OpenStack in the Enterprise

BRKRST-2644

Shannon McFarland
Principal Engineer
CCIE #5245

Cisco live!

Agenda

- What is OpenStack?
- OpenStack Participation
- OpenStack Deployment in the Enterprise
- Deployment Walk-thru
- Running Applications
- Monitoring
- Cisco Product Integration
- Havana/Demo
- Conclusion



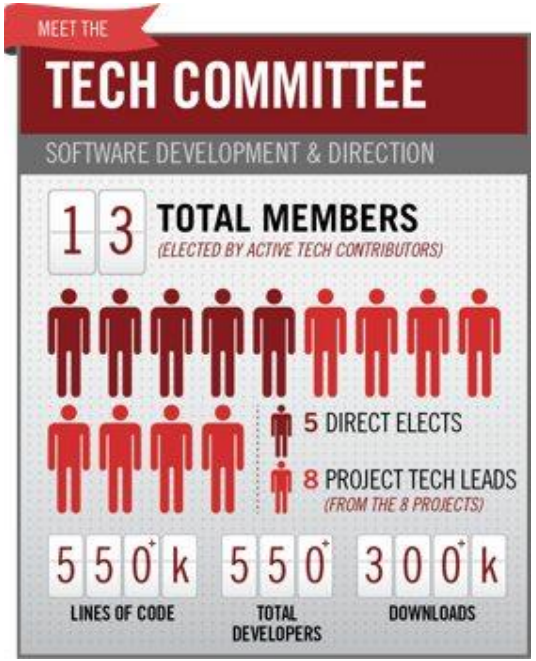

What is OpenStack?

OpenStack

- "OpenStack is a collection of open source technologies delivering a massively scalable cloud operating system" - openstack.org
- Basically, it is a full open source cloud stack that can be used as a starting point for a private or public cloud
- Releases are on a 6-month interval: Havana (November 2013 is the latest release and Icehouse is next
- Unreal community growth since its inception
- Timeline:
  - NASA Launches Nebula - One of the first cloud computing platforms built for Federal Government Private Cloud
  - March 2010: Rackspace Open Sources Cloud Files software, aka Swift
  - May 2010: NASA open sources compute software, aka "Nova"
  - June 2010: OpenStack is formed
  - July 2010: The inaugural Design Summit
  - April 2012: Openstack foundation formed
  - November 2013: Havana released
  - April 2014: Icehouse scheduled to release
- OpenStack is not a 1:1 replacement for your fill-in-the-blank DC Server Virtualisation Platform

OpenStack Foundation

- https://www.openstack.org/foundation/
  - Elected technical committee, elected board, individual and organisation membership



OpenStack is “Project” Based

Compute

“Nova”

- Houses VMs
- API driven
- Support for multi-hypervisors

Storage

Image, Object, Block

“Glance, Swift, Cinder”

- Instance/VM image storage
- Cloud object storage
- Persistent block level storage

Dashboard

“Horizon”

- Web app for controlling OpenStack resources
- Self-service portal

Identity

“Keystone”

- Centralised policies
- Tenant mgmt.
- RBAC
- Ext. integration (LDAP)

Networking

“Neutron”

- Networking as a service
- Multiple models
- IP address mgmt.
- Plugins to external HW

Metering

“Telemetry”

- Central collection point
- Metering and monitoring

Orchestration

“Heat”

- Template-based orchestration engine
- More rapid deployment of applications

A Note on Name Changes

- The OpenStack Foundation made the decision to remove “Quantum” from their references due to some naming/trademark conflicts. “Quantum”, as of the Havana release is known as “Neutron”

- There may be references in text and my statements that still include Quantum but when I say “Quantum” or “Neutron” – they are the same thing

- Also, Ceilometer is now Telemetry

Getting Started

- Try/Dev/Demo:
  - http://devstack.org/
  - http://www.stackops.com/
  - http://trystack.org/
- Many, many, many blogs on setting up OpenStack on every virtual platform imaginable
- Grizzly:
  - http://docwiki.cisco.com/wiki/OpenStack:Grizzly:All-In-One
  - http://docwiki.cisco.com/wiki/OpenStack:Grizzly-Multinode
- Havana:
  - http://docwiki.cisco.com/wiki/Openstack:Havana-Openstack-Installer
  - http://docwiki.cisco.com/wiki/OpenStack_Havana_Release:_High-Availability_Manual_Deployment_Guide

OpenStack Participation

Who is Involved in OpenStack?

- You name it – Compute, Storage, Networking vendors, Universities, Gov't, massive pile of OpenStack-specific startups
- Traditional HW vendors – Cisco, HP, Dell, Arista, etc…
- Providers – Rackspace, AT&T, Comcast, etc…
- Startups – PistonCloud, SwiftStack and many, many more…
- Distributions & Support – Red Hat, Canonical, SUSE
- Some are focused on only small parts of OpenStack such as driving object storage features (SwiftStack), or automated deployment and support (PistonCloud) or networking and compute pull-thru as well as project leadership (Cisco – Nexus, UCS, services, Quantum/Neutron)

Cisco + OpenStack

- Cisco is deeply involved on many fronts and we will get even more involved over time

- Lew Tucker, VP/CTO, Cloud Computing “owns” OpenStack at Cisco but many other teams involved: CE, SDU, SAVTG, AS, WebEx, etc …

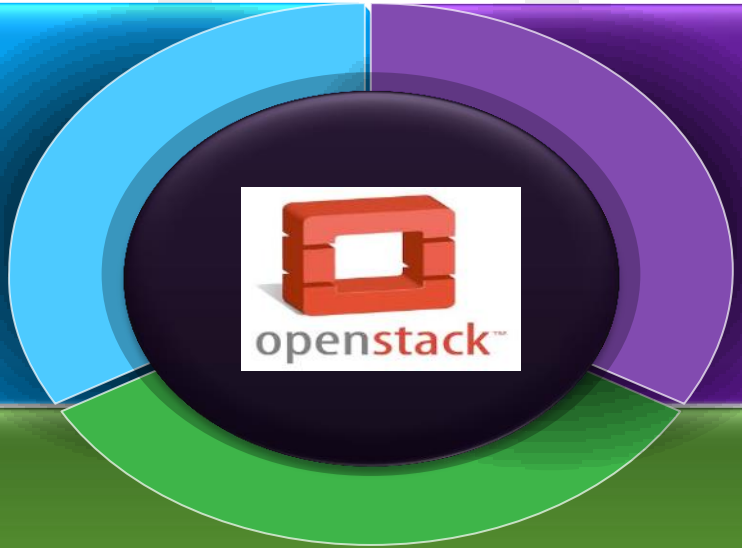
- External portals are being developed and matured:
  - External Cisco.com: www.cisco.com/go/openstack
  - External Docwiki: http://docwiki.cisco.com/wiki/OpenStack
  - GitHub Cisco Repository: https://github.com/CiscoSystems/puppet_openstack_builder

- Multiple simultaneous efforts underway

Cisco's Focus on OpenStack - Today

Community

- Neutron – Network Service
- Horizon – Dashboard
- Keystone – Identity
- Swift – Object Storage
- Automation – PuppetLabs
- HA Design
- OpenStack Board/PTL



Engineering

- Cisco Product Integration
- Nexus Plugins – Quantum
- UCS
- CIAC
- Co-developed solutions (Red Hat, Canonical, SUSE)

Customers

- Cisco Designs on specific releases in 'beachhead' accounts
- Start simple, build from there – Focus on automation and HA
- Evangelisation of what Cisco is doing - Thought Leadership – Help customers know What, When, Where & How

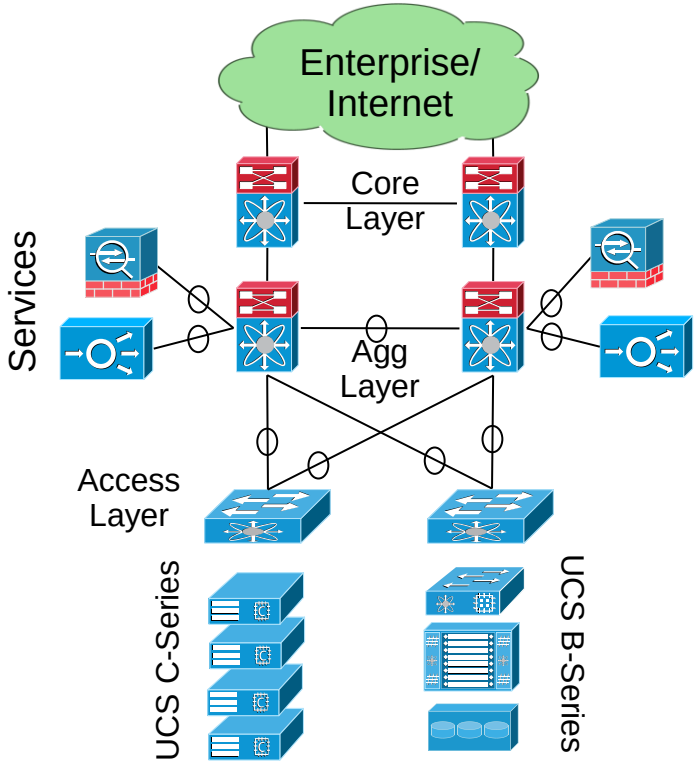
Cisco + Other Distributions/Vendors

- Red Hat: http://www.cisco.com/en/US/prod/collateral/ps10265/wp_openstack.pdf
- CVD: http://www.cisco.com/en/US/docs/solutions/Enterprise/Data_Center/OpenStack/Grizzly/OpenStack_Red_Hat_RDO.pdf
- FlexPod: http://nt-ap.com/IfgPlx
- Solution Accelerator Paks: http://www.cisco.com/web/solutions/openstack/le_sb_open.pdf
- Collateral with Canonical and SUSE on the way

