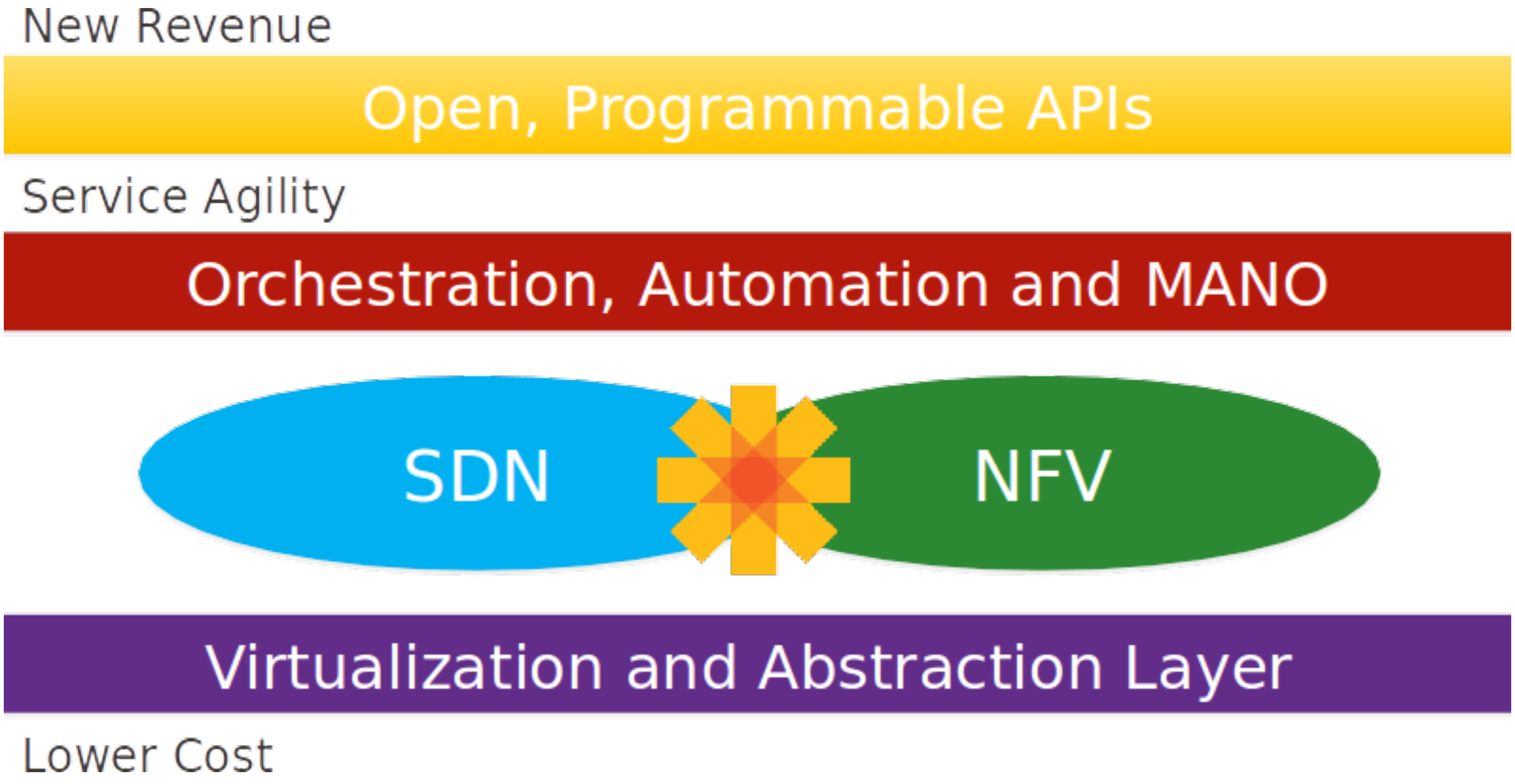
- What the OVSDB Project offers?
- Why it's the Center of Attraction?
- Brief Overview of Open vSwitch & Management Protocol
- High Level Architecture and Control Flow
- What we have accomplished in Lithium
- What are we planning for Beryllium?
- Let’s ./stack!
- Looking to contribute?

# Reason 1: OpenStack Integration



- OpenDaylight exposes a single common OpenStack Service Northbound
  - API exposed matches Neutron REST API precisely
  - Multiple implementations of Neutron providers in OpenDaylight
- The OpenDaylight OpenStack Neutron Service is a thin plugin that is a simple pass through of the Neutron REST APIs
  - Simplifies OpenStack plugin
  - Pushes complexity to OpenDaylight

# Reason 2: SDN, NFV and OpenDaylight



New Revenue

Open, Programmable APIs

Service Agility

Orchestration, Automation and MANO

SDN

NFV

Virtualization and Abstraction Layer

Lower Cost

# Reason 3: Growing Pains with OpenStack Neutron

- Neutron is a tenant facing cloud networking API, but a poor SDN controller implementation.
    - Complex architecture with neutron agents and custom protocols to communicate network needs to OVS network devices.
    - The result has had fundamental scaling and robustness issues.
- Neutron as an API service is focused on tenants.
    - It does not provide any APIs or functionality for managing your network.
    - This would show up most when debugging a network issue and needing to use two separate tools (Neutron, plus host tools, plus fabric tools).

# How OpenDaylight can help with those pains and other benefits

- OpenDaylight is designed to handle heterogeneous networking needs at scale using common network protocols to communicate to a wide variety of networking devices.
- OpenDaylight can manage both network virtualization needs (driven directly by OpenStack) and manage underlying physical fabric.  Especially useful to inform the underlay about the overlay.
- HW support for offloads in the form of, e.g. hw_vtep are a natural extension of ODL.

# ....will talk about:

- What the OVSDB Project offers?
- Why it's the Center of Attraction?
- Brief Overview of Open vSwitch & Management Protocol
- High Level Architecture and Control Flow
- What we have accomplished in Lithium
- What are we planning for Beryllium?
- Let’s ./stack!
- Looking to contribute?

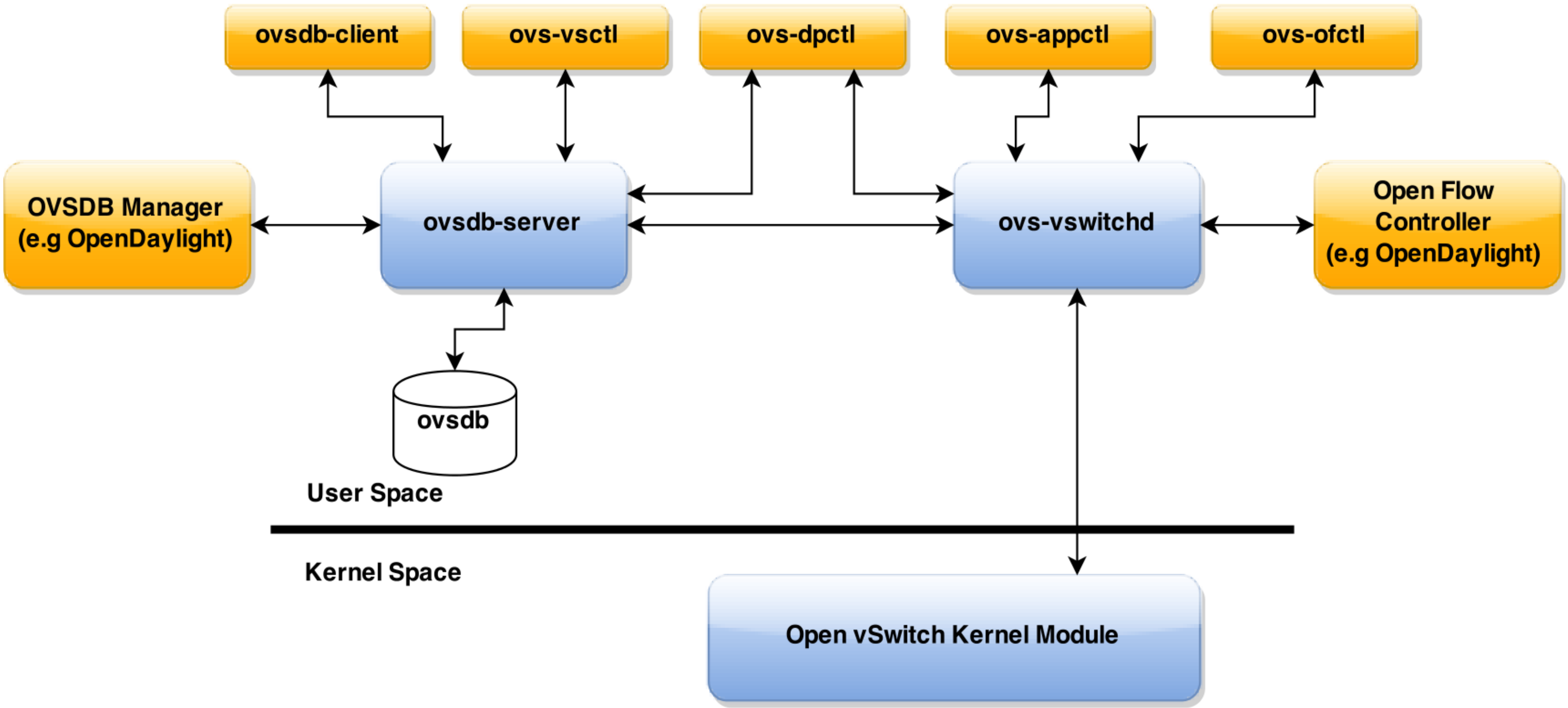
# Brief Overview of Open vSwitch: *Main Features*

- Open vSwitch is an open source switching stack for virtualization.
- Enables massive network automation through programmatic extensions
- Open vSwitch brings many features standard in hardware devices to virtualized environments:
  - VLANs
  - A variety of tunneling protocols
  - LACP and other bonding modes
  - QoS shaping and policing
  - ACLs over a range of L2-L4 protocols
  - NetFlow, sFlow, IPFIX, mirroring
- Plus remote programmability and management features:
  - OVSDB
  - OpenFlow 1.0/1.3 support
  - All features and status remotely configurable and viewable.
  - Support for many extensions (openflow, nicira)

# Brief Overview of Open vSwitch: *Programmability Aspect*

- Extensive flow matching capabilities
  - Layer 1 – Tunnel ID, In Port, QoS priority, skb mark
  - Layer 2 – MAC address, VLAN ID, Ethernet type
  - Layer 3 – IPv4/IPv6 fields, ARP
  - Layer 4 – TCP/UDP, ICMP, ND
- Possible chain of actions
  - Output to port (port range, flood, mirror)
  - Discard, Resubmit to table x
  - Packet Modification (Push/Pop VLAN header, TOS, ...)
  - Send to controller, Learn
- Centralized Control through
  - OpenFlow connection per datapath
  - Management channel per system

# Brief Overview of Open vSwitch: *High Level Architecture*



# Open vSwitch Components

- ovsdb-server
  - Database that holds switch-level configuration
  - Custom database with nice properties: value constraints, weak references, garbage collection
  - Log based
  - Speaks management protocol (OVSDB, JSON-RPC) to manager and ovs-vswitchd
  - Supports multiple connections
- ovs-vswitchd:
  - Core component in the system:
    - Communicates with outside world using OpenFlow
    - Communicates with ovsdb--server using management protocol
    - Communicates with kernel module over netlink
    - Communicates with the system through netdev abstract interface
  - Packet classifier supports efficient flow lookup with wildcards and “explodes” these (possibly) wildcard rules for fast processing by the datapath
  - Supports multiple independent datapaths (bridges)