OpenStack Deployment in the Enterprise

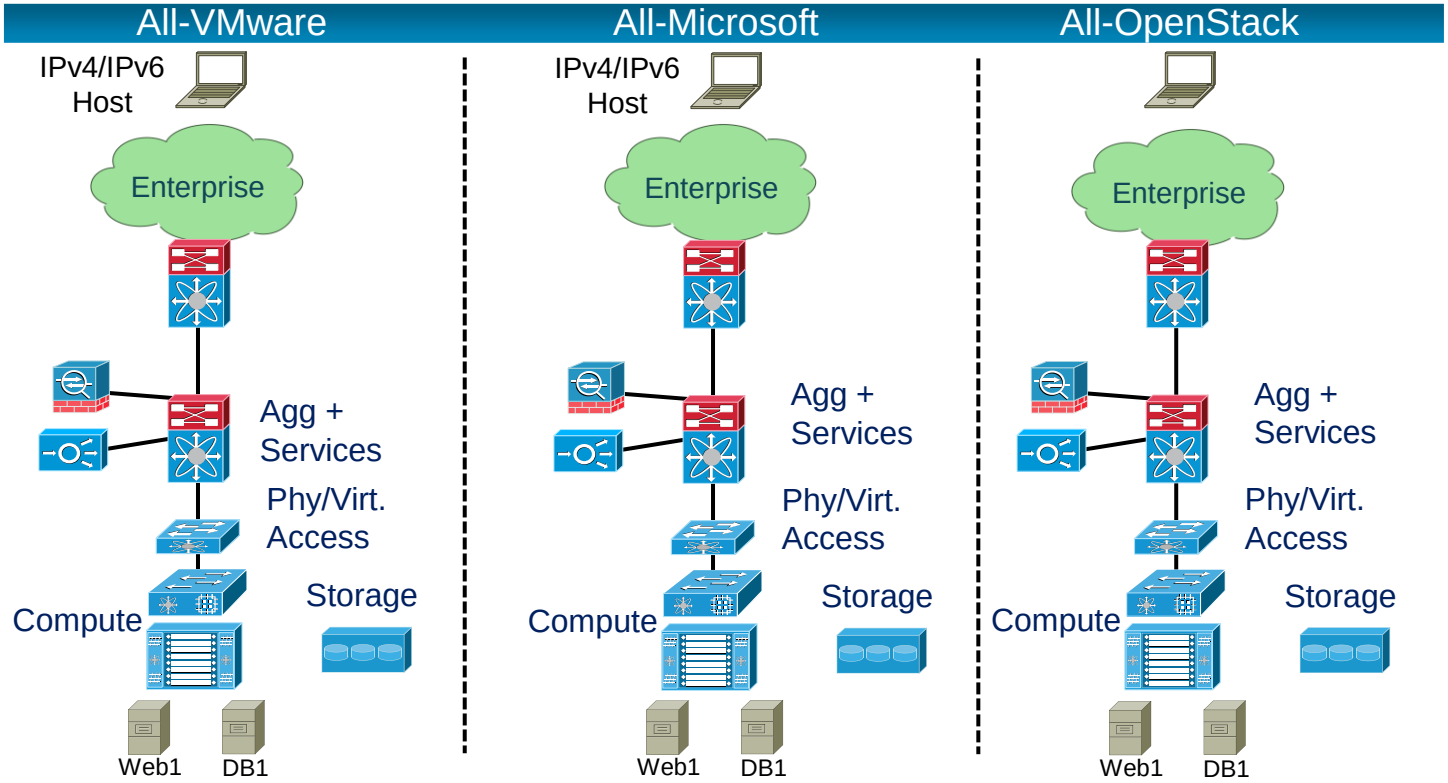
What Really Changes in my Data Centre/Internet Edge?

- OpenStack components live South of the Top-of-Rack switch
- Your existing DC, Internet Edge and BN architecture stays the same
- It's about the compute, storage and orchestration/management tiers
- Even your apps go largely unchanged



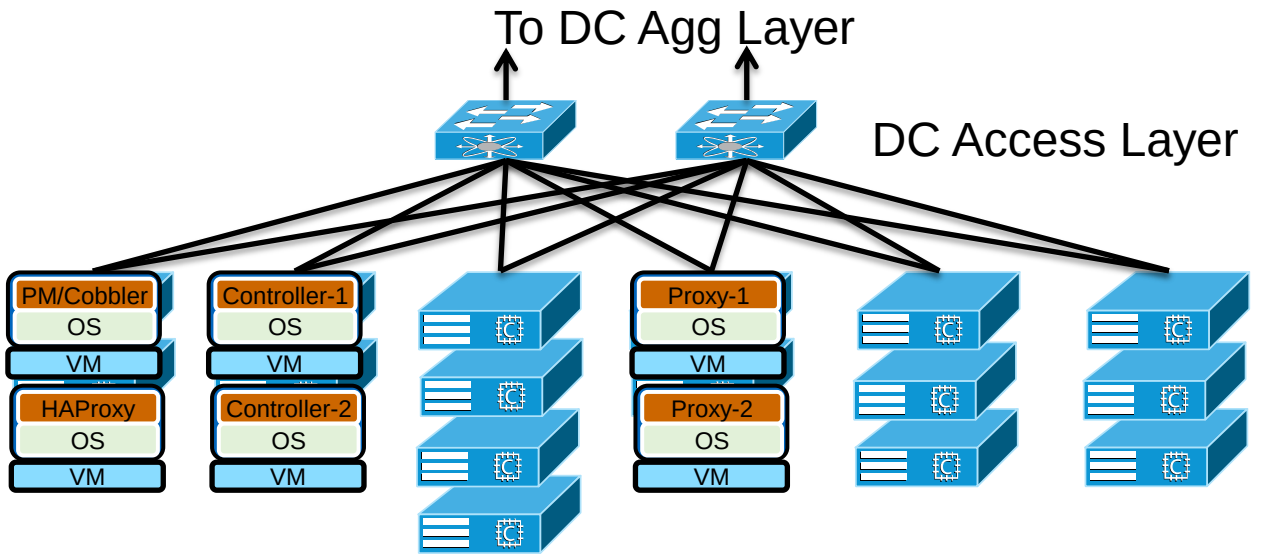
What is Different Between These Three?

Answer: Nothing, topologically



OpenStack Nodes/Roles

- Example on UCS C-series
- Active/Active controllers
- HAProxy/Keepalived or HW SLB for Swift Proxy Nodes
- Object and block storage
  - Images, app data
  - Persistent storage
- Support nodes (Ctrl/Proxy also) often run as VMs or can be baremetal



*Support Nodes
-Puppet Master
-Cobbler
-DNS
-HAProxy/
Keepalived
*Can run as VMs

A/A Controller Nodes

Compute Nodes

Swift Proxy Nodes

Swift Object Storage Nodes

Cinder Block Storage Nodes

To Automate or Not and How Much to Automate

- Manually deploy it all? Automate only the OpenStack setup? Automate OpenStack + Apps?
- **Single Shot** – Manually setup everything (the best way to learn OpenStack):
  - http://docwiki.cisco.com/wiki/COE_Grizzly_Release:_High-Availability_Manual_Installation_Guide
- **Semi-Automatic** – Use automation for 'some' of the setup and maintain/modify manually:
  - http://docwiki.cisco.com/wiki/OpenStack:Grizzly:All-In-One
  - http://docwiki.cisco.com/wiki/OpenStack:Grizzly-Multinode
  - http://docwiki.cisco.com/wiki/OpenStack_Grizzly_Release:_High-Availability_Automated_Deployment_Guide
  - http://puppetlabs.com/
  - http://www.opscode.com/chef/
  - https://juju.ubuntu.com/
- **Automatic** – Automate everything with Puppet, Chef, JuJu or turnkey automation stuff: http://www.pistoncloud.com/

High-Level Planning Summary

- Deploy OpenStack in existing ‘pod’ or a new one?
- Hardware inventory – All rack servers, all blade servers, HW + VMs
- What app(s) do you plan to run in the new deployment?
- To multi-tenant or not? This is a functional and business topic as much as a technical one – Always deploy with multi-tenancy in mind
- IP address planning – NAT inside OpenStack? No NAT? Overlapping IPs?
- Automation choices
- Use a ‘pure’ OpenStack (only OpenStack projects) deployment or a hybrid deployment where you use some of what OpenStack offers and leverage 3rd party applications/management/monitoring services
- Knowing the limitations of current high-availability/disaster-recovery (HA/DR) models with OpenStack
- Other stuff we will talk about along the way ….

Network Decisions

- Use Neutron (AKA: Quantum) with various networking models
  - Flat, FlatDHCP, VLAN modes, GRE or VLAN, VXLAN?