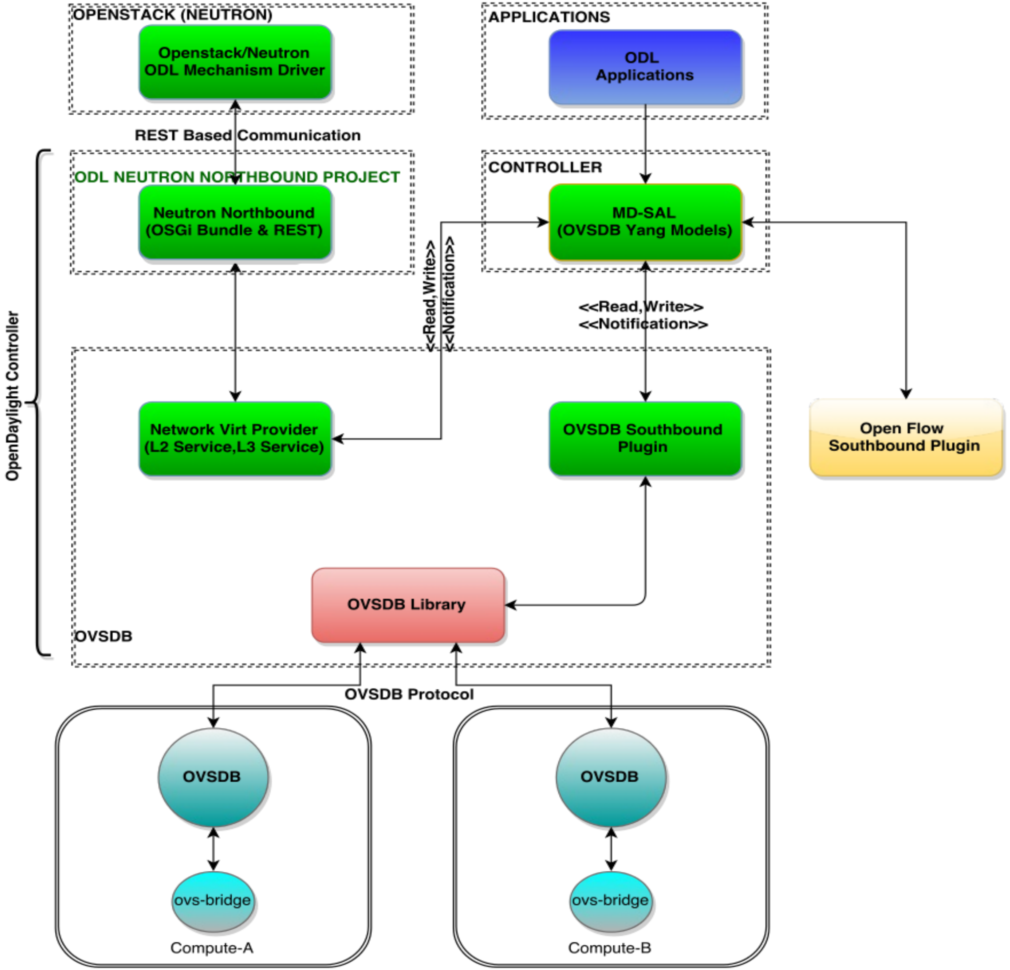
# OVSDB Management Protocol

- JSON-RPC based protocol
- Interact with OVSDB database for managing and configuring Open vSwitch Instance
- Provides methods like
  - Transact
  - Monitor
  - Get Schema
  - Notifications
- Allows database operations like
  - Insert and Delete
  - Mutate
  - Update
  - Select
  - Abort
  - Comment

# ....will talk about:

- What the OVSDB Project offers?
- Why it's the Center of Attraction?
- Brief Overview of Open vSwitch & Management Protocol
- High Level Architecture and Control Flow
- What we have accomplished in Lithium
- What are we planning for Beryllium?
- Let’s ./stack!
- Looking to contribute?

# High Level Architecture



# NetVirt Logical Flow Pipeline

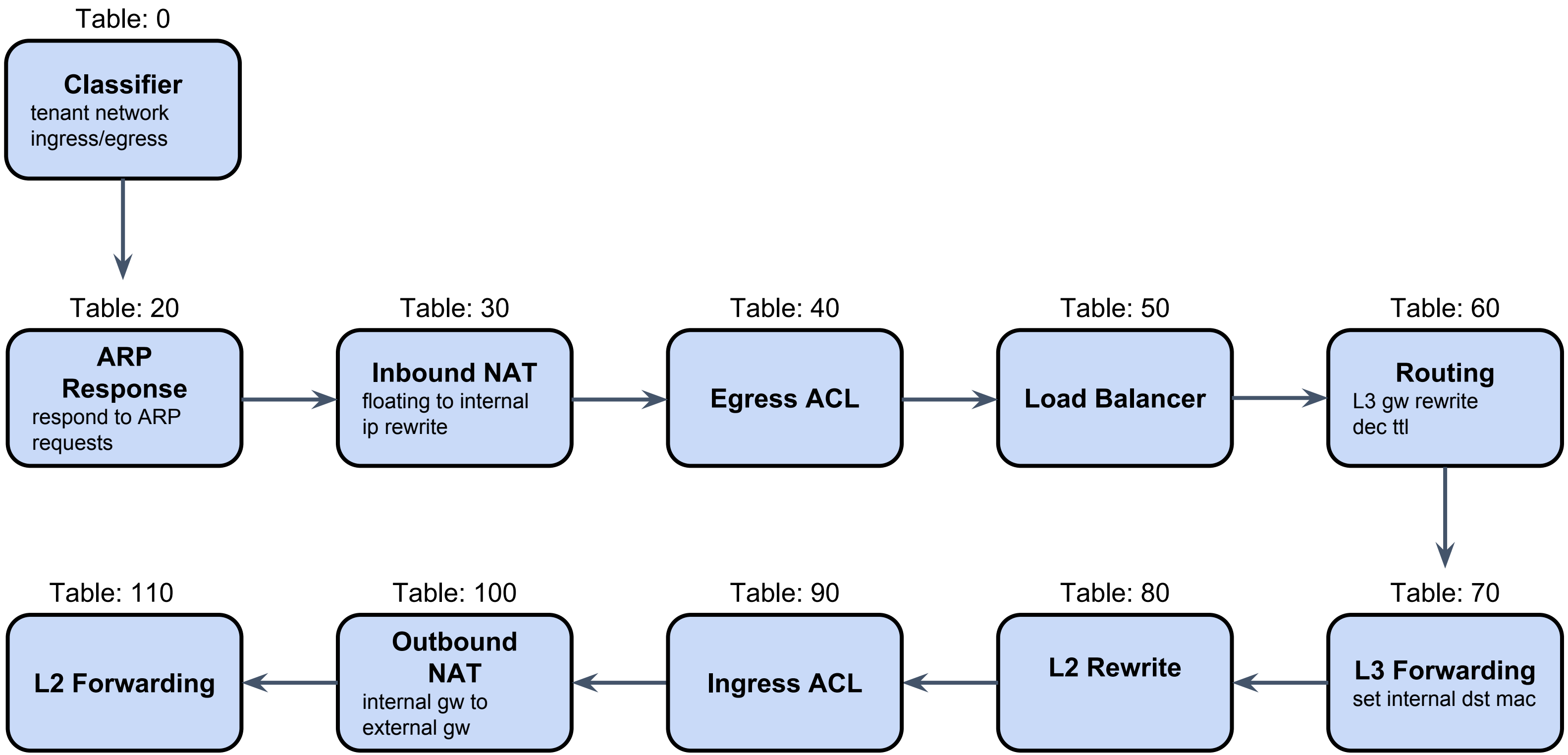


Table: 0

**Classifier**
tenant network ingress/egress

Table: 20

**ARP Response**
respond to ARP requests

Table: 30

**Inbound NAT**
floating to internal ip rewrite

Table: 40

**Egress ACL**

Table: 50

**Load Balancer**

Table: 60

**Routing**
L3 gw rewrite
dec ttl

Table: 70

**L3 Forwarding**
set internal dst mac

Table: 80

**L2 Rewrite**

Table: 90

**Ingress ACL**

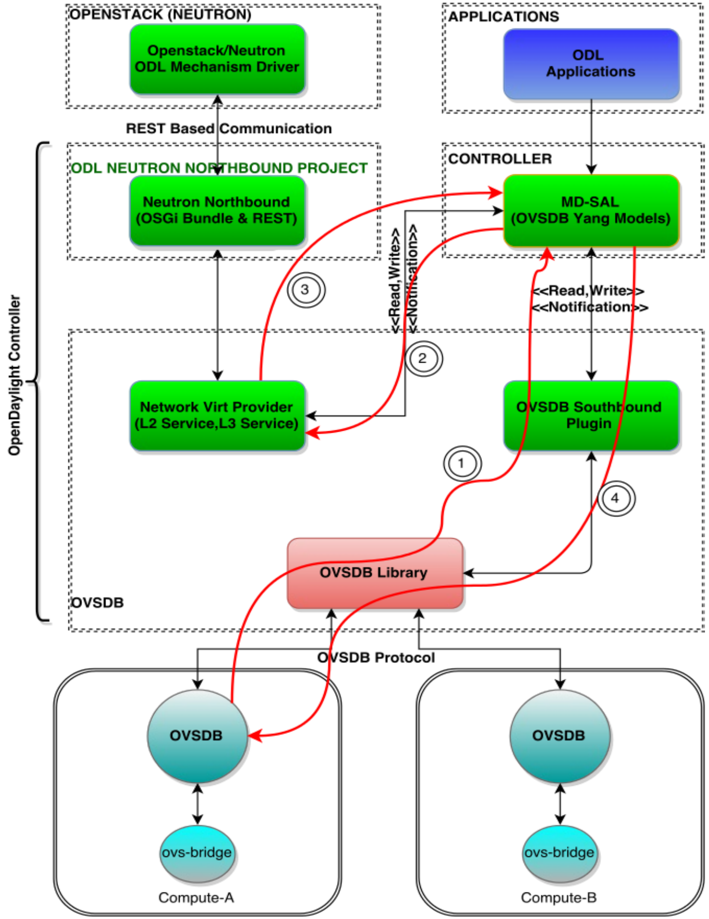
Table: 100

**Outbound NAT**
internal gw to external gw

Table: 110

**L2 Forwarding**

# High Level Control Flow: *Connect Ovsdb to Controller*



(1) Connect compute node to controller by setting ovsdb manager pointing to controller

   (a) Southbound plugin accepts connection
   (b) It writes data to operational data store
   (c) Data store notifies addition of node to all the listeners

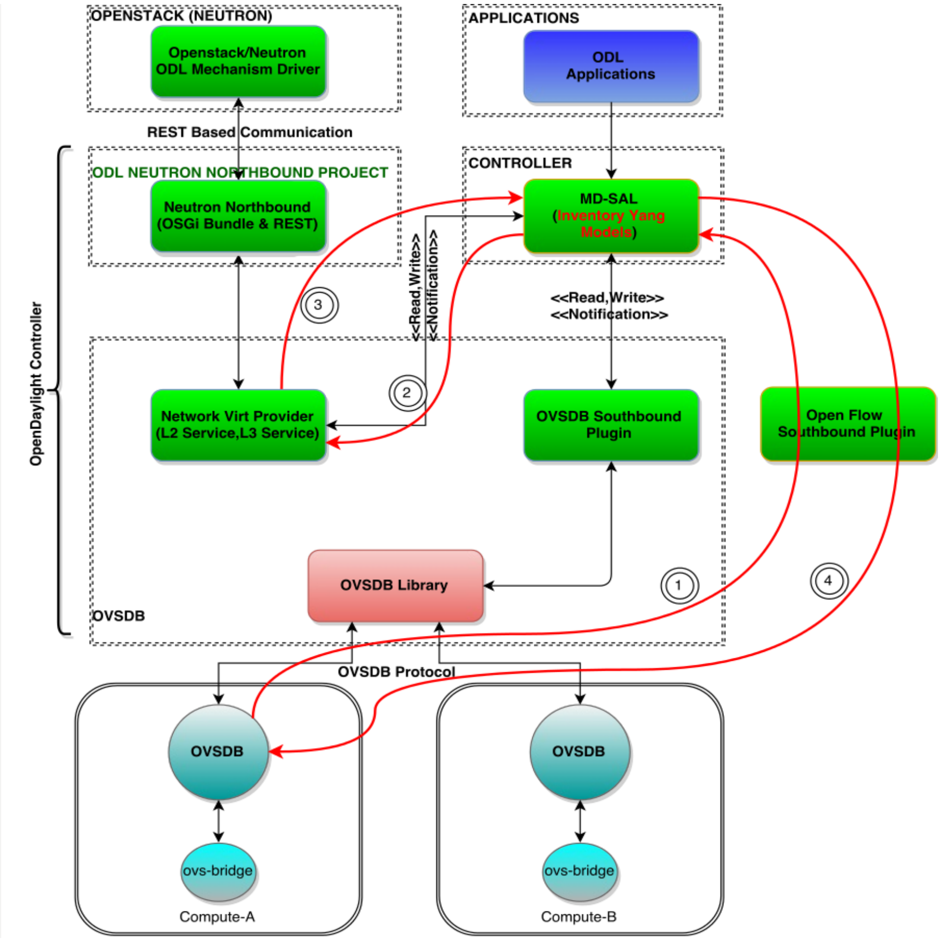
(2) MD-SAL data store broker sends notification to NetVirt about new node

(3) NetVirt writes data to MD-SAL config data store to create “br-int” and set controller

(4) MD-SAL data store notifies Southbound plugin about the “br-int” config data addition

   (a) Southbound plugin instructs OVSDB library to create bridge
   (b) Also sets controller for the bridge to connect to controller through OpenFlow Plugin

# High Level Control Flow: *Connect “br-int” to Controller*



(1) Connect “br-int” to controller

- (a) OpenFlow southbound plugin accepts connection
- (b) It writes the new node data to operational data store

(2) MD-SAL data store notifies NetVirt provider about “br-int”