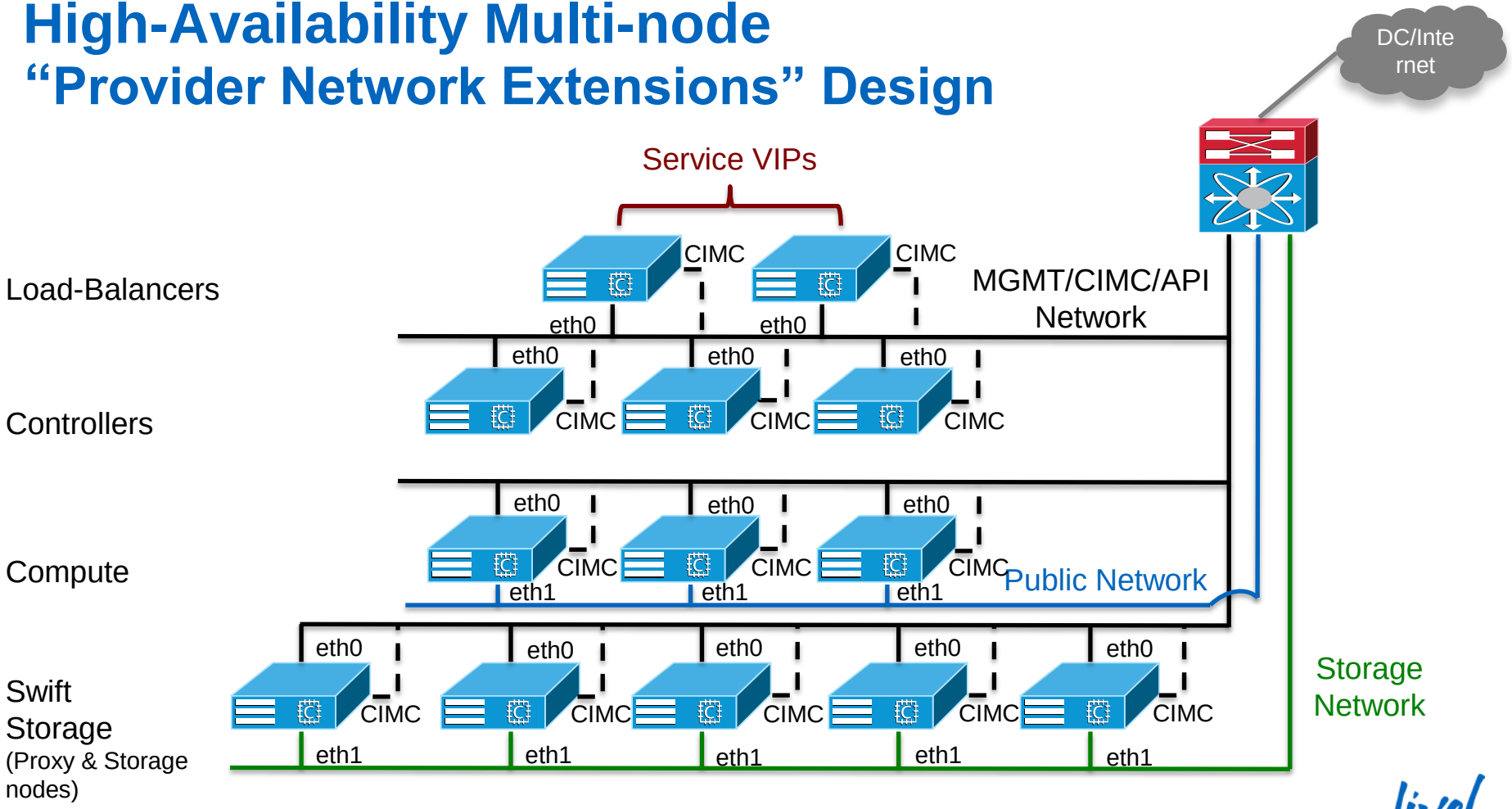
- OpenStack Networking
  - http://docs.openstack.org/trunk/openstack-network/admin/content/use_cases.html
  - OpenStack Network Controller role running on the OpenStack Controller node
  - Dedicated Network Controller Node: http://docwiki.cisco.com/wiki/Folsom_Manual_Install
  - HA Network Controllers with Provider Networks
  - Multiple networks/subnets and multiple routers – Use with multiple tenants

- Networking Scale
  - Overlapping IPv4 Addresses
  - To NAT or not to NAT
  - How many physical networks do you use? Management, public, private, etc…
  - GRE vs. VLAN

High Availability Decisions

- Know what you don't know
- Pick your release – HA matures on every release: Folsom (sucked for HA) -> Grizzly (getting better) -> Havana (MUCH better)– You may have to use other open source tools to get a complete system highly available
- Cisco HA design – http://docwiki.cisco.com/wiki/OpenStack_Havana_Release:_High-Availability_Manual_Deployment_Guide
- Many components are:
  - Databases: Options include MySQL-WSREP and Galera
  - Message Queue: RabbitMQ Clustering and RabbitMQ Mirrored Queues
  - API/Web services: HAProxy, Keepalived, traditional SLB
  - Swift proxy nodes: HAProxy, Keepalived, traditional SLB
  - Swift nodes: Architecturally designed to be available (i.e. multiple copies of objects)
  - Compute node: Nothing directly HA, but can use Migration for planned maintenance windows
- Puppet HA: Search "puppet master redundancy" or "masterless puppet" – you will land plenty of reading choices ;-)

High-Availability Multi-node "Provider Network Extensions" Design



OpenStack Deployment Walk Through

Getting Started – All-in-One (AIO) Deployment

- If you are looking for a more simple starting point for learning, testing and deploying OpenStack but are needing something more realistic than DevStack
  - http://docwiki.cisco.com/wiki/OpenStack:Grizzly:All-In-One
  - http://docwiki.cisco.com/wiki/Openstack:Havana-Openstack-Installer

- Multiple networking and storage models available

Cisco Design on Grizzly – Automated Deployment

- Grizzly release using Puppet and a variety of other tools to automate the OpenStack deployment tasks
- http://docwiki.cisco.com/wiki/OpenStack:Grizzly-Multinode
- Example will include:
  - Build Server running: Puppet master, Cobbler, etc…
  - Controller node
  - 2 x Compute nodes
  - Three tenants and associated users/admins: “OpenStack”, “Sales-Rockies”, “Dev-Cloud”
  - Multiple physical and virtual networks (more on this later)
  - Running on Cisco UCS C-Series
  - Ubuntu 12.04 LTS

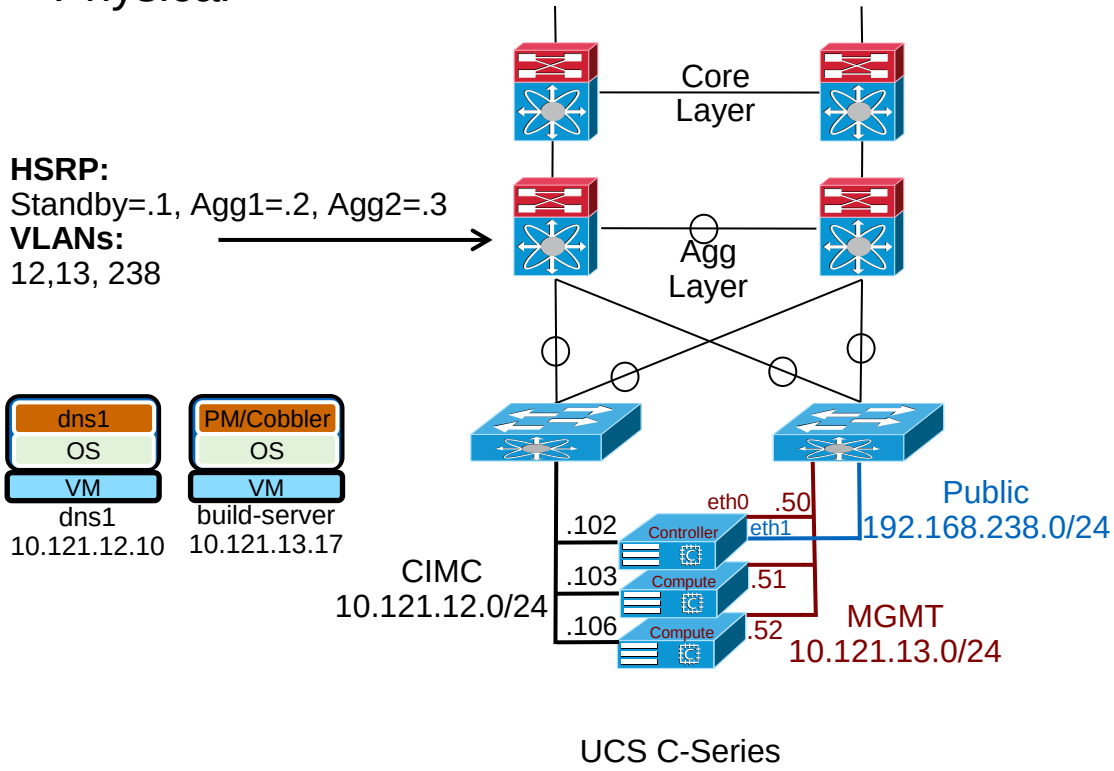
User Story: example.com

- Goal: Build a simple OpenStack cloud that will host three user groups that are transitioning from another Data Centre
- Mix of existing VMs running in same DC as new OpenStack deployment and new physical Cisco UCS C-Series servers
- Everything is ‘contained’ in a single DC Access layer
- Using Quantum with Open vSwitch plugin
- OpenStack security groups are augmented by more robust security in existing DC services layer
- Puppet is used along with other tools such as Cobbler to provision/manage OpenStack deployment
- A separate (existing) Puppet deployment will be used for instance/VM provisioning/application management
- All tenants/networks that need ‘outside’ public access will use a shared public network range using floating IPs (NAT)

example.com Multinode Topo

Physical

HSRP:
Standby=.1, Agg1=.2, Agg2=.3
VLANs:
12,13, 238



Host	Interface	Address	Role
Agg1/2	VLAN12, 13, 238	HSRP=.1 Intf=.2,.3	L3 Agg layer switches
dns1	eth0	10.121.12.10	DNS server
build-server	eth0	10.121.13.17	puppet, cobbler, nagios, etc..
control-server	eth0 eth1	10.121.13.50 Bridge: 192.168.238.x	OS controller, Quantum network controller
compute-server01	eth0	10.121.13.51	OS compute
compute-server02	eth0	10.121.13.52	OS compute

Building the Nodes

Automated Deployment Steps

Reference

1. **Document your addressing (IP/MAC)/node roles**
2. **Cable your servers and configure any networking gear needed by OpenStack**
3. **Deploy the build server**
4. **Customise the build server (Modify example site.pp/apply manifest)**
5. **Kick-off control server and compute server(s) builds**
6. Manual or automated (test scripts) Quantum/Neutron setup
7. Download images and upload into Glance (if not using test script in step 6)
8. Boot instance, test connectivity
9. Modify setup to meet your needs
10. Have a nice day ☺

Build Server - Step 1: Login, setup build-server

- Minimum requirements for build server: 2 GB RAM, 20 GB storage, Internet connectivity and on the same network as management interfaces of OpenStack nodes
- We are going to operate as root:

```
localadmin@build-server:~$ sudo –i
```

- Perform updates/upgrades, install puppet, git, ipmitool and debmirror and perform git clones of required puppet manifests and modules
  - You can do this in one of two ways (Note – See URL below for proxy instructions):
    - Method 1: Automated – Run a single script

```
root@build-server:~# curl -s -k -B
https://raw.github.com/CiscoSystems/grizzly-manifests/g.3/install_os_puppet |
/bin/bash
```

    - Method 2: Manual:http://docwiki.cisco.com/wiki/OpenStack:Grizzly-Multinode#Model_2:_Run_the_Commands_Manually

Build Server - Step 2: site.pp walk-thru

- Copy example site.pp file and edit it to match your environment

```
root@build-server:~# cp /etc/puppet/manifests/site.pp.example /etc/puppet/manifests/site.pp
root@build-server:~# vi /etc/puppet/manifests/site.pp
```

- Let's review the key parts of the site.pp that need to be edited

```
# If using a proxy, set it here
$proxy                  = http://10.129.16.14:8080
# Select either the FTP distribution location or HTTP. Note HTTP works better behind proxies
#$location              = 'ftp://ftpeng.cisco.com/openstack'
# Alternate, uncomment this one, and comment out the one above
$location               = 'http://openstack-repo.cisco.com/openstack'
# Supplemental repo for non-core OpenStack packages
# $supplemental_repo    = 'ftp://ftpeng.cisco.com/openstack/cisco_supplemental'
$supplemental_repo      = 'http://openstack-repo.cisco.com/openstack/cisco_supplemental'
# Hostname of build server. If changed from default, make sure it is changed throughout file
$build_node_name        = 'build-server'
```

Build Server - site.pp continued

```
# Set your local NTP server
$ntp_servers = [’ntp.ubuntu.com']

# Build Server Cobbler Variables. IP/Network of build-server and domain name
$cobbler_node_ip        = '10.121.13.17'
$node_subnet            = '10.121.13.0'
$node_netmask           = '255.255.255.0’
$node_gateway           = '10.121.13.1’
$domain_name            = 'example.com’

# Local user/password (“ubuntu”) created on each OpenStack node. Change to your liking
$admin_user             = 'localadmin'
$password_crypted       = '$6$UfgWxrIv$
k4KfzAEMqMg.fppmSOTd0usI4j6gfjs0962.JXsoJRWa5wMz8yQk4SfInn4.WZ3L/MCt5u.62tHDGB36EhiKF1’
$autostart_puppet       = true
# If using UCS B-Series blades, enter the port on which the UCSM accepts requests
$ucsm_port              = ‘443’
```

Build Server - site.pp continued

```
########### OpenStack Variables ############
# These values define parameters which will be used to deploy and configure OpenStack
# once Ubuntu is installed on your nodes
#
# Change these next 3 parameters to the network settings of the node which will be your
# OpenStack control node
$controller_node_address       = '10.121.13.50'
$controller_node_network       = '10.121.13.0'
$controller_hostname           = 'control-server'
# Specify the network which should have access to the MySQL database on the OpenStack control
# node. Typically, this will be the same network as defined in the controller_node_network
# parameter above. Use MySQL network wild card syntax to specify the desired network.
$db_allowed_network            = '10.121.13.%'
# Define network connectivity of the OpenStack controller
$controller_node_public        = $controller_node_address
$controller_node_internal      = $controller_node_address
```

Build Server - site.pp continued

```
# Specify which interface in each node is the API Interface
# This is also known as the Management Interface
$public_interface               = 'eth0'
# Specify the interface used for external connectivity such as floating IPs (only in
network/controller node)
$external_interface             = 'eth1'


# Select the drive on which Ubuntu and OpenStack will be installed in each node. Current
# assumption is that all nodes will be installed on the same device name
$install_drive           = '/dev/sda'
```

Build Server - site.pp continued

Reference

```
############ OpenStack Service Credentials ############

# This block of parameters is used to change the user names and passwords used by the services which make up
# OpenStack. The following defaults should be changed for any production deployment

$admin_email              = 'root@localhost'
$admin_password           = 'Cisco123'
$keystone_db_password     = 'keystone_db_pass'
$keystone_admin_token     = 'keystone_admin_token'
$mysql_root_password      = 'mysql_db_pass'
$nova_user                = 'nova'
$nova_db_password         = 'nova_pass'
$nova_user_password       = 'nova_pass'
$libvirt_type             = 'kvm'
$glance_db_password       = 'glance_pass'
$glance_user_password     = 'glance_pass'
$glance_sql_connection    = "mysql://glance:${glance_db_password}@${controller_node_address}/glance"
$cinder_user              = 'cinder'
$cinder_user_password     = 'cinder_pass'
$cinder_db_password       = 'cinder_pass'
$quantum_user_password    = 'quantum_pass'
$quantum_db_password      = 'quantum_pass'
$rabbit_password          = 'openstack_rabbit_password'
$rabbit_user              = 'openstack_rabbit_user'
$swift_password           = 'swift_pass'
$swift_hash               = 'swift_secret'
```

Build Server - site.pp continued

```
localadmin@control-server:~$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:10:18:cf:b0:18
          inet addr:10.121.13.50  Bcast:10.121.13.255  Mask:255.255.255.0
```

```
cobbler_node { 'control-server':
   mac             => '00:10:18:CF:B0:18',
   ip              => '10.121.13.50',
   power_address   => '10.121.12.102',
   power_user      => 'admin',
   power_password => 'password',
   power_type      => 'ipmitool',
 }
```

```
cobbler_node { 'compute-server01':
   mac             => '00:10:18:CF:AE:48',
   ip              => '10.121.13.51',
   power_address   => '10.121.12.103',
   power_user      => 'admin',
   power_password => 'password',
   power_type      => 'ipmitool',
 }
```

Build Server - site.pp continued

```
### Node types ###
# These lines specify the host names in your OpenStack cluster and what the function of each
# host is
# Change build_server to the host name of your build node
node build-server inherits build-node { }


# Change control_server to the host name of your control node
node 'control-server' inherits os_base {
  class { 'control':
  }
# Change compute_serverXX to the host name of your compute nodes
node 'compute-server01' inherits os_base {
  class { 'compute':
    internal_ip         => '10.121.13.51',
  }
node 'compute-server02' inherits os_base {
  class { 'compute':
    internal_ip         => '10.121.13.52',
  }
```

Build Server - Step 3: Puppet Apply

- Run “puppet apply” against the site.pp file we just built -

```
root@build-server:~# puppet apply -v /etc/puppet/manifests/site.pp
```

- Puppet apply will install the following on the build server as well as prepare for deployment of the OpenStack nodes we defined in the site.pp file:
  - ntpd – Time synchronisation
  - tftpd-hpa – TFTP server for PXE boot of OpenStack nodes
  - dnsmasq – DNS and DHCP server
  - cobbler – Installation and boot management
  - apt-cacher-ng – Caching proxy for package installation
  - nagios – Infrastructure monitoring application
  - collectd – Statistics collection
  - graphite/carbon – Real-time graphing system
  - apache – Web server for hosting graphite, nagios and puppet services

Build Server - Step 4: Puppet Plugins/Cobbler list

- Stage the puppet plugins:

```
root@build-server:~# puppet plugin download
```

- Ensure that the nodes we defined are in the cobbler system:

```
root@build-server:~# cobbler system list
   compute-server01
   compute-server02
   control-server
```

Build Server - Step 5: Build the OS Nodes

- Run the “clean_node.sh” script for each node:

```
root@build-server:~# /etc/puppet/manifests/clean_node.sh control-server example.com
```

- Run a for loop to kick of script for all nodes in system list:

```
root@build-server:~# for n in `cobbler system list`; do clean_node.sh $n ; done
```

- This will take awhile to install Ubuntu on each node and for the installed Puppet agent to run and install the OpenStack components

Quantum Setup for Initial Testing – Manual

```
# Source the "openrc" file to export authentication/tenant info

root@control-server:~# source openrc

# Create a Quantum public network

root@control-server:~# quantum net-create public --router:external=True

# Create a Quantum subnet for the public network. Set starting address to be higher than
upstream DC agg-layer HSRP addresses (.1, .2, .3)

root@control-server:~# quantum subnet-create --allocation-pool
start=192.168.238.5,end=192.168.238.254 public 192.168.238.0/24

# Create internal (data) network used by "openstack" tenant created by puppet process

root@control-server:~# quantum net-create private

# Create a subnet for the private network. Alter DNS servers if needed.

root@control-server:~# quantum subnet-create --name private-10.10.10.x private
10.10.10.0/24 --dns_nameservers list=true 10.121.12.10

# Create a Quantum router

root@control-server:~# quantum router-create os-router-1

# Add Quantum router interface to previously create private subnet

root@control-server:~# quantum router-interface-add os-router-1 private-10.10.10.x

# Set the Quantum router's gateway to the public network (Like a default gw)

root@control-server:~# quantum router-gateway-set os-router-1 public
```

Quantum Setup for Initial Testing – Manual Cont'd

```
# Get the router ID. Note output below is just a snippet
root@control-server:~# quantum router-list
| id                                   | name        |
+--------------------------------------+-------------+
| db31b4fa-96e7-4bc6-98e0-4d945a46d136 | os-router-1 |
# Get the Quantum router address on the public network
root@control-server:~# quantum port-list -- --device_id db31b4fa-96e7-4bc6-98e0-4d945a46d136 --device_owner network:router_gateway
| id                                   | name | mac_address       | fixed_ips
+--------------------------------------+------+-------------------+----------+
| 8a8db076-b3ff-4fac-88a9-0abbfcf6079e |      | fa:16:3e:14:3b:ad | {"subnet_id": "92978329-0494-4bb5-9e7d-98f47f106ad0", "ip_address": "192.168.238.6"} |
```

```
# Set a static route on the upstream DC Agg layer (or first L3 hop device) for the new subnet with the next hop of the Quantum router address


n7k-agg-1(config)# ip route 10.10.10.0 255.255.255.0 192.168.238.6
```

Image Download – Upload to Glance

```
# Download Ubuntu Precise image
root@control-server:~# wget http://cloud-images.ubuntu.com/precise/current/precise-server-cloudimg-amd64-disk1.img


# Upload image into Glance
root@control-server:~# glance add name="precise" is_public=true container_format=ovf
disk_format=qcow2 < precise-server-cloudimg-amd64-disk1.img
```

```
# Alternatively, you can download a Cirros image
root@control-server:~# wget wget http://download.cirros-cloud.net/0.3.1/cirros-0.3.1-x86_64-disk.img
# Upload image into Glance
root@control-server:~# glance add name="cirros-x86_64" is_public=true disk_format=qcow2
container_format=ovf < cirros-0.3.1-x86_64-disk.img
```