(3) NetVirt provider writes pipeline processing flow to MD-SAL config data store

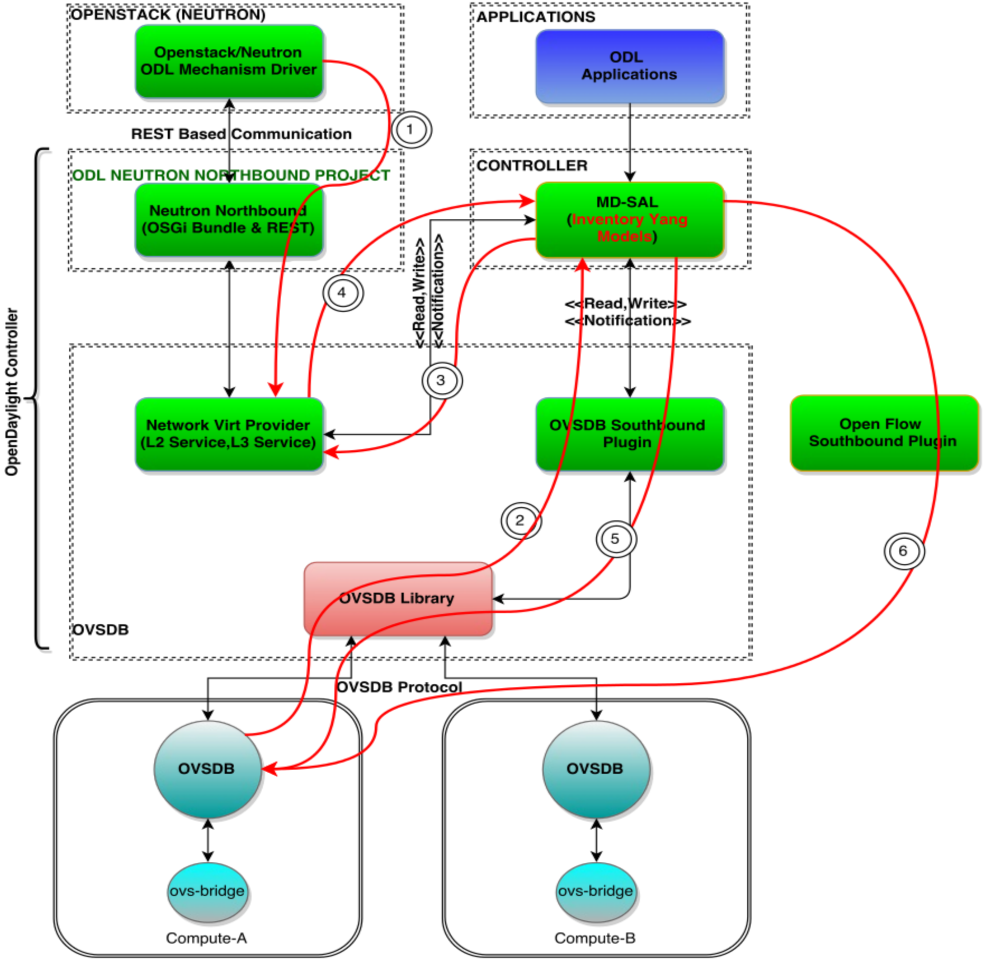
(4) OpenFlow Southbound plugin gets notification from MD-SAL data store about new flows added to config data store and it installs flow to “br-int”

# High Level Control Flow: *Programmed Flows – Pipeline processing*

```
Openstack-setup-compute# ovs-vsctl show
4575bb26-b73b-4e0a-a62a-9b3ff06e19af
    Manager "tcp:192.168.57.1:6640"
        is_connected: true
    Bridge br-int
        Controller "tcp:192.168.57.1:6653"
            is_connected: true
        fail_mode: secure
        Port br-int
            Interface br-int
    ovs_version: "2.0.2"
```

```
Openstack-setup-compute# ovs-ofctl dump-flows br-int -O OpenFlow13

 cookie=0x0, duration=23.662s, table=0, n_packets=0, n_bytes=0, dl_type=0x88cc
actions=CONTROLLER:65535

 cookie=0x0, duration=17.982s, table=0, n_packets=4, n_bytes=320, priority=0 actions=goto_table:20

 cookie=0x0, duration=17.474s, table=20, n_packets=1, n_bytes=70, priority=0 actions=goto_table:30

 cookie=0x0, duration=16.966s, table=30, n_packets=1, n_bytes=70, priority=0 actions=goto_table:40

 cookie=0x0, duration=16.449s, table=40, n_packets=1, n_bytes=70, priority=0 actions=goto_table:50

 cookie=0x0, duration=15.933s, table=50, n_packets=1, n_bytes=70, priority=0 actions=goto_table:60

 cookie=0x0, duration=15.417s, table=60, n_packets=1, n_bytes=70, priority=0 actions=goto_table:70

 cookie=0x0, duration=14.913s, table=70, n_packets=1, n_bytes=70, priority=0 actions=goto_table:80

 cookie=0x0, duration=14.404s, table=80, n_packets=1, n_bytes=70, priority=0 actions=goto_table:90

 cookie=0x0, duration=13.896s, table=90, n_packets=0, n_bytes=0, priority=0 actions=goto_table:100

 cookie=0x0, duration=13.387s, table=100, n_packets=0, n_bytes=0, priority=0 actions=goto_table:110

 cookie=0x0, duration=12.875s, table=110, n_packets=0, n_bytes=0, priority=0 actions=drop
```

# High Level Control Flow: *Create Network / Subnet / Port*



(1) OpenStack sends request for Network/Subnet/Port creation (for VM) to Neutron Northbound
   (a) NN passes it to NetVirt provider

(2) Spawning VM will create port on compute node and
   (a) that will trigger notification from ovsdb
   (b) OVSDB library will notify SB Plugin
   (c) SB Plugin will update the MD-SAL operational data store

(3) MD-SAL data store will notify NetVirt provider about new port creation

(4) NetVirt will write data into MD-SAL config data store for tunnel creation

(5) SB Plugin gets notification from MD-SAL data store about new tunnel data and it sends instructions to library for tunnel interface creation

(6) NetVirt also installs the required flows for VM traffic routing

# High Level Control Flow: *Bridge configuration changes*

```
Openstack-setup-compute# ovs-vsctl show
4575bb26-b73b-4e0a-a62a-9b3ff06e19af
    Manager "tcp:192.168.57.1:6640"
        is_connected: true
    Bridge br-int
        Controller "tcp:192.168.57.1:6633"
            is_connected: true
        fail_mode: secure
        Port br-int
            Interface br-int
        Port "vxlan-192.168.201.128"
            Interface "vxlan-192.168.201.128"
                type: vxlan
                options: {key=flow, local_ip="192.168.201.129", remote_ip="192.168.201.128"}
        Port "tap860039e7-9b"
            Interface "tap860039e7-9b"
    ovs_version: "2.0.2"
```

# High Level Control Flow: *Programmed Flows - L2 Routing (First VM Created)*

```
Openstack-setup-compute# ovs-ofctl dump-flows br-int -O OpenFlow13
table=0,   dl_type=0x88cc actions=CONTROLLER:65535
table=0,   priority=0 actions=goto_table:20
table=20,   priority=0 actions=goto_table:30
……
…...
table=90,   priority=0 actions=goto_table:100
table=100,   priority=0 actions=goto_table:110
table=110,  priority=0 actions=drop
table=110, tun_id=0x1,dl_dst=fa:16:3e:e5:e2:e1 actions=output:2 (Incoming traffic for VM)
table=0,  tun_id=0x1,in_port=1 actions=load:0x2->NXM_NX_REG0[],goto_table:20 (Other Incoming Traffic)
table=110, priority=16384,reg0=0x2,tun_id=0x1,dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=output:2 (If
 Multicast, send it VM port-- that's the only port related to network with vxlan-id = 0x1)
table=110, priority=8192,tun_id=0x1 actions=drop (Else drop it)

table=0,  in_port=2,dl_src=fa:16:3e:e5:e2:e1 actions=set_field:0x1->tun_id,load:0x1->NXM_NX_REG0[],goto_table:20
 (Mark outgoing VM Traffic)
table=110, priority=16383,reg0=0x1,tun_id=0x1,dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=output:2,output:1
 (If multicast, send it on all ports)
table=110, tun_id=0x1,dl_dst=fa:16:3e:e3:35:86 actions=output:1 (DHCP traffic of the network– send it out)
table=0,  priority=8192,in_port=2 actions=drop (Drop rest all traffic from VM)
```

# High Level Control Flow: *Programmed Flows - L2 Routing (Second VM Created)*

```
Openstack-setup-compute# ovs-ofctl dump-flows br-int -O OpenFlow13
table=0,   dl_type=0x88cc actions=CONTROLLER:65535
table=0,   priority=0 actions=goto_table:20
table=20,   priority=0 actions=goto_table:30
……
…...
table=90,   priority=0 actions=goto_table:100
table=100,   priority=0 actions=goto_table:110
table=110,  priority=0 actions=drop
table=110, tun_id=0x1,dl_dst=fa:16:3e:e5:e2:e1 actions=output:2 (Incoming traffic for VM)
table=0,  tun_id=0x1,in_port=1 actions=load:0x2->NXM_NX_REG0[],goto_table:20 (Other Incoming Traffic)
table=110, priority=16384,reg0=0x2,tun_id=0x1,dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=output:2 (If
 multicast, send it VM port-- that's the only port related to network with vxlan-id = 0x1)
table=110, priority=8192,tun_id=0x1 actions=drop (Else drop it)

table=0,  in_port=2,dl_src=fa:16:3e:e5:e2:e1 actions=set_field:0x1->tun_id,load:0x1->NXM_NX_REG0[],goto_table:20
 (Tag outgoing VM Traffic)
table=110, priority=16383,reg0=0x1,tun_id=0x1,dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=output:2,output:1
 (If multicast, sent it on all ports)
table=110, tun_id=0x1,dl_dst=fa:16:3e:e3:35:86 actions=output:1 (DHCP traffic of the network– send it out)
table=0,  priority=8192,in_port=2 actions=drop (Drop rest all traffic from VM)
table=110, tun_id=0x1,dl_dst=fa:16:3e:49:e9:5a actions=output:2 (VM1-->VM2)
```

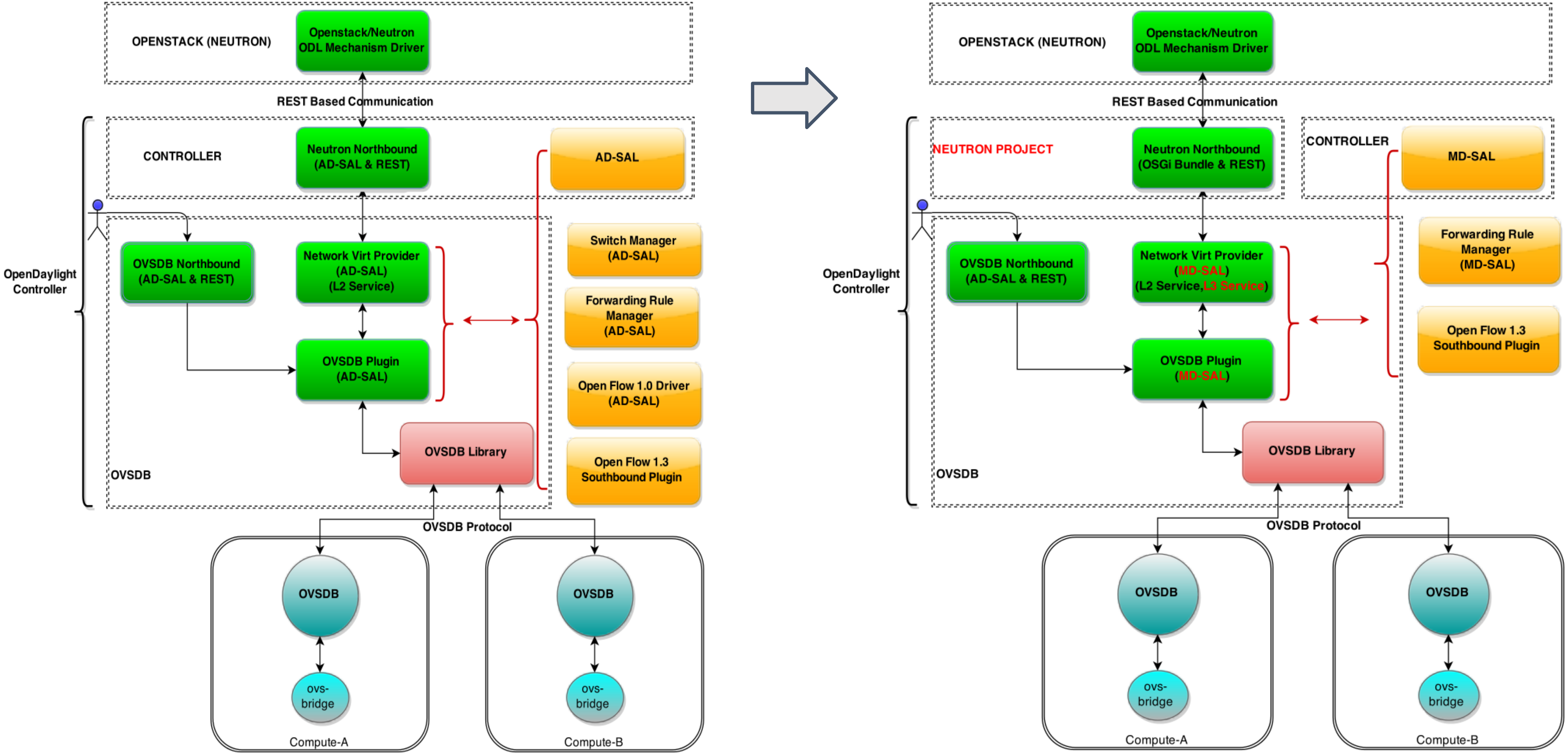
# ....will talk about:

- What the OVSDB Project offers?
- Why it's the Center of Attraction?
- Brief Overview of Open vSwitch & Management Protocol
- High Level Architecture and Control Flow
- What we have accomplished in Lithium
- What are we planning for Beryllium?
- Let’s ./stack!
- Looking to contribute?