SSH Keys and Boot Image

```
# Generate a new SSH key pair
root@control-server:~# ssh-keygen
# Add a new keypair into Nova
root@control-server:~/.ssh# nova keypair-add --pub_key ~/.ssh/id_rsa.pub ctrl-key
```

```
# Get a list of the networks
root@control-server:~# quantum net-list
+--------------------------------------+----------+--------------------------------------+
| id                                   | name     | subnets                              |
+--------------------------------------+----------+--------------------------------------+
| a43a64ac-7707-419c-b719-ce9638083888 | public   | 92978329-0494-4bb5-9e7d-98f47f106ad0 |
| a59a0230-f118-4f28-bcce-7e45ae7c0344 | private  | 1fb02d09-1462-4e4a-adab-c4bece336c13 |
+--------------------------------------+----------+--------------------------------------+
```

```
# Boot an instance using ID for the "private" network
root@control-server:~# nova boot --image precise --flavor m1.tiny --key_name ctrl-key --nic
net-id=a59a0230-f118-4f28-bcce-7e45ae7c0344 Test-VM
```

It's Alive!

```
root@control-server:~# nova list
+--------------------------------------+---------+--------+--------------------+
|                  ID                  |   Name  | Status |      Networks      |
+--------------------------------------+---------+--------+--------------------+
| 4f391561-505f-4d17-adae-df3a904dfb87 | Test-VM | ACTIVE | private=10.10.10.3 |
+--------------------------------------+---------+--------+--------------------+
```

```
root@control-server:~# ip netns exec qrouter-db31b4fa-96e7-4bc6-98e0-4d945a46d136 ping
10.10.10.3
PING 10.10.10.3 (10.10.10.3) 56(84) bytes of data.
64 bytes from 10.10.10.3: icmp_req=1 ttl=64 time=41.6 ms
64 bytes from 10.10.10.3: icmp_req=2 ttl=64 time=0.665 ms
64 bytes from 10.10.10.3: icmp_req=3 ttl=64 time=0.527 ms
root@control-server:~# ip netns exec qrouter-db31b4fa-96e7-4bc6-98e0-4d945a46d136 ssh
ubuntu@10.10.10.3
ubuntu@test-vm:~$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr fa:16:3e:a9:4f:11
          inet addr:10.10.10.3  Bcast:10.10.10.255  Mask:255.255.255.0
```

Access the VM from the 'Outside' – Gather Info

```
root@control-server:~# quantum net-list
+--------------------------------------+---------+--------------------------------------+
| id                                   | name    | subnets                              |
+--------------------------------------+---------+--------------------------------------+
| a43a64ac-7707-419c-b719-ce9638083888 | public  | 92978329-0494-4bb5-9e7d-98f47f106ad0 |
| a59a0230-f118-4f28-bcce-7e45ae7c0344 | private | 1fb02d09-1462-4e4a-adab-c4bece336c13 |
+--------------------------------------+---------+--------------------------------------+
root@control-server:~# quantum port-list
+--------------------------------------+------+-------------------+-------------------------------------------------------------------------------------+
| id                                   | name | mac_address       | fixed_ips                                                                           |
+--------------------------------------+------+-------------------+-------------------------------------------------------------------------------------+
| 024a0619-7113-4075-bd81-9a6009a19e17 |      | fa:16:3e:a7:95:f9 | {"subnet_id": "1fb02d09-1462-4e4a-adab-c4bece336c13", "ip_address": "10.10.10.1"}   |
| 2340872e-68f9-407e-a0ef-bcfa97e53e70 |      | fa:16:3e:7e:f8:b6 | {"subnet_id": "1fb02d09-1462-4e4a-adab-c4bece336c13", "ip_address": "10.10.10.2"}   |
| 3fe91abf-c88d-4072-b75d-eed627b33199 |      | fa:16:3e:18:99:5e | {"subnet_id": "92978329-0494-4bb5-9e7d-98f47f106ad0", "ip_address": "192.168.238.5"} ||
| 82f70e96-a5bc-48fc-97f1-60a9878e4fdf |      | fa:16:3e:a9:4f:11 | {"subnet_id": "1fb02d09-1462-4e4a-adab-c4bece336c13", "ip_address": "10.10.10.3"}   |
| 8a8db076-b3ff-4fac-88a9-0abbfcf6079e |      | fa:16:3e:14:3b:ad | {"subnet_id": "92978329-0494-4bb5-9e7d-98f47f106ad0", "ip_address": "192.168.238.6"} |
```

Access the VM from the ‘Outside’ – FloatingIP

```
root@control-server:~# quantum floatingip-create --port_id 82f70e96-a5bc-48fc-97f1-60a9878e4fdf a43a64ac-7707-419c-b719-ce9638083888
Created a new floatingip:
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    | 10.10.10.3                           |
| floating_ip_address | 192.168.238.7                        |
| floating_network_id | a43a64ac-7707-419c-b719-ce9638083888 |
| id                  | 6bce4afd-6afd-4f38-8fca-fbec8192d47d |
| port_id             | 82f70e96-a5bc-48fc-97f1-60a9878e4fdf |
| router_id           | db31b4fa-96e7-4bc6-98e0-4d945a46d136 |
| tenant_id           | 0a59fafa44084dac9c66cc83ca48fdf4     |
+---------------------+--------------------------------------+
```

Access the VM from the 'Outside' – Security

```
# Add Security Group rules to the default (or create a new group). Allow Ping/SSH
root@control-server:~# quantum security-group-rule-create --protocol icmp --direction ingress default
root@control-server:~# quantum security-group-rule-create --protocol tcp --port-range-min 22 --port-range-max 22 --direction ingress default
# Ping FloatingIP address of VM
root@control-server:~# ping 192.168.238.7
PING 192.168.238.7 (192.168.238.7) 56(84) bytes of data.
64 bytes from 192.168.238.7: icmp_req=1 ttl=62 time=91.7 ms
64 bytes from 192.168.238.7: icmp_req=2 ttl=62 time=0.732 ms

# SSH into the VM
root@control-server:~# ssh ubuntu@192.168.238.7
The authenticity of host '192.168.238.7 (192.168.238.7)' can't be established.
ECDSA key fingerprint is b8:3e:3e:00:5a:d2:94:b9:18:d4:43:fa:ce:d2:2a:82.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.238.7' (ECDSA) to the list of known hosts.
ubuntu@test-vm:~$
```




Defining Tenants, Users, Networks, etc…

example.com Tenant Layout

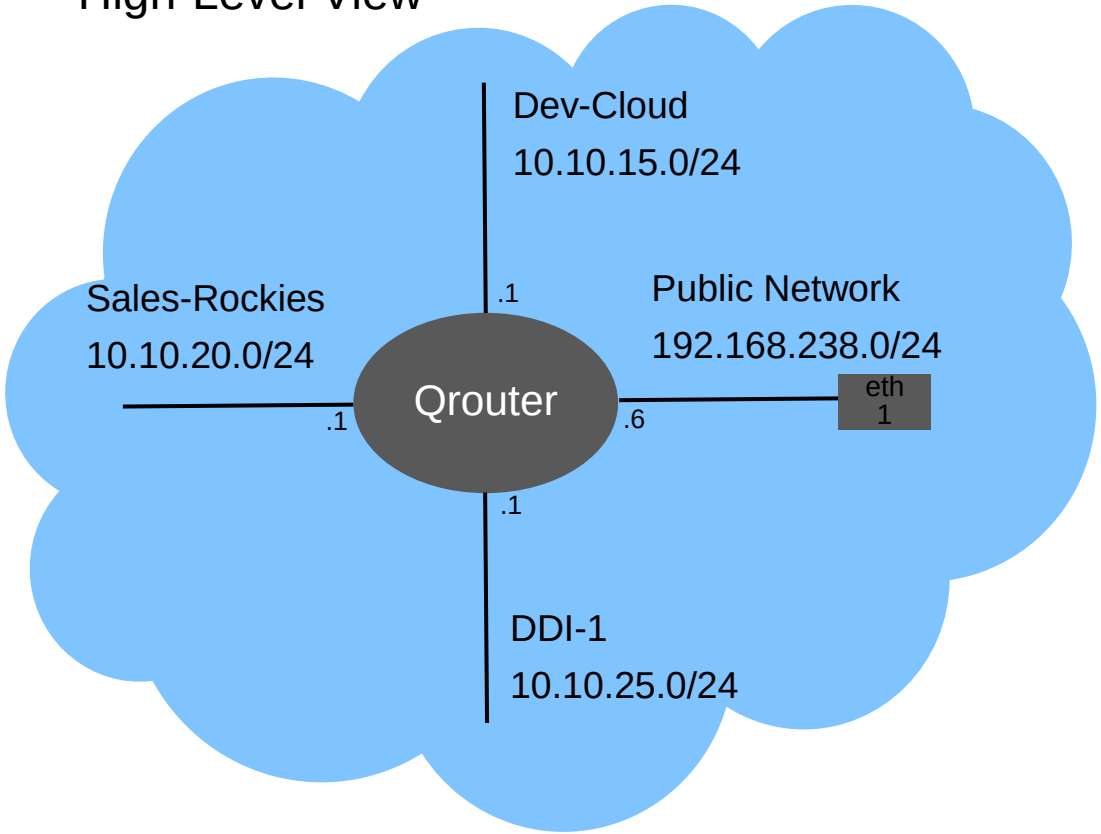
Reference

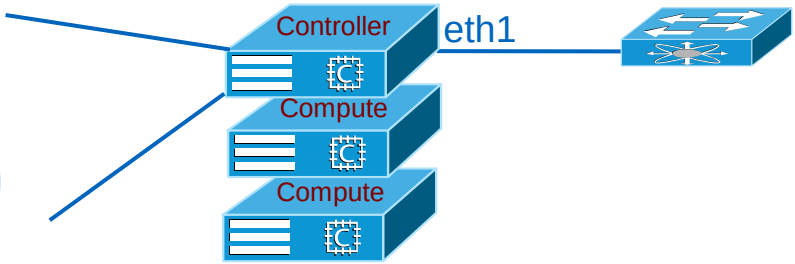
Project/Tenant	Users	Private Network	Public Network (Shared across tenants)	Quantum DHCP/Router Addresses	Instance/VM DHCP range	Security Policy
openstack #test tenant	admin	10.10.10.x/24	192.168.238.0/24	10.10.10.1-qrouter 10.10.10.2-DHCP 192.168.238.5-DHCP 192.168.238.6-qrouter	10.10.10.3-254	os-ssh-ping: Allow: SSH Allow: ICMP/Ping
Dev-Cloud #For Cloud Developer Group	dev-cloud-admin1 dev-cloud-user1	10.10.15.0/24	192.168.238.0/24	10.10.15.1-qrouter 10.10.15.2-DHCP 192.168.238.5-DHCP 192.168.238.6-qrouter	10.10.15.3-254	dev-cloud-sec-group1: Allow: SSH Allow: ICMP/Ping Allow: HTTP/HTTPS
Sales-Rockies #For Rockies Region Sales Group	sales-rockies-admin1 sales-rockies-user1	10.10.20.0/24	192.168.238.0/24	10.10.20.1-qrouter 10.10.20.2-DHCP 192.168.238.5-DHCP 192.168.238.6-qrouter	10.10.20.3-254	sales-rockies-sec-group1: Allow: SSH Allow: ICMP/Ping Allow: HTTP/HTTPS
DDI-1 #Developer Desktop Inf. Group	ddi-admin1 ddi-user1	10.10.25.0/24	192.168.238.0.24	10.10.25.1-qrouter 10.10.25.2-DHCP 192.168.238.5-DHCP 192.168.238.6-qrouter	10.10.25.3-254	ddi-sec-group1: Allow: SSH Allow: ICMP/Ping Allow: VNC Allow: HTTP/HTTPS