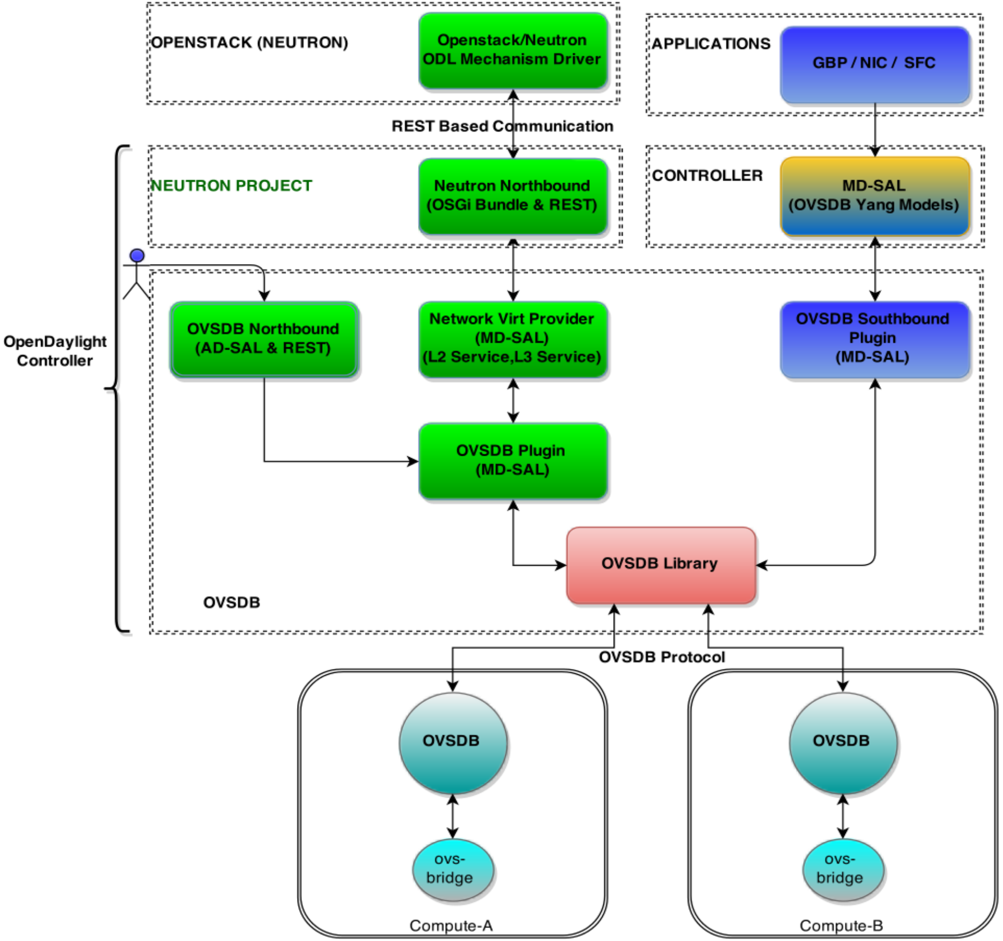
# What we accomplished in Lithium

- Migrated following AD-SAL based modules to MD-SAL
  - NetVirt provider
  - Plugin bundle
- Implemented Yang based Southbound Plugin module
- Migrated NetVirt provider from OVSDB plugin to new Yang based Southbound Plugin
- Implemented L3 Service
  - East-West Traffic Routing
  - North-South Traffic Routing
  - Floating IP/DNAT
- Implemented SAL compatibility layer to support backward compatibility for VTN project
- Improved unit and integration tests and code coverage
- Cleaned up stale code

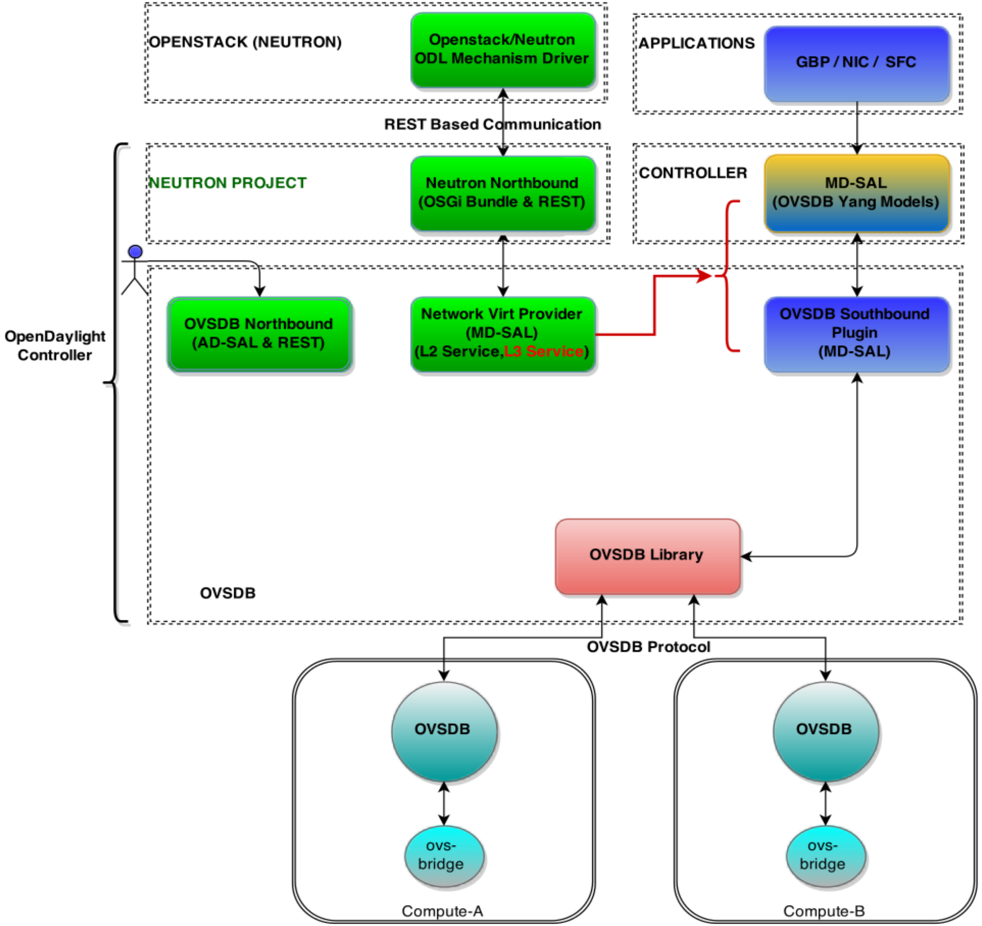
# Lithium: *Migration to MD-SAL & L3 Service*


OPENSTACK (NEUTRON)
Openstack/Neutron ODL Mechanism Driver
REST Based Communication
CONTROLLER
Neutron Northbound (AD-SAL & REST)
AD-SAL
OpenDaylight Controller
OVSDB Northbound (AD-SAL & REST)
Network Virt Provider (AD-SAL) (L2 Service)
OVSDB Plugin (AD-SAL)
OVSDB Library
OVSDB
Switch Manager (AD-SAL)
Forwarding Rule Manager (AD-SAL)
Open Flow 1.0 Driver (AD-SAL)
Open Flow 1.3 Southbound Plugin
OVSDB Protocol
OVSDB
ovs-bridge
Compute-A
OVSDB
ovs-bridge
Compute-B
OPENSTACK (NEUTRON)
Openstack/Neutron ODL Mechanism Driver
REST Based Communication
NEUTRON PROJECT
Neutron Northbound (OSGi Bundle & REST)
CONTROLLER
MD-SAL
OpenDaylight Controller
OVSDB Northbound (AD-SAL & REST)
Network Virt Provider (MD-SAL) (L2 Service,L3 Service)
OVSDB Plugin (MD-SAL)
OVSDB Library
OVSDB
Forwarding Rule Manager (MD-SAL)
Open Flow 1.3 Southbound Plugin
OVSDB Protocol
OVSDB
ovs-bridge
Compute-A
OVSDB
ovs-bridge
Compute-B

# Lithium: *Introduced OVSDB Southbound Plugin*


OPENSTACK (NEUTRON)
Openstack/Neutron ODL Mechanism Driver
APPLICATIONS
GBP / NIC / SFC
REST Based Communication
NEUTRON PROJECT
Neutron Northbound (OSGi Bundle & REST)
CONTROLLER
MD-SAL (OVSDB Yang Models)
OpenDaylight Controller
OVSDB Northbound (AD-SAL & REST)
Network Virt Provider (MD-SAL) (L2 Service,L3 Service)
OVSDB Southbound Plugin (MD-SAL)
OVSDB Plugin (MD-SAL)
OVSDB Library
OVSDB
OVSDB Protocol
OVSDB
ovs-bridge
Compute-A
OVSDB
ovs-bridge
Compute-B

# Lithium: *NetVirt Migration to OVSDB Southbound Plugin*


OPENSTACK (NEUTRON)
Openstack/Neutron ODL Mechanism Driver
APPLICATIONS
GBP / NIC / SFC
REST Based Communication
NEUTRON PROJECT
Neutron Northbound (OSGi Bundle & REST)
CONTROLLER
MD-SAL (OVSDB Yang Models)
OpenDaylight Controller
OVSDB Northbound (AD-SAL & REST)
Network Virt Provider (MD-SAL) (L2 Service,L3 Service)
OVSDB Southbound Plugin (MD-SAL)
OVSDB Library
OVSDB
OVSDB Protocol
OVSDB
ovs-bridge
Compute-A
OVSDB
ovs-bridge
Compute-B

# ....will talk about:

- What the OVSDB Project offers?
- Why it's the Center of Attraction?
- Brief Overview of Open vSwitch & Management Protocol
- High Level Architecture and Control Flow
- What we have accomplished in Lithium
- What are we planning for Beryllium?
- Let’s ./stack!
- Looking to contribute?

## What are we planning for Beryllium?

- Clustering support to provide HA, Scalability and Performance
- Continue to improve code quality and stability
- Increase testing coverage
- Improve documentation
- Add support for new OpenStack services
  - Complete Security Groups and LBaaS
  - Implement SNAT, IPv6 and FWaaS
  - SFC/NFV Integration
- Implement hardware vtep southbound plugin
- Implement support for hardware vtep L2 Gateway
- Migrate NetVirt to consume Neutron Yang Models
- Continue growing an open ecosystem
- Help people to come onboard and solve interesting network virtualization problems with us.

# ....will talk about:

- What the OVSDB Project offers?
- Why it's the Center of Attraction?
- Brief Overview of Open vSwitch & Management Protocol
- High Level Architecture and Control Flow
- What we have accomplished in Lithium
- What are we planning for Beryllium?
- Let’s ./stack!
- Looking to contribute?