*Note: There are ‘system’ tenants such as “services” as well as system users not listed here

example.com – Tenant/Network Layout

High-Level View





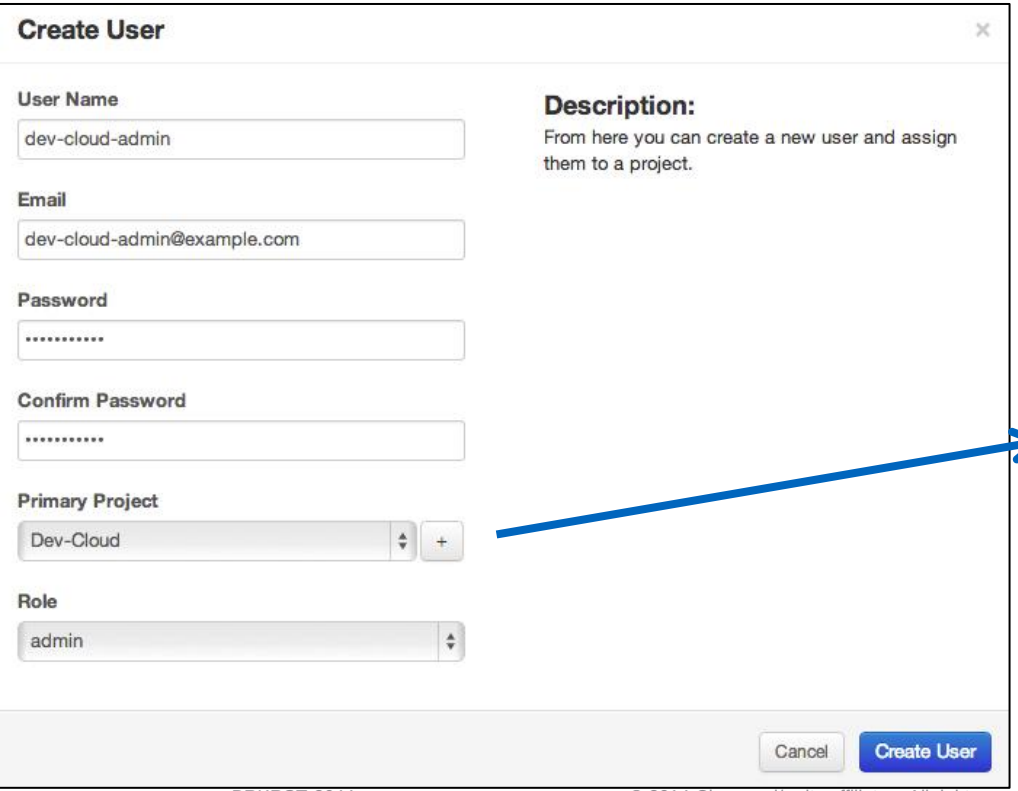
Setup the Project/Tenants

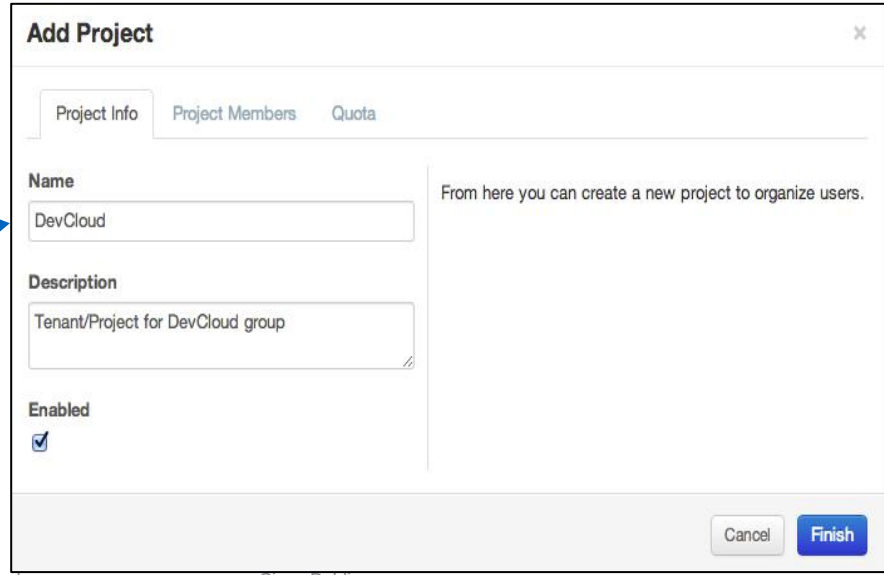
Reference

- Create users and projects
- Create security groups and rules
- Create Quantum networks, subnets and add router interface(s)
- Upload project-specific images into glance (if different from ‘shared’ images)
- Create volumes (nova-volume/Cinder)
- Launch instances

Create Project/Tenant Users

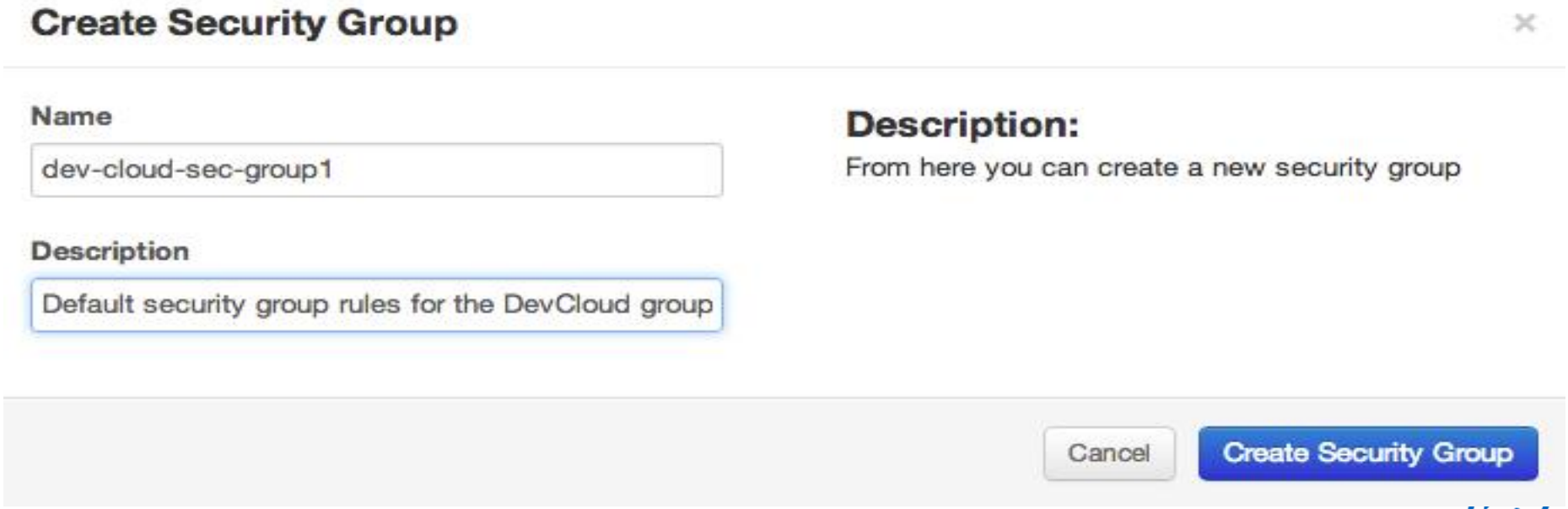
- Dashboard > Admin Tab > Users > Create User
- Add “member” users
- Repeat for each Project





Create Security Group

- Logged in as Project Admin user
- Dashboard > Access & Security > Create Security Group



Edit Rules

- Logged in as Project Admin user
- Dashboard > Access & Security > Edit Rules

Security Group Rules

+ Add Rule | Delete Rules

	IP Protocol	From Port	To Port	Source	Actions
☐	TCP	80	80	0.0.0.0/0 (CIDR)	Delete Rule
☐	ICMP	-1	-1	0.0.0.0/0 (CIDR)	Delete Rule
☐	TCP	22	22	0.0.0.0/0 (CIDR)	Delete Rule
☐	TCP	443	443	0.0.0.0/0 (CIDR)	Delete Rule

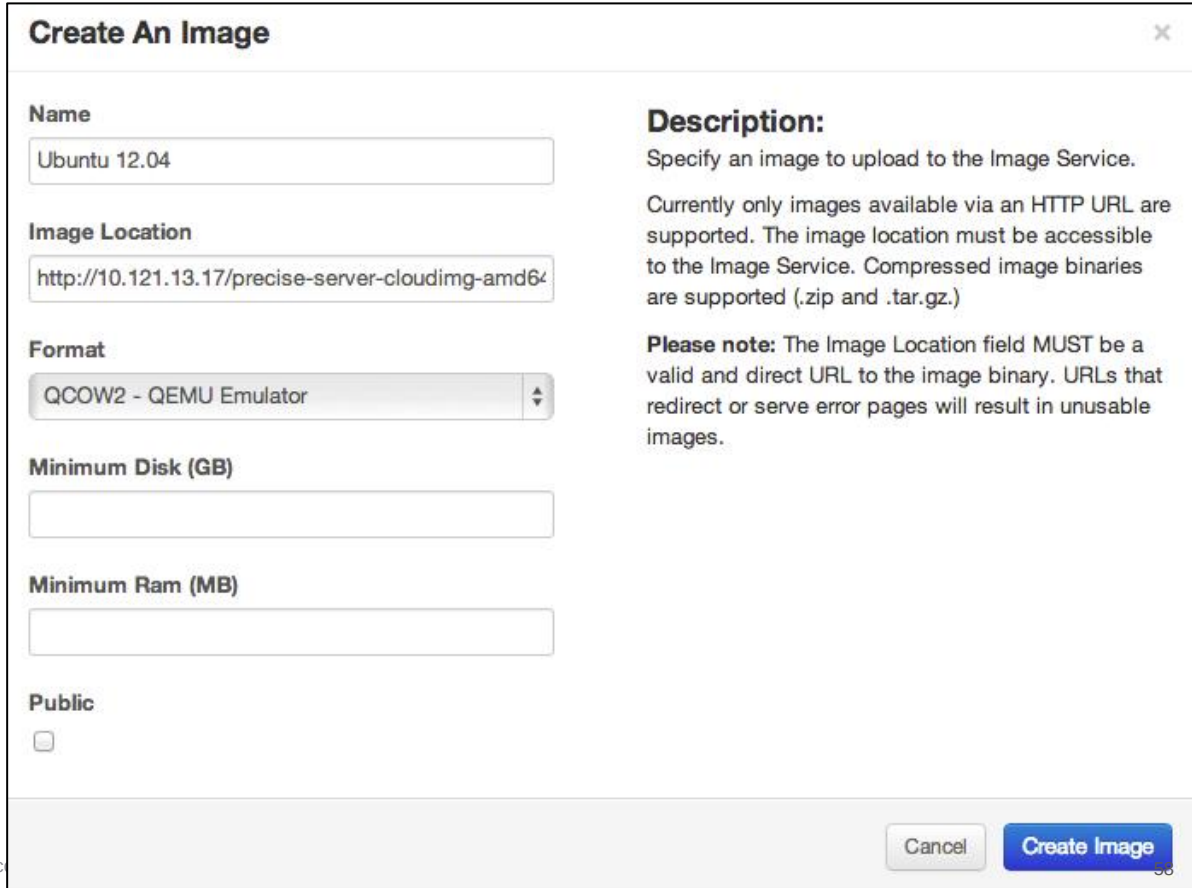
Displaying 4 items

Quantum Setup for Project - DevCloud

```
# Source the "openrc" file to export authentication/tenant info for dev-cloud-admin
root@control-server:~# source openrc-dev-cloud
# Create a Quantum network
root@control-server:~# quantum net-create dev-cloud-priv1
# Create internal (data) network used by the project DevCloud
root@control-server:~# quantum subnet-create --name dev-cloud-10.10.15.x dev-cloud-priv1
10.10.15.0/24 --dns_nameservers list=true 10.121.12.10
# Add Quantum router interface to previously create subnet
root@control-server:~# quantum router-interface-add os-router-1 dev-cloud-10.10.15.x
```

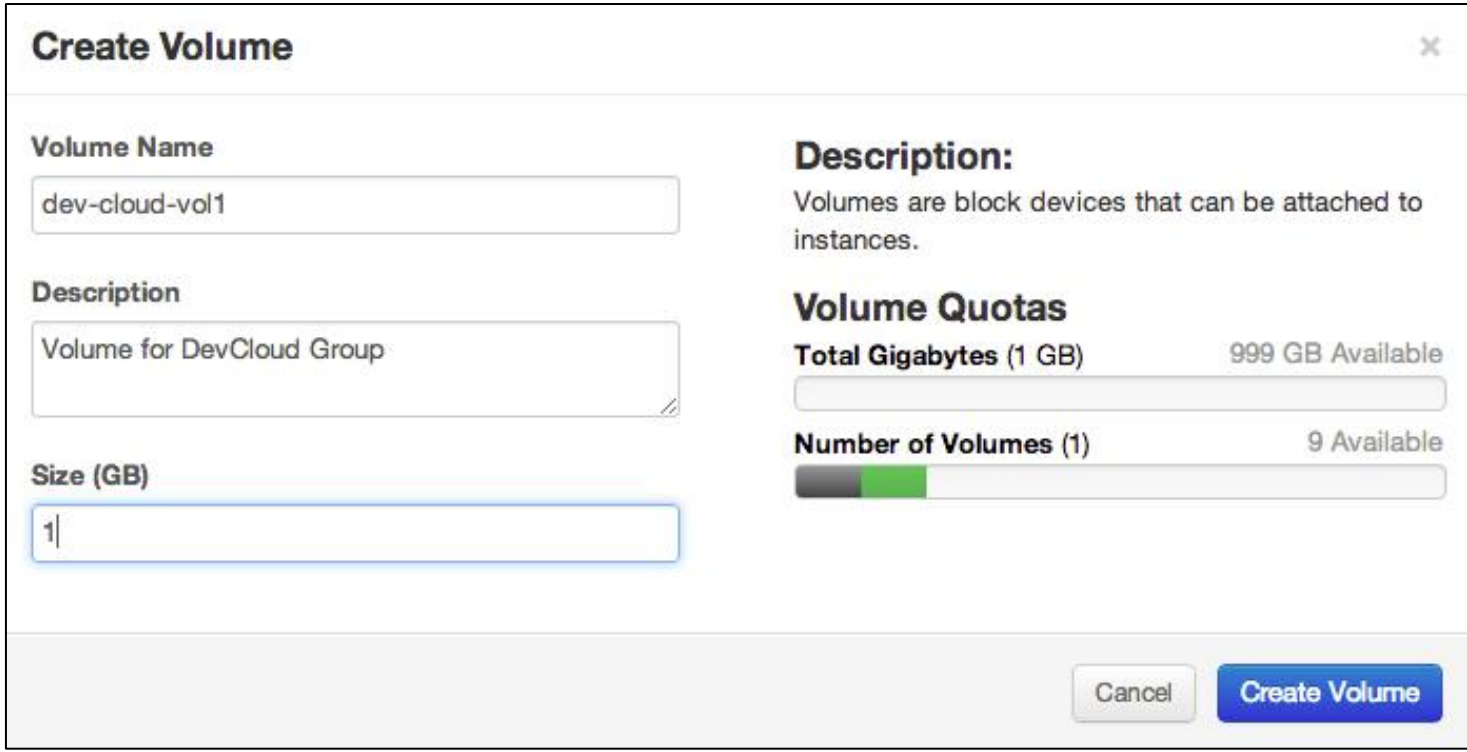
Upload Image into Glance

- Logged in as Project Admin user
- Dashboard > Images & Snapshots > Create Image



Create a Volume

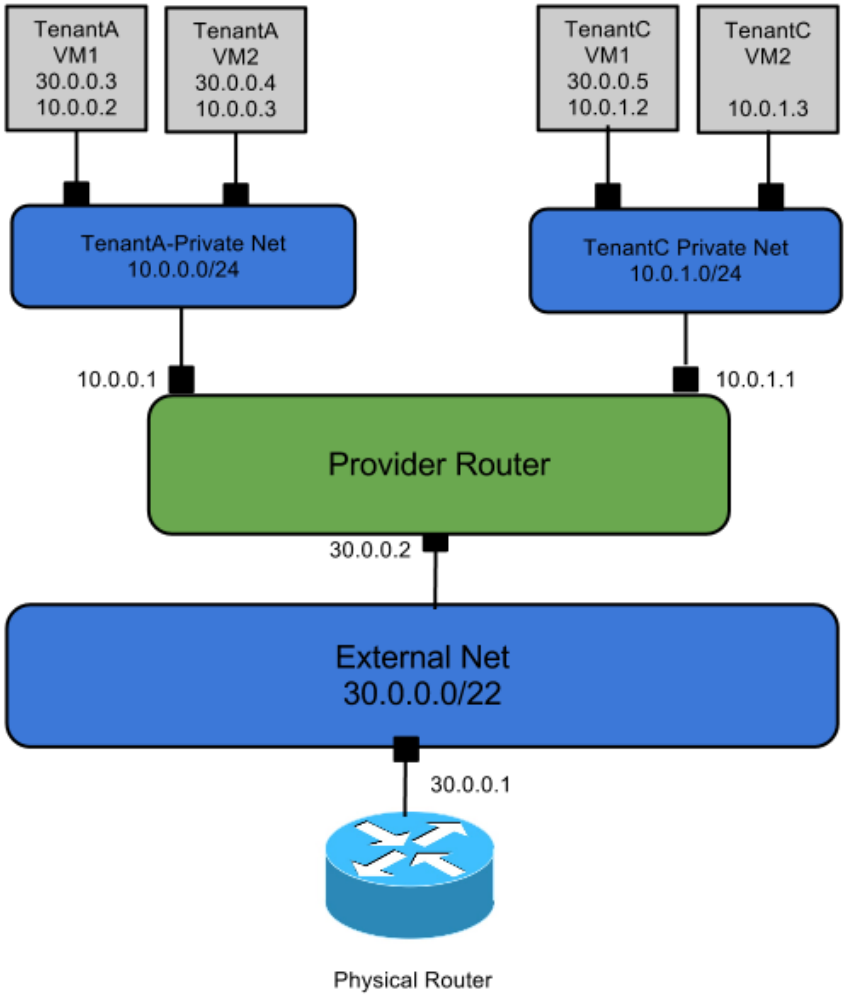
- Logged in as Project Admin user
- Dashboard > Volumes > Create Volume