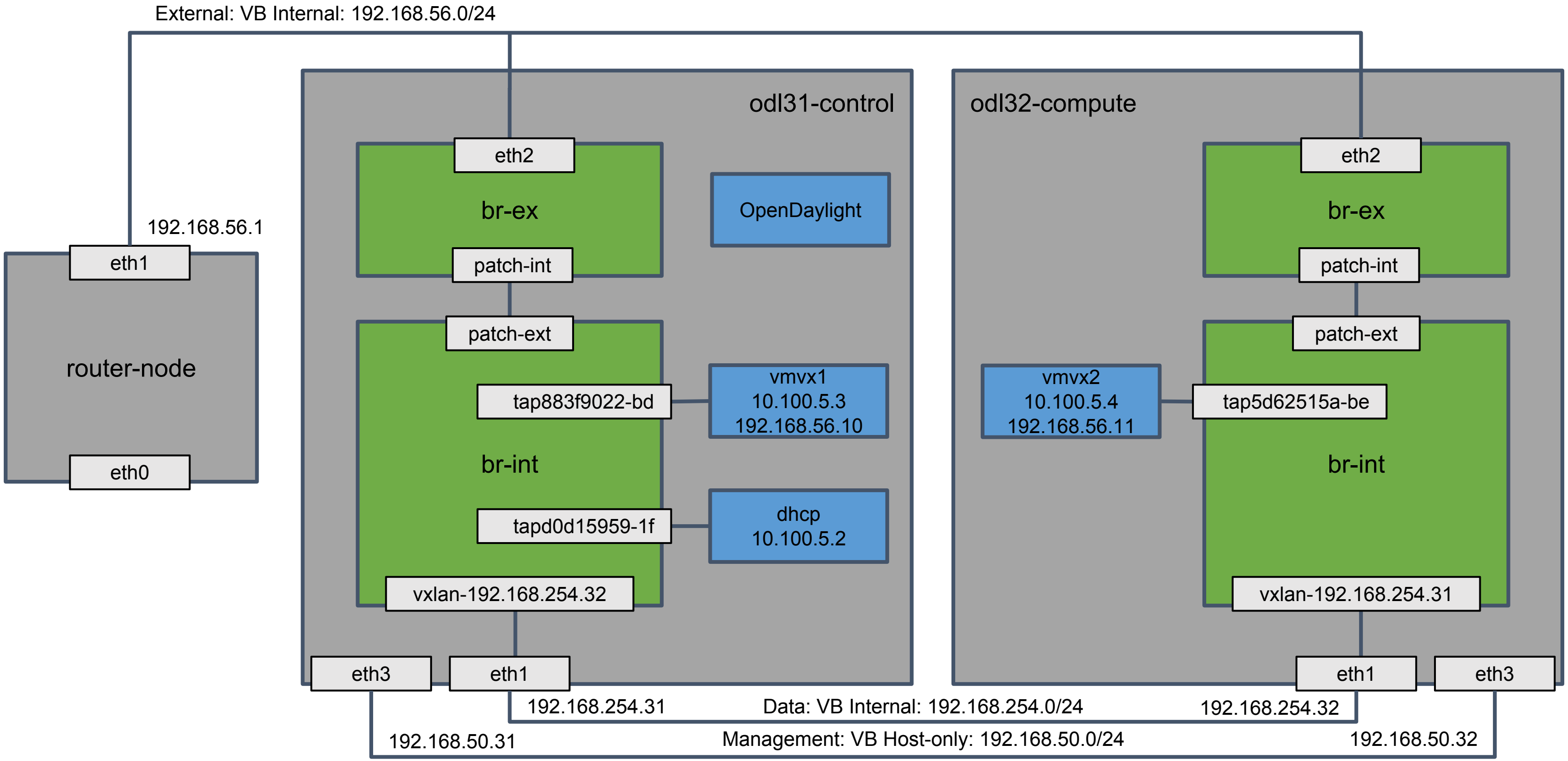
# Demo Description

- Demonstrate network virtualization using vxlan overlay, L3 and floating ip
- Three nodes in a single ova that can be consumed by vm players:
  - openstack control, compute, OpenDaylight, CentOS 7, devstack
  - openstack compute, CentOS 7, devstack
  - router for external access, CentOS 6.5

# Demo Steps: Import VMs and Start DevStack

1. Change the vboxnet0 IPv4 Address to 192.168.50.1. Find the setting at File->Preferences->Network->Host-only Networks
2. Import the OVA into VirtualBox
    a. Copy ovsdbtutorial15_2.ova to local system
    b. File->Import Appliance, Browse to ovsdbtutorial15_2.ova
    c. Do not select “Reinitialize the MAC address of all network cards”
    d. Import: odl31-compute, odl31-control and router-node will be imported
3. Start all three VMs via the VirtualBox interface
4. Log into the odl31-control node. ssh odl@192.168.50.31, pw: odl
5. Start devstack
    a. cd /opt/devstack
    b. ./stack.sh
6. Repeat 4 and 5 to start devstack on odl32-compute, ssh odl@192.168.50.32, pw: odl

# Topology


External: VB Internal: 192.168.56.0/24
192.168.56.1
eth1
router-node
eth0
odl31-control
eth2
br-ex
patch-int
OpenDaylight
patch-ext
tap883f9022-bd
vmvx1
10.100.5.3
192.168.56.10
br-int
tapd0d15959-1f
dhcp
10.100.5.2
vxlan-192.168.254.32
eth3
eth1
odl32-compute
eth2
br-ex
patch-int
patch-ext
vmvx2
10.100.5.4
192.168.56.11
tap5d62515a-be
br-int
vxlan-192.168.254.31
eth1
eth3
192.168.254.31
Data: VB Internal: 192.168.254.0/24
192.168.254.32
192.168.50.31
Management: VB Host-only: 192.168.50.0/24
192.168.50.32

# Topology Details

- eth0: management, requires adding VB port-forwarding to reach from host
- eth1: internal data network for tenant traffic
- eth2: external network for floating-ip’s - note this is eth1 for the router-node
- eth3: management, reachable from host via the vboxnet0 Host-only Network

| **VM** | **Services** | **eth0**<br>**VB NAT** | **eth1**<br>**VB Internal 1** | **eth2**<br>**VB Internal 2** | **eth3**<br>**VB vboxnet0** |
|---|---|---|---|---|---|
| odl31-control | control, ODL | 10.0.2.15 | 192.168.254.30 | 0.0.0.0 | 192.168.50.31 |
| odl32-compute | compute | 10.0.2.15 | 192.168.254.31 | 0.0.0.0 | 192.168.50.32 |
| router-node | router, DHCP | 10.0.2.15 | 192.168.56.1<br>**VB internal 2** | NA | NA |

# Topology Mappings

| Description | MAC Address | IP Address | Floating-IP<br>MAC Address | Port |
|---|---|---|---|---|
| vx-net gw internal | fa:16:3e:30:19:de | 10.100.5.1 | | |
| vx-net dhcp | fa:16:3e:9f:82:6c | 10.100.5.2 | | 1 |
| vmvx1 | fa:16:3e:13:44:69 | 10.100.5.3 | 192.168.56.10<br>fa:16:3e:84:87:1a | 4 |
| vmvx2 | fa:16:3e:ce:d7:ad | 10.100.5.4 | 192.168.56.11<br>fa:16:3e:2e:ee:39 | |
| patch-ext | 72:48:60:5e:44:7b | | | 2 |
| vxlan-192.168.254.32 | 6a:6c:f2:ef:f5:d7 | | | 3 |

# Neutron Commands (1 of 2)

**source openrc admin admin**

**os_addnano.sh:**

```
nova flavor-create m1.nano auto 64 0 1
```

**os_addadminkey.sh:**

```
nova keypair-add --pub-key ~/.ssh/id_rsa.pub admin_key
```

**os_addextnetrtr.sh:**

```
neutron net-create ext-net --router:external --provider:physical_network public --provider:
 network_type flat
neutron subnet-create --name ext-subnet --allocation-pool start=192.168.56.9,end=192.168.56.14 --
 disable-dhcp --gateway 192.168.56.1 ext-net 192.168.56.0/24

neutron net-create vx-net --provider:network_type vxlan --provider:segmentation_id 1500
neutron subnet-create vx-net 10.100.5.0/24 --name vx-subnet --dns-nameserver 8.8.8.8

neutron router-create ext-rtr
neutron router-gateway-set ext-rtr ext-net
neutron router-interface-add ext-rtr vx-subnet
```

# Neutron Commands (2 of 2)

**os_addvms.sh:**

```
nova boot --poll --flavor m1.nano --image $(nova image-list | grep 'uec\s' | awk '{print $2}' | tail -1) --
 nic net-id=$(neutron net-list | grep -w vx-net | awk '{print $2}') vmvx1 --availability_zone=nova:odl31
 --key_name admin_key

nova boot --poll --flavor m1.nano --image $(nova image-list | grep 'uec\s' | awk '{print $2}' | tail -1) --
 nic net-id=$(neutron net-list | grep -w vx-net | awk '{print $2}') vmvx2 --availability_zone=nova:odl32
 --key_name admin_key
```

**os_addfloatingips.sh:**

```
for vm in vmvx1 vmvx2; do
    vm_id=$(nova list | grep $vm | awk '{print $2}')
    port_id=$(neutron port-list -c id -c fixed_ips -- --device_id $vm_id | grep subnet_id | awk '{print $2}')
    neutron floatingip-create --port_id $port_id ext-net
done;
```

# DevStack local.conf ODL_MODE for networking-odl

https://github.com/flavio-fernandes/networking-odl/blob/heliumkilo/devstack/settings#L27

```
ODL_MODE=${ODL_MODE:-allinone}
# ODL_MODE is used to configure how devstack works with OpenDaylight. You
# can configure this three ways:
# ODL_MODE=allinone
# Use this mode if you want to run ODL in this devstack instance. Useful
# for a single node deployment or on the control node of a multi-node
# devstack environment.
# ODL_MODE=compute
# Use this for the compute nodes of a multi-node devstack install.
# ODL_MODE=externalodl
# This installs the neutron code for ODL, but does not attempt to
# manage ODL in devstack. This is used for development environments
# similar to the allinone case except where you are using bleeding edge ODL
# which is not yet released, and thus don't want it managed by
# devstack.
# ODL_MODE=manual
# You're on your own here, and are enabling services outside the scope of
# the ODL_MODE variable.
```

# odl31-control local.conf

```
disable_all_services
enable_service g-api g-reg key n-api n-crt n-obj n-cpu n-cond n-sch n-novnc n-xvnc n-cauth horizon neutron q-dhcp q-meta q-svc mysql rabbit
enable_service odl-server odl-compute
…
HOST_IP=192.168.254.31
HOST_NAME=odl31
…
enable_plugin networking-odl https://github.com/flavio-fernandes/networking-odl summit15demo
ODL_MODE=manual
NEUTRON_CREATE_INITIAL_NETWORKS=False
ODL_L3=True
PUBLIC_INTERFACE=eth2
```

# odl32-compute local.conf

```
disable_all_services
enable_service n-cpu n-novnc neutron rabbit
enable_service odl-compute
…
HOST_IP=192.168.254.32
HOST_NAME=odl32
SERVICE_HOST_NAME=odl31
SERVICE_HOST=192.168.254.31
Q_HOST=$SERVICE_HOST
…
ODL_MODE=manual
ODL_L3=True
PUBLIC_INTERFACE=eth2
```

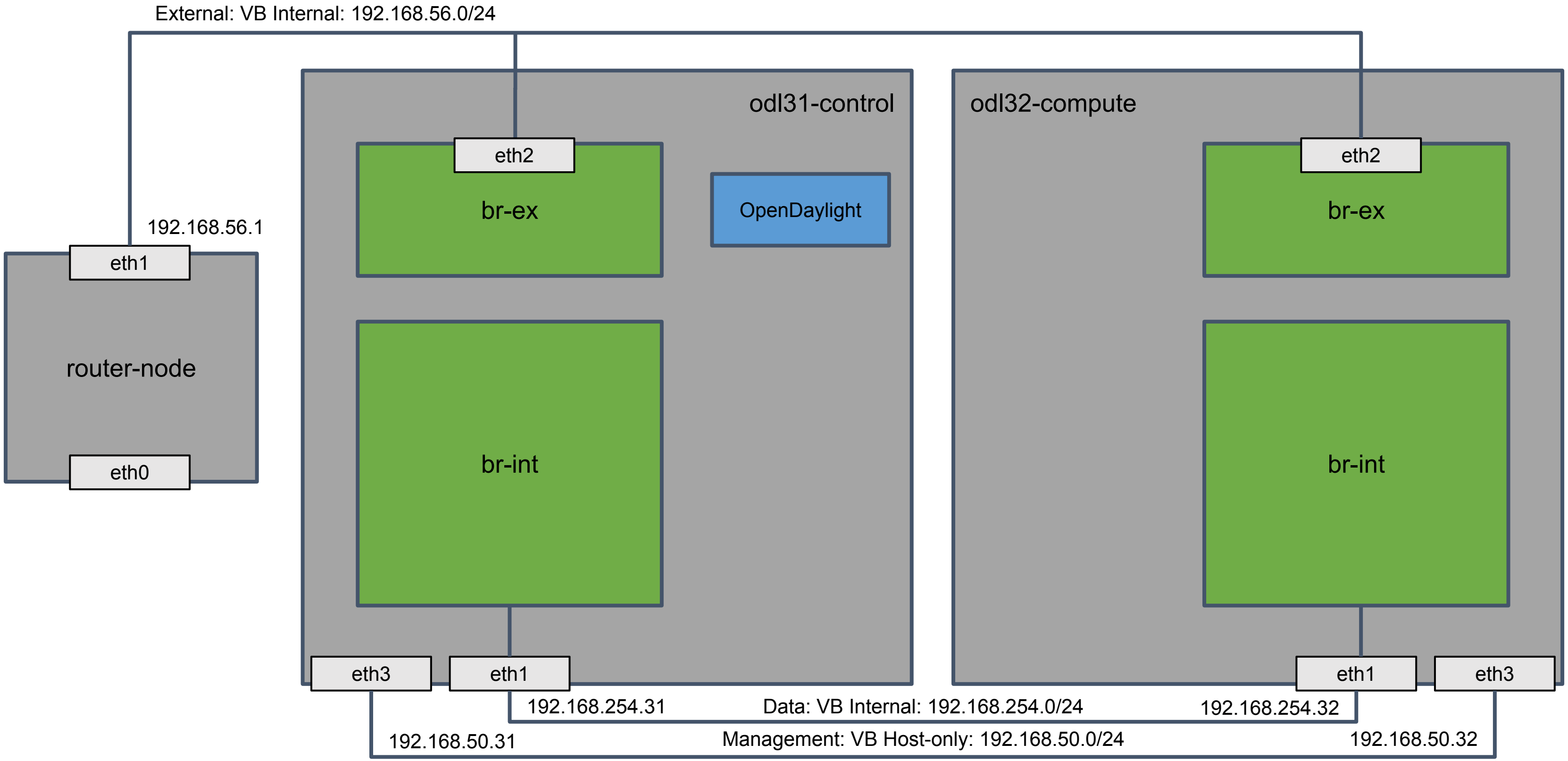
# Demo Steps: Create Networks, L3 and Floating IPs

Individual steps:

1. source openrc admin admin
2. ../tools/os_addnano.sh: add a nano flavor of the vms
3. ../tools/os_addadminkey.sh: add ssh keys to have password-less logins to the tenant vms
4. ../tools/os_addextnetrtr.sh: add external and vxlan networks and attach to router
5. ../tools/os_addvms: launch two vms, one on each compute node
6. ../tools/os_addfloatingip.sh: assign floating ip’s to each vm

Or just use ../tools/doitall.sh: But it’s more fun to do each step and see what happens...

# Topology: After Stacking



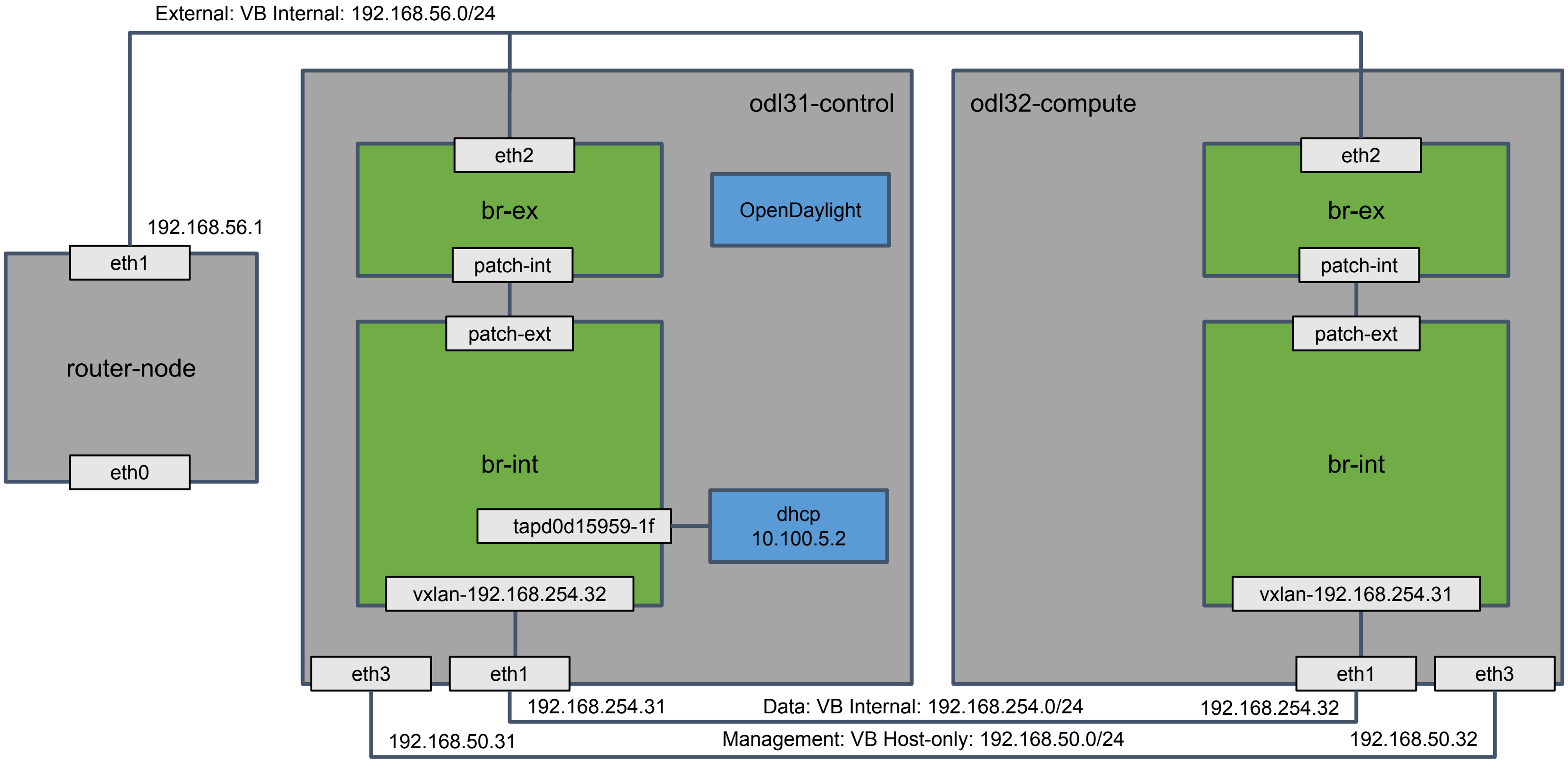
# OVSDB: After Stacking

```
sudo ovs-vsctl show
d9904cbd-34c7-48e2-b714-fb5d04a4d899
    Manager "tcp:192.168.254.31:6640"
        is_connected: true
    Bridge br-ex
        Controller "tcp:192.168.254.31:6653"
            is_connected: true
        fail_mode: secure
        Port br-ex
            Interface br-ex
                type: internal
        Port "eth2"
            Interface "eth2"
    Bridge br-int
        Controller "tcp:192.168.254.31:6653"
            is_connected: true
        fail_mode: secure
        Port br-int
            Interface br-int
                type: internal
```

# Flows: After Stacking

```
sudo ovs-ofctl --protocol=OpenFlow13 dump-flows br-ex
 cookie=0x0, duration=49.967s, table=0, n_packets=0, n_bytes=0, priority=0 actions=NORMAL
 cookie=0x0, duration=49.967s, table=0, n_packets=4, n_bytes=452, dl_type=0x88cc actions=CONTROLLER:65535

sudo ovs-ofctl --protocol=OpenFlow13 dump-flows br-int
 cookie=0x0, duration=49.482s, table=0, n_packets=0, n_bytes=0, priority=0 actions=goto_table:20
 cookie=0x0, duration=49.998s, table=0, n_packets=0, n_bytes=0, dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=49.472s, table=20, n_packets=0, n_bytes=0, priority=0 actions=goto_table:30
 cookie=0x0, duration=49.466s, table=30, n_packets=0, n_bytes=0, priority=0 actions=goto_table:40
 cookie=0x0, duration=49.456s, table=40, n_packets=0, n_bytes=0, priority=0 actions=goto_table:50
 cookie=0x0, duration=49.446s, table=50, n_packets=0, n_bytes=0, priority=0 actions=goto_table:60
 cookie=0x0, duration=49.435s, table=60, n_packets=0, n_bytes=0, priority=0 actions=goto_table:70
 cookie=0x0, duration=49.424s, table=70, n_packets=0, n_bytes=0, priority=0 actions=goto_table:80
 cookie=0x0, duration=49.407s, table=80, n_packets=0, n_bytes=0, priority=0 actions=goto_table:90
 cookie=0x0, duration=49.403s, table=90, n_packets=0, n_bytes=0, priority=0 actions=goto_table:100
 cookie=0x0, duration=49.391s, table=100, n_packets=0, n_bytes=0, priority=0 actions=goto_table:110
 cookie=0x0, duration=49.366s, table=110, n_packets=0, n_bytes=0, priority=0 actions=drop
```

# Topology: After Adding Neutron Networks and Router



# OVSDB: After Adding Neutron Networks and Router

```
sudo ovs-vsctl show
d9904cbd-34c7-48e2-b714-fb5d04a4d899
    Manager "tcp:192.168.254.31:6640"
        is_connected: true
    Bridge br-ex
        Controller "tcp:192.168.254.31:6653"
            is_connected: true
        fail_mode: secure
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-ext}
        Port br-ex
            Interface br-ex
                type: internal
        Port "eth2"
            Interface "eth2"
Bridge br-int
        Controller "tcp:192.168.254.31:6653"
            is_connected: true
        fail_mode: secure
        Port br-int
            Interface br-int
                type: internal
        Port patch-ext
            Interface patch-ext
                type: patch
                options: {peer=patch-int}
        Port "tapd0d15959-1f"
            Interface "tapd0d15959-1f"
                type: internal
        Port "vxlan-192.168.254.32"
            Interface "vxlan-192.168.254.32"
                type: vxlan
                options: {key=flow, local_ip="
192.168.254.31", remote_ip="
192.168.254.32"}
    ovs_version: "2.3.1"
```

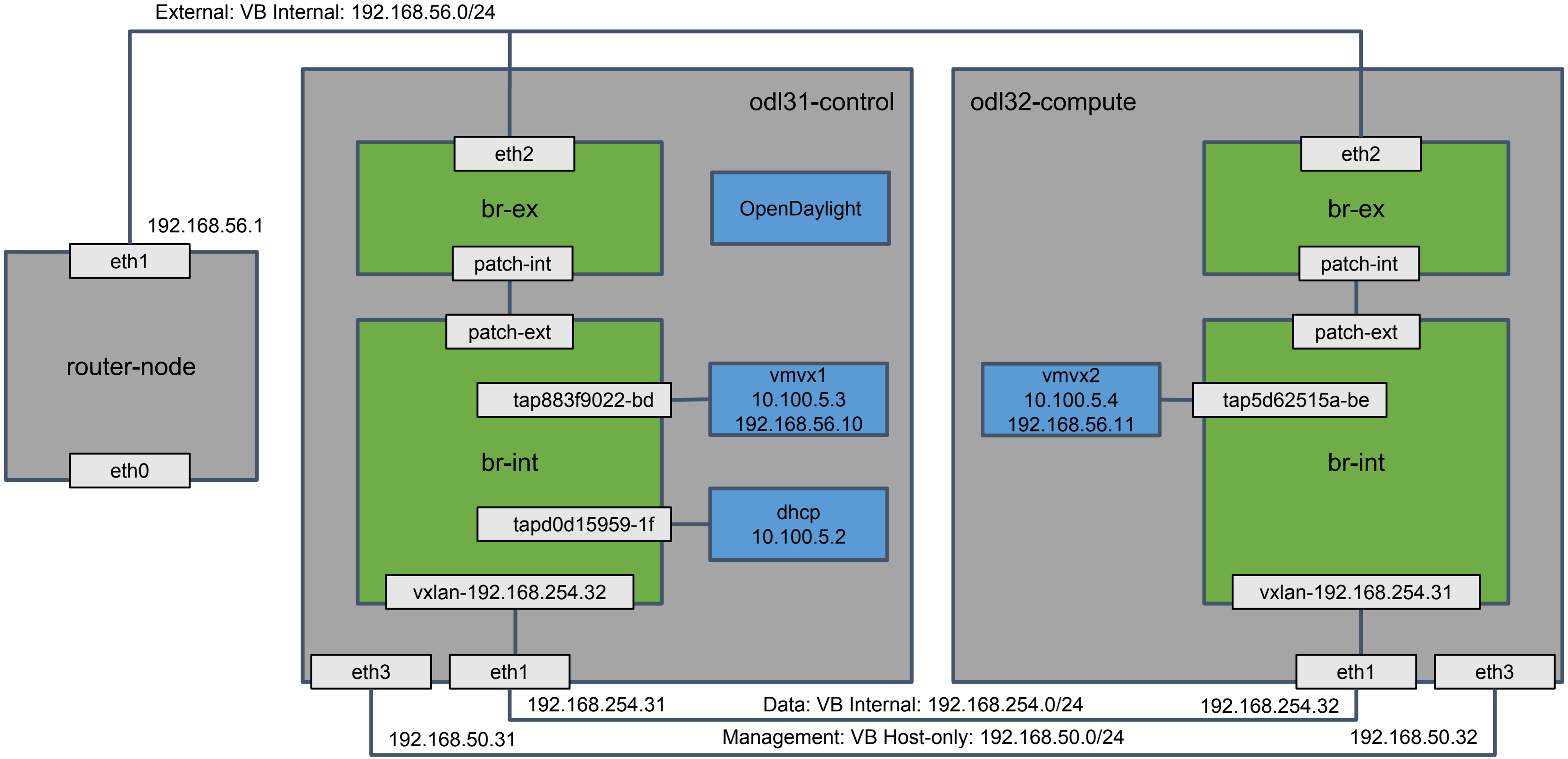
# Flows: After Adding Neutron Networks and Router (1 of 2)

```
sudo ovs-ofctl --protocol=OpenFlow13 dump-flows br-int
 cookie=0x0, duration=35.009s, table=0, n_packets=7, n_bytes=558, in_port=1,dl_src=fa:16:3e:9f:82:6c actions=set_field:0x5dc-
 >tun_id,load:0x1->NXM_NX_REG0[],goto_table:20 (DHCP port ingress)
 cookie=0x0, duration=179.731s, table=0, n_packets=1, n_bytes=90, priority=0 actions=goto_table:20 (pipeline)
 cookie=0x0, duration=35.011s, table=0, n_packets=0, n_bytes=0, priority=8192,in_port=1 actions=drop (drop everything else)
 cookie=0x0, duration=34.793s, table=0, n_packets=0, n_bytes=0, tun_id=0x5dc,in_port=3 actions=load:0x2->NXM_NX_REG0
 [],goto_table:20 (tunnel ingress)
 cookie=0x0, duration=180.247s, table=0, n_packets=16, n_bytes=1808, dl_type=0x88cc actions=CONTROLLER:65535 (LLDP
 punt)
 cookie=0x0, duration=179.721s, table=20, n_packets=8, n_bytes=648, priority=0 actions=goto_table:30 (pipeline)
 cookie=0x0, duration=29.644s, table=20, n_packets=0, n_bytes=0, priority=1024,arp,tun_id=0x5dc,arp_tpa=10.100.5.1
 actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],set_field:fa:16:3e:30:19:de->eth_src,load:0x2-
 >NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],move:NXM_OF_ARP_SPA[]-
 >NXM_OF_ARP_TPA[],load:0xfa163e3019de->NXM_NX_ARP_SHA[],load:0xa640501->NXM_OF_ARP_SPA[],IN_PORT (ARP
 response for vxnet gw)
 cookie=0x0, duration=29.574s, table=20, n_packets=0, n_bytes=0, priority=1024,arp,tun_id=0x5dc,arp_tpa=10.100.5.2
 actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],set_field:fa:16:3e:9f:82:6c->eth_src,load:0x2-
 >NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],move:NXM_OF_ARP_SPA[]-
 >NXM_OF_ARP_TPA[],load:0xfa163e9f826c->NXM_NX_ARP_SHA[],load:0xa640502->NXM_OF_ARP_SPA[],IN_PORT (ARP
 response for vxnet DHCP namespace)
 cookie=0x0, duration=179.715s, table=30, n_packets=8, n_bytes=648, priority=0 actions=goto_table:40 (pipeline)
 cookie=0x0, duration=179.705s, table=40, n_packets=8, n_bytes=648, priority=0 actions=goto_table:50 (pipeline)
 cookie=0x0, duration=35.165s, table=40, n_packets=0, n_bytes=0, priority=61012,udp,tp_src=68,tp_dst=67 actions=goto_table:
 50 (allow DHCP)
 cookie=0x0, duration=179.695s, table=50, n_packets=8, n_bytes=648, priority=0 actions=goto_table:60 (pipeline)
```

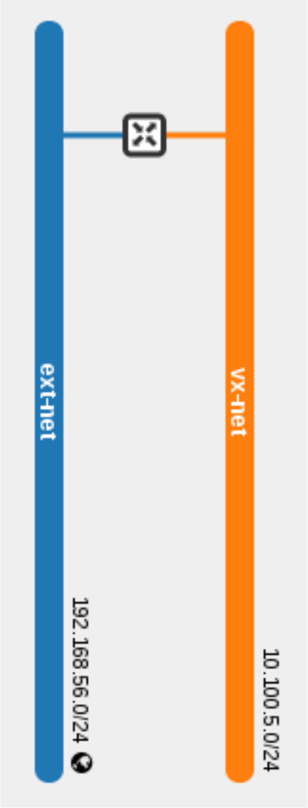
# Flows: After Adding Neutron Networks and Router (2 of 2)

```
cookie=0x0, duration=179.684s, table=60, n_packets=8, n_bytes=648, priority=0 actions=goto_table:70 (pipeline)
cookie=0x0, duration=29.657s, table=60, n_packets=0, n_bytes=0, priority=2048,ip,reg3=0x5dc,nw_dst=10.100.5.0/24 actions=set_field:fa:16:3e:30:19:de->eth_src,dec_ttl,set_field:0x5dc->tun_id,goto_table:70 (l3 src mac of tenant router)
cookie=0x0, duration=179.673s, table=70, n_packets=8, n_bytes=648, priority=0 actions=goto_table:80 (pipeline)
cookie=0x0, duration=29.578s, table=70, n_packets=0, n_bytes=0, priority=1024,ip,tun_id=0x5dc,nw_dst=10.100.5.2 actions=set_field:fa:16:3e:9f:82:6c->eth_dst,goto_table:80 (l3 forward to DHCP)
cookie=0x0, duration=179.656s, table=80, n_packets=8, n_bytes=648, priority=0 actions=goto_table:90 (pipeline)
cookie=0x0, duration=179.652s, table=90, n_packets=8, n_bytes=648, priority=0 actions=goto_table:100 (pipeline)
cookie=0x0, duration=179.640s, table=100, n_packets=8, n_bytes=648, priority=0 actions=goto_table:110 (pipeline)
cookie=0x0, duration=29.631s, table=100, n_packets=0, n_bytes=0, priority=1024,ip,tun_id=0x5dc,nw_dst=10.100.5.0/24 actions=goto_table:110 (allow subnet destined traffic)
cookie=0x0, duration=34.801s, table=110, n_packets=0, n_bytes=0, priority=8192,tun_id=0x5dc actions=drop (pipeline)
cookie=0x0, duration=179.615s, table=110, n_packets=1, n_bytes=90, priority=0 actions=drop (pipeline)
cookie=0x0, duration=34.848s, table=110, n_packets=0, n_bytes=0, priority=16384,reg0=0x2,tun_id=0x5dc, dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=output:1 ({multi,broad}cast tunnel ingress)
cookie=0x0, duration=34.830s, table=110, n_packets=7, n_bytes=558, priority=16383,reg0=0x1,tun_id=0x5dc, dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=output:1,output:3 ({multi,broad}cast)
cookie=0x0, duration=34.998s, table=110, n_packets=0, n_bytes=0, tun_id=0x5dc,dl_dst=fa:16:3e:9f:82:6c actions=output:1 (l2 forward to DHCP port)
```

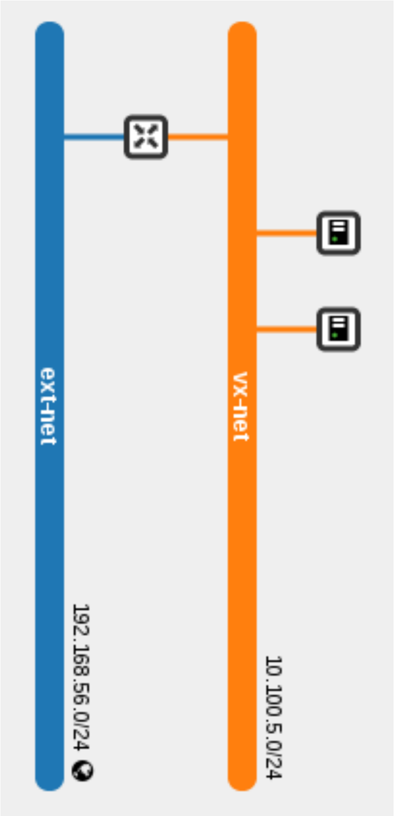
# Topology: After Adding VMs



# OpenStack Network Dashboard



External and VxLAN networks created



Tenant VMs created

# OVSDB: After Adding VMs

```
sudo ovs-vsctl show
d9904cbd-34c7-48e2-b714-fb5d04a4d899
    Manager "tcp:192.168.254.31:6640"
        is_connected: true
    Bridge br-ex
        Controller "tcp:192.168.254.31:6653"
            is_connected: true
        fail_mode: secure
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-ext}
        Port br-ex
            Interface br-ex
                type: internal
        Port "eth2"
            Interface "eth2"
    Bridge br-int
        Controller "tcp:192.168.254.31:6653"
            is_connected: true
        fail_mode: secure
        Port "tap883f9022-bd"
            Interface "tap883f9022-bd"
        Port br-int
            Interface br-int
                type: internal
        Port patch-ext
            Interface patch-ext
                type: patch
                options: {peer=patch-int}
        Port "tapd0d15959-1f"
            Interface "tapd0d15959-1f"
                type: internal
        Port "vxlan-192.168.254.32"
            Interface "vxlan-192.168.254.32"
                type: vxlan
                options: {key=flow, local_ip="
192.168.254.31", remote_ip="192.168.254.32"}
    ovs_version: "2.3.1"
```

# Flows: On odl31-control After Adding VMs (1 of 3)

```
sudo ovs-ofctl --protocol=OpenFlow13 dump-flows br-int
cookie=0x0, duration=230.486s, table=0, n_packets=13, n_bytes=2076, in_port=1,dl_src=fa:16:3e:9f:82:6c actions=set_field:
0x5dc->tun_id,load:0x1->NXM_NX_REG0[],goto_table:20
cookie=0x0, duration=35.882s, table=0, n_packets=23, n_bytes=2504, in_port=4,dl_src=fa:16:3e:13:44:69 actions=set_field:
0x5dc->tun_id,load:0x1->NXM_NX_REG0[],goto_table:20 (VM port ingress)
cookie=0x0, duration=375.208s, table=0, n_packets=1, n_bytes=90, priority=0 actions=goto_table:20
cookie=0x0, duration=230.488s, table=0, n_packets=0, n_bytes=0, priority=8192,in_port=1 actions=drop
cookie=0x0, duration=35.876s, table=0, n_packets=0, n_bytes=0, priority=8192,in_port=4 actions=drop
cookie=0x0, duration=230.270s, table=0, n_packets=8, n_bytes=1142, tun_id=0x5dc,in_port=3 actions=load:0x2-
>NXM_NX_REG0[],goto_table:20
cookie=0x0, duration=375.724s, table=0, n_packets=94, n_bytes=10622, dl_type=0x88cc actions=CONTROLLER:65535
cookie=0x0, duration=375.198s, table=20, n_packets=42, n_bytes=5686, priority=0 actions=goto_table:30
cookie=0x0, duration=36.659s, table=20, n_packets=1, n_bytes=42, priority=1024,arp,tun_id=0x5dc,arp_tpa=10.100.5.3
actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],set_field:fa:16:3e:13:44:69->eth_src,load:0x2-
>NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],move:NXM_OF_ARP_SPA[]-
>NXM_OF_ARP_TPA[],load:0xfa163e134469->NXM_NX_ARP_SHA[],load:0xa640503->NXM_OF_ARP_SPA[],IN_PORT (ARP
response for vmvx1 on odl31-control)
cookie=0x0, duration=225.121s, table=20, n_packets=1, n_bytes=42, priority=1024,arp,tun_id=0x5dc,arp_tpa=10.100.5.1
actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],set_field:fa:16:3e:30:19:de->eth_src,load:0x2-
>NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],move:NXM_OF_ARP_SPA[]-
>NXM_OF_ARP_TPA[],load:0xfa163e3019de->NXM_NX_ARP_SHA[],load:0xa640501->NXM_OF_ARP_SPA[],IN_PORT
cookie=0x0, duration=22.664s, table=20, n_packets=1, n_bytes=42, priority=1024,arp,tun_id=0x5dc,arp_tpa=10.100.5.4
actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],set_field:fa:16:3e:ce:d7:ad->eth_src,load:0x2-
>NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],move:NXM_OF_ARP_SPA[]-
>NXM_OF_ARP_TPA[],load:0xfa163eced7ad->NXM_NX_ARP_SHA[],load:0xa640504->NXM_OF_ARP_SPA[],IN_PORT (ARP
response for vmvx2 on odl32-compute)
```

# Flows: On odl31-control After Adding VMs (2 of 3)

```
cookie=0x0, duration=225.051s, table=20, n_packets=0, n_bytes=0, priority=1024,arp,tun_id=0x5dc,arp_tpa=10.100.5.2
actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],set_field:fa:16:3e:9f:82:6c->eth_src,load:0x2-
>NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],move:NXM_OF_ARP_SPA[]-
>NXM_OF_ARP_TPA[],load:0xfa163e9f826c->NXM_NX_ARP_SHA[],load:0xa640502->NXM_OF_ARP_SPA[],IN_PORT
cookie=0x0, duration=375.192s, table=30, n_packets=42, n_bytes=5686, priority=0 actions=goto_table:40
cookie=0x0, duration=35.889s, table=40, n_packets=14, n_bytes=1320, priority=36001,ip,in_port=4,dl_src=fa:16:3e:13:44:69,
nw_src=10.100.5.3 actions=goto_table:50 (allow vmvx1)
cookie=0x0, duration=375.182s, table=40, n_packets=24, n_bytes=3018, priority=0 actions=goto_table:50
cookie=0x0, duration=230.642s, table=40, n_packets=4, n_bytes=1348, priority=61012,udp,tp_src=68,tp_dst=67
actions=goto_table:50
cookie=0x0, duration=35.896s, table=40, n_packets=0, n_bytes=0, priority=61011,udp,in_port=4,tp_src=67,tp_dst=68
actions=drop
cookie=0x0, duration=375.172s, table=50, n_packets=42, n_bytes=5686, priority=0 actions=goto_table:60
cookie=0x0, duration=375.161s, table=60, n_packets=42, n_bytes=5686, priority=0 actions=goto_table:70
cookie=0x0, duration=225.134s, table=60, n_packets=0, n_bytes=0, priority=2048,ip,reg3=0x5dc,nw_dst=10.100.5.0/24
actions=set_field:fa:16:3e:30:19:de->eth_src,dec_ttl,set_field:0x5dc->tun_id,goto_table:70
cookie=0x0, duration=375.150s, table=70, n_packets=38, n_bytes=4252, priority=0 actions=goto_table:80
cookie=0x0, duration=22.679s, table=70, n_packets=2, n_bytes=717, priority=1024,ip,tun_id=0x5dc,nw_dst=10.100.5.4
actions=set_field:fa:16:3e:ce:d7:ad->eth_dst,goto_table:80 (l3 forward to vmvx2)
cookie=0x0, duration=225.055s, table=70, n_packets=0, n_bytes=0, priority=1024,ip,tun_id=0x5dc,nw_dst=10.100.5.2
actions=set_field:fa:16:3e:9f:82:6c->eth_dst,goto_table:80
cookie=0x0, duration=36.681s, table=70, n_packets=2, n_bytes=717, priority=1024,ip,tun_id=0x5dc,nw_dst=10.100.5.3
actions=set_field:fa:16:3e:13:44:69->eth_dst,goto_table:80 (l3 forward to vmvx1)
```

# Flows: On odl31-control After Adding VMs (3 of 3)

```
cookie=0x0, duration=375.133s, table=80, n_packets=42, n_bytes=5686, priority=0 actions=goto_table:90
cookie=0x0, duration=375.129s, table=90, n_packets=38, n_bytes=4252, priority=0 actions=goto_table:100
cookie=0x0, duration=35.904s, table=90, n_packets=4, n_bytes=1434, priority=61006,udp,dl_src=fa:16:3e:9f:82:6c,
tp_src=67,tp_dst=68 actions=goto_table:100
cookie=0x0, duration=375.117s, table=100, n_packets=32, n_bytes=3664, priority=0 actions=goto_table:110
cookie=0x0, duration=225.108s, table=100, n_packets=10, n_bytes=2022, priority=1024,ip,tun_id=0x5dc,nw_dst=10.
100.5.0/24 actions=goto_table:110
cookie=0x0, duration=230.278s, table=110, n_packets=14, n_bytes=1320, priority=8192,tun_id=0x5dc actions=drop
cookie=0x0, duration=375.092s, table=110, n_packets=1, n_bytes=90, priority=0 actions=drop
cookie=0x0, duration=230.325s, table=110, n_packets=8, n_bytes=1142, priority=16384,reg0=0x2,tun_id=0x5dc,
dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=output:1,output:4
cookie=0x0, duration=230.307s, table=110, n_packets=15, n_bytes=1700, priority=16383,reg0=0x1,tun_id=0x5dc,
dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 actions=output:1,output:3,output:4
cookie=0x0, duration=21.534s, table=110, n_packets=2, n_bytes=717, tun_id=0x5dc,dl_dst=fa:16:3e:ce:d7:ad
actions=output:3 (l2 forward to tunnel for vmvx2)
cookie=0x0, duration=230.475s, table=110, n_packets=0, n_bytes=0, tun_id=0x5dc,dl_dst=fa:16:3e:9f:82:6c
actions=output:1
cookie=0x0, duration=35.868s, table=110, n_packets=2, n_bytes=717, tun_id=0x5dc,dl_dst=fa:16:3e:13:44:69
actions=output:4 (l2 forward to vmvx1 port)
```

# Flows: On odl31-control After Adding Floating-IPs

```
sudo ovs-ofctl --protocol=OpenFlow13 dump-flows br-int
 ...
 cookie=0x0, duration=17.988s, table=20, n_packets=0, n_bytes=0, priority=1024,arp,in_port=2,
 arp_tpa=192.168.56.10 actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],set_field:fa:
 16:3e:84:87:1a->eth_src,load:0x2->NXM_OF_ARP_OP[],move:NXM_NX_ARP_SHA[]-
 >NXM_NX_ARP_THA[],move:NXM_OF_ARP_SPA[]->NXM_OF_ARP_TPA[],load:0xfa163e84871a-
 >NXM_NX_ARP_SHA[],load:0xc0a8380a->NXM_OF_ARP_SPA[],IN_PORT (ARP response for
 floating-ip of vmvx1)
 ...
 cookie=0x0, duration=17.943s, table=30, n_packets=0, n_bytes=0, priority=1024,ip,in_port=2,
 nw_dst=192.168.56.10 actions=set_field:10.100.5.3->ip_dst,load:0x5dc->NXM_NX_REG3[],
 goto_table:40 (NAT rewrite for floating-ip to vmvx1)
 …
 cookie=0x0, duration=17.920s, table=100, n_packets=0, n_bytes=0, priority=512,ip,tun_id=0x5dc,
 dl_dst=fa:16:3e:30:19:de,nw_src=10.100.5.3 actions=set_field:fa:16:3e:84:87:1a->eth_src,dec_ttl,
 set_field:52:54:00:34:10:b5->eth_dst,set_field:192.168.56.10->ip_src,output:2 (NAT rewrite from
 internal gw to external gw)
```

# Tools

**odl_tools**: Useful scripts and other tools are located in /opt/tools. Download from: https://github.com/shague/odl_tools

- os_xxx: openstack neutron commands for creating networks, vms and floating ips
- os_ssh: os_ssh.sh <vm ip>: logs into tenant vms via the dhcp namespace
- ossbg.sh, osdbg2.sh: collects debugging information about the ovsdb node: addresses, interfaces, namespaces, flows
- osreset.sh: uses unstack.sh and more to fully clean the ovsdb/openvswitch between tests and clean the logs
- dbgiptables.sh: dumps the iptables
- finderrors.sh: greps through stack logs to find errors

**showOvsdbMdsal.py**: Useful for parsing and dumping the mdsal datastore

# OVSDB MDSAL Parser - showOvsdbMdsal.py

```
/opt/tools/showOvsdbMdsal.py --port 8087 [-c] [--ip <servicehost>]

aliasMap:
  alpha      ->  openflow:7690419299910    br-int  00:00:06:fe:90:b5:e6:46
  bravo      ->  openflow:135157385393989  br-int  00:00:7a:ec:c7:f1:f7:45
  charlie    ->  openflow:183039298907979  br-ex   00:00:a6:79:28:64:2f:4b
  delta      ->  openflow:200144153366857  br-ex   00:00:b6:07:b1:2a:51:49

ovsdbNode:192.168.254.31:51687 mgr:192.168.254.31:6640 version:2.3.1
  alpha:br-int
    of:1 tapd0d15959-1f mac:fa:16:3e:9f:82:6c ifaceId:d0d15959-1f1d-44d4-b531-
93c96d892418
    of:2 patch-ext
    of:3 vxlan-192.168.254.32
    of:4 tap883f9022-bd mac:fa:16:3e:13:44:69 ifaceId:883f9022-bdf5-4dff-b4e0-
fcc8ae8096ed
  delta:br-ex
    of:1 eth2
    of:2 patch-int
```

# OVSDB MDSAL Parser - showOvsdbMdsal.py - Continued

```
operational tree flows at alpha
  table 0: DEFAULT_PIPELINE_FLOW_0
  table 0: DropFilter_1
  table 0: DropFilter_4
  table 0: LLDP
  table 0: LocalMac_1500_1_fa:16:3e:9f:82:6c
  table 0: LocalMac_1500_4_fa:16:3e:13:44:69
  table 0: TunnelIn_1500_3
  table 20: ArpResponder_1500_10.100.5.1
  table 20: ArpResponder_1500_10.100.5.2
  table 20: ArpResponder_1500_10.100.5.3
  table 20: ArpResponder_1500_10.100.5.4
  table 20: ArpResponder_OFPort|2_192.168.56.10
  table 20: DEFAULT_PIPELINE_FLOW_20
  table 30: DEFAULT_PIPELINE_FLOW_30
  table 30: InboundNAT_2_1500_192.168.56.10
  table 40: DEFAULT_PIPELINE_FLOW_40
  table 40: Egress_Allow_VM_IP_MAC_4fa:16:3e:13:44:
 69_Permit_
  table 40: Egress_DHCP_Client_Permit_
  table 40: Egress_DHCP_Server_4_DROP_
  table 50: DEFAULT_PIPELINE_FLOW_50
```

```
  table 60: DEFAULT_PIPELINE_FLOW_60
  table 60: Routing_external_1500_10.100.5.1/24
  table 70: DEFAULT_PIPELINE_FLOW_70
  table 70: L3Forwarding_1500_10.100.5.2
  table 70: L3Forwarding_1500_10.100.5.3
  table 70: L3Forwarding_1500_10.100.5.4
  table 80: DEFAULT_PIPELINE_FLOW_80
  table 90: DEFAULT_PIPELINE_FLOW_90
  table 90: Ingress_DHCP_Server1500_FA:16:3E:9F:82:
6C_Permit_
  table 100: DEFAULT_PIPELINE_FLOW_100
  table 100: OutboundNATExclusion_1500_10.100.5.0
/24
  table 100: OutboundNAT_1500_10.100.5.3
  table 110: BcastOut_1500
  table 110: DEFAULT_PIPELINE_FLOW_110
  table 110: LocalTableMiss_1500
  table 110: TunnelFloodOut_1500
  table 110: TunnelOut_1500_3_fa:16:3e:ce:d7:ad
  table 110: UcastOut_1500_1_fa:16:3e:9f:82:6c
  table 110: UcastOut_1500_4_fa:16:3e:13:44:69
```

# Want to bake your own pizza?

- Clone OpenDaylight ovsdb code: git clone https://git.opendaylight.org/gerrit/ovsdb
- Build it: mvn clean install
- Setup one or two node openstack setup. Do following config on network node to connect it to OpenDaylight controller:
  - Stop neutron-plugin-openvswitch-agent (if running)
  - Configure ml2_conf.ini for OpenDaylight
    - type_drivers = local,gre,vxlan
    - tenant_network_types = vxlan
    - mechanism_drivers = opendaylight
  - Configure ml2_conf_odl.ini
    - url = http://<controller-ip>:8080/controller/nb/v2/neutron
    - username = admin
    - password = admin
  - Restart neutron server
- Set “local_ip” attribute for ovsdb on both control and compute node
  - OVSUUID=$(ovs-vsctl get Open_vSwitch . _uuid); ovs-vsctl set Open_vSwitch
  - $OVSUUID other_config:local_ip=<local-ip>
- Set manager for ovsdb instance on all the nodes
  - ovs-vsctl set-manager tcp:<controller-ip>:6640
- Setup is ready to create the network.
- For Devstack based setup:
  - https://wiki.opendaylight.org/view/OVSDB:Helium_and_Openstack_on_Fedora20
  - https://wiki.opendaylight.org/view/OVSDB:Lithium_and_Openstack_on_CentOS7 (work in progress)

# ....will talk about:

- What the OVSDB Project offers?
- Why it's the Center of Attraction?
- Brief Overview of Open vSwitch & Management Protocol
- High Level Architecture and Control Flow
- What we have accomplished in Lithium
- What are we planning for Beryllium?
- Let’s ./stack!
- Looking to contribute?

# Start From Here

- Checkout all the info on the project wiki:
  - https://wiki.opendaylight.org/view/OVSDB_Integration:Main
  - Weekly meetings on Tuesday's at 12:00p PST
  - Getting started: How to pull and build the code
  - Tutorials
- Connect with active developers in the community on the #opendaylight-ovsdb IRC channel at freenode.net
- Poke {vishnoianil,shague,flaviof} on irc #opendaylight-ovsdb
- OVSDB Trello page for project task tracking: https://trello.com/odlovsdb
- Join the conversation through lists.opendaylight.org and ask.opendaylight.org