Understanding Quantum + OVS for example.com

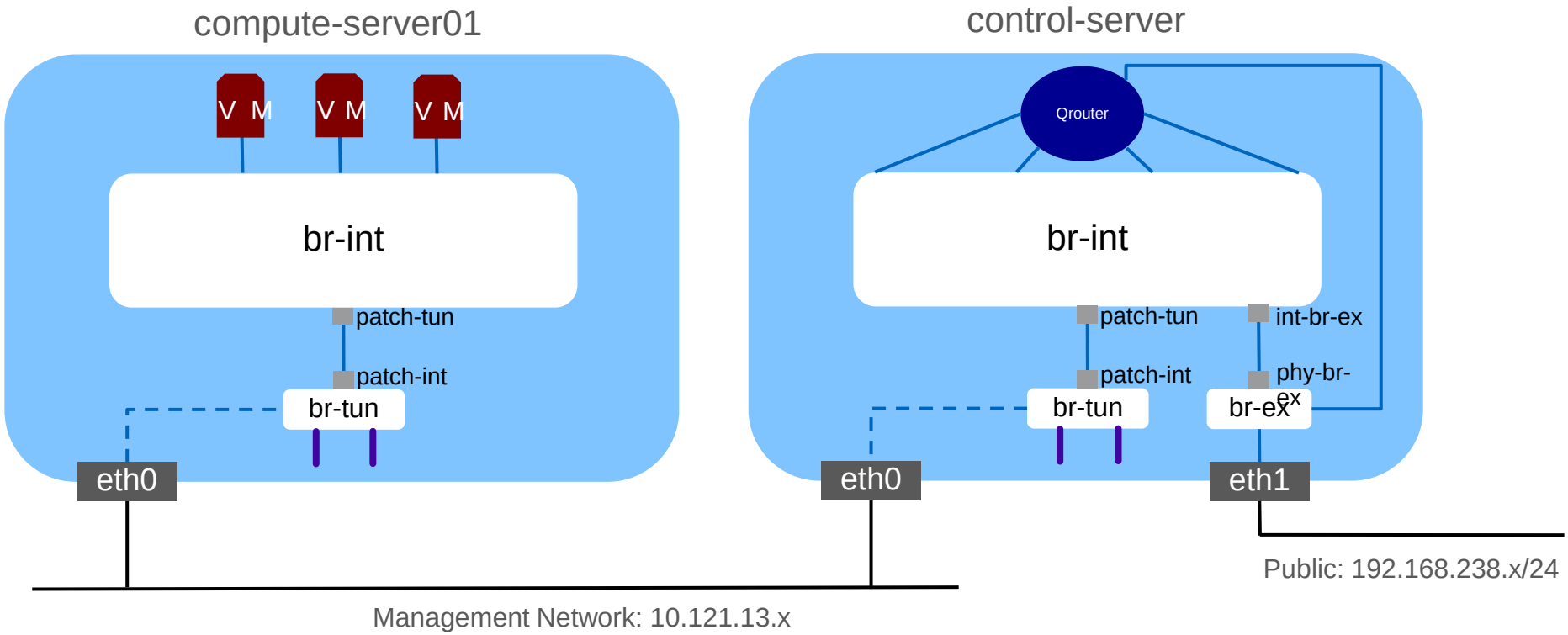
OpenStack Networking Deployment Use Cases

- http://docs.openstack.org/trunk/openstack-network/admin/content/use_cases.html
- Single Flat
- Multiple Flat
- Mixed Flat and Private Network
- Provider Router with Private Networks – This is basically what we are using in our example
- Per-tenant Routers with Private Networks



example.com – Tenant/Network Layout

Host View

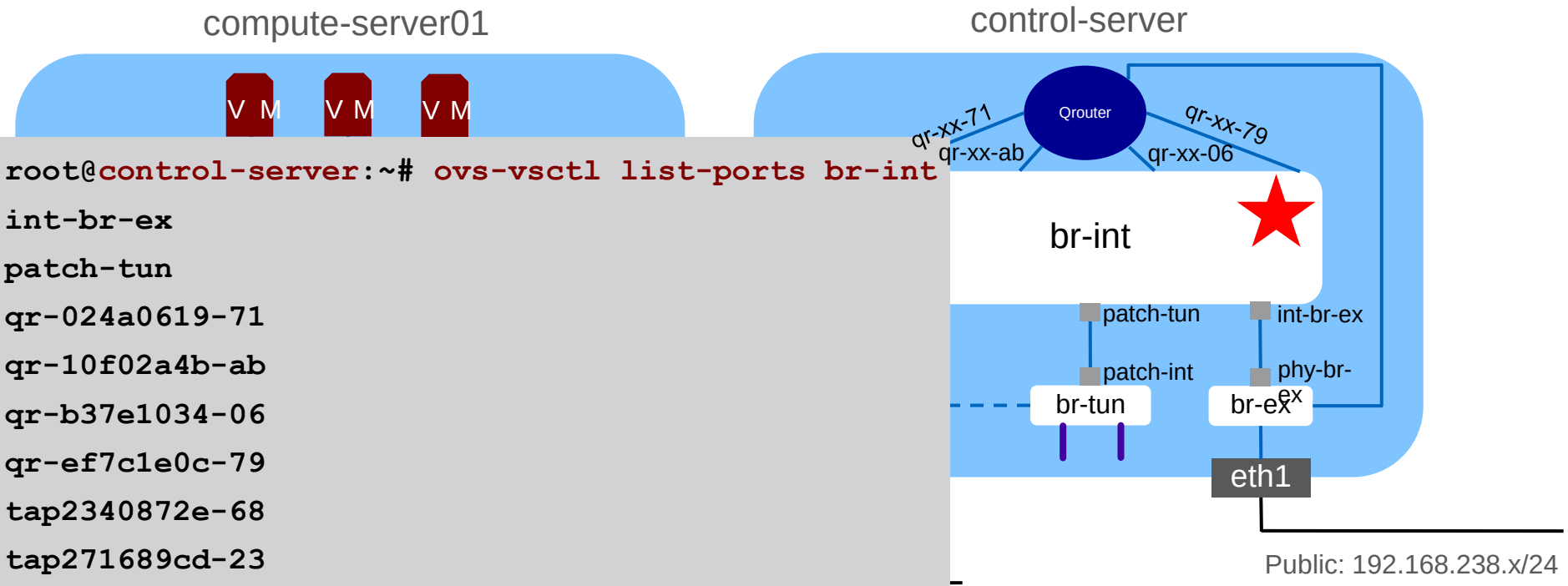


A Few of my Favourite Quantum/OVS Commands

- ovs-vsctl show
- ovs-vsctl list-ports <BRIDGE>
- brctl show
- quantum port-list
- quantum port-show <id-from-port-list>
- quantum router-list
- ip netns exec qrouter-<router-id-from-router-list> ip addr #must have IP namespaces enabled

example.com – OVS Bridge/Quantum Router

"br-int" view




```
root@control-server:~# ovs-vsctl list-ports br-int
int-br-ex
patch-tun
qr-024a0619-71
qr-10f02a4b-ab
qr-b37e1034-06
qr-ef7c1e0c-79
tap2340872e-68
tap271689cd-23
tap3fe91abf-c8
tap60a25081-14
tap6d3911a5-44
```

example.com – OVS Bridge/Quantum Router

"br-int" view qr-xx & tapxx

compute-server01

control-server

```
root@control-server:~# ovs-vsctl list-ports br-int
int-br-ex
patch-tun
qr-024a0619-71
qr-10f02a4b-ab
qr-b37e1034-06
qr-ef7c1e0c-79
tap2340872e-68
tap271689cd-23
tap3fe91abf-c8
tap60a25081-14
tap6d3911a5-44
```

bridge-to-router

A tap interface for each network used for DHCP service:

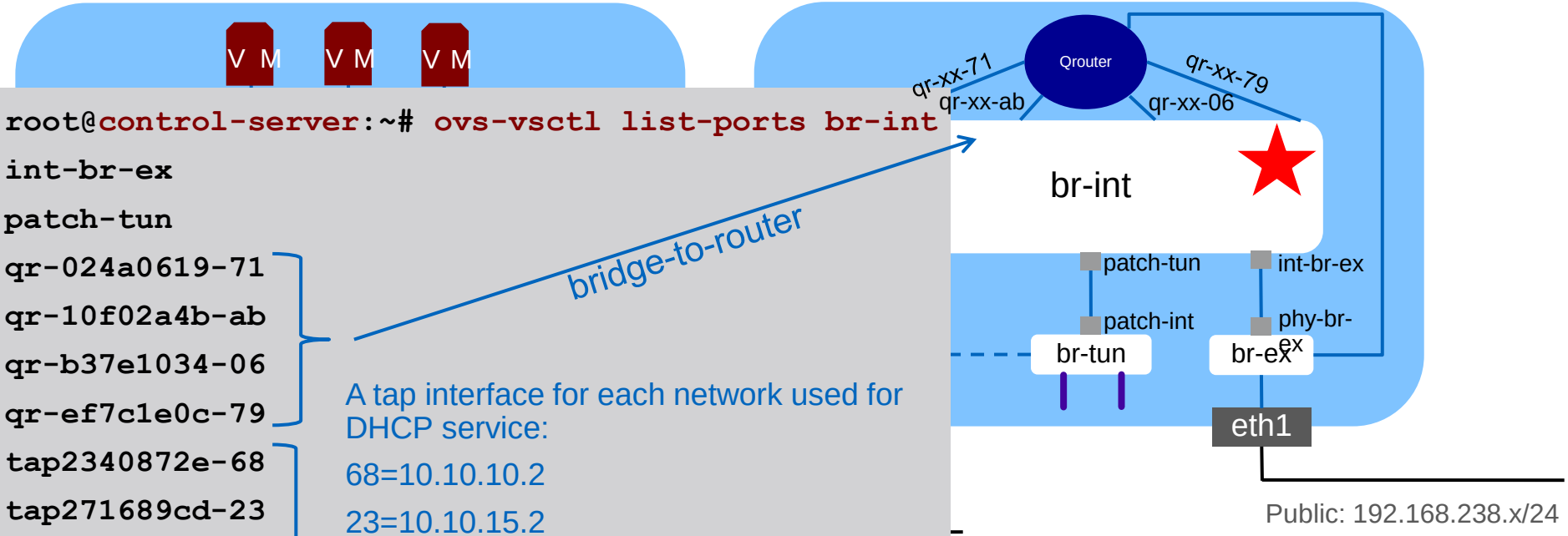
68=10.10.10.2

23=10.10.15.2

c8=192.168.238.5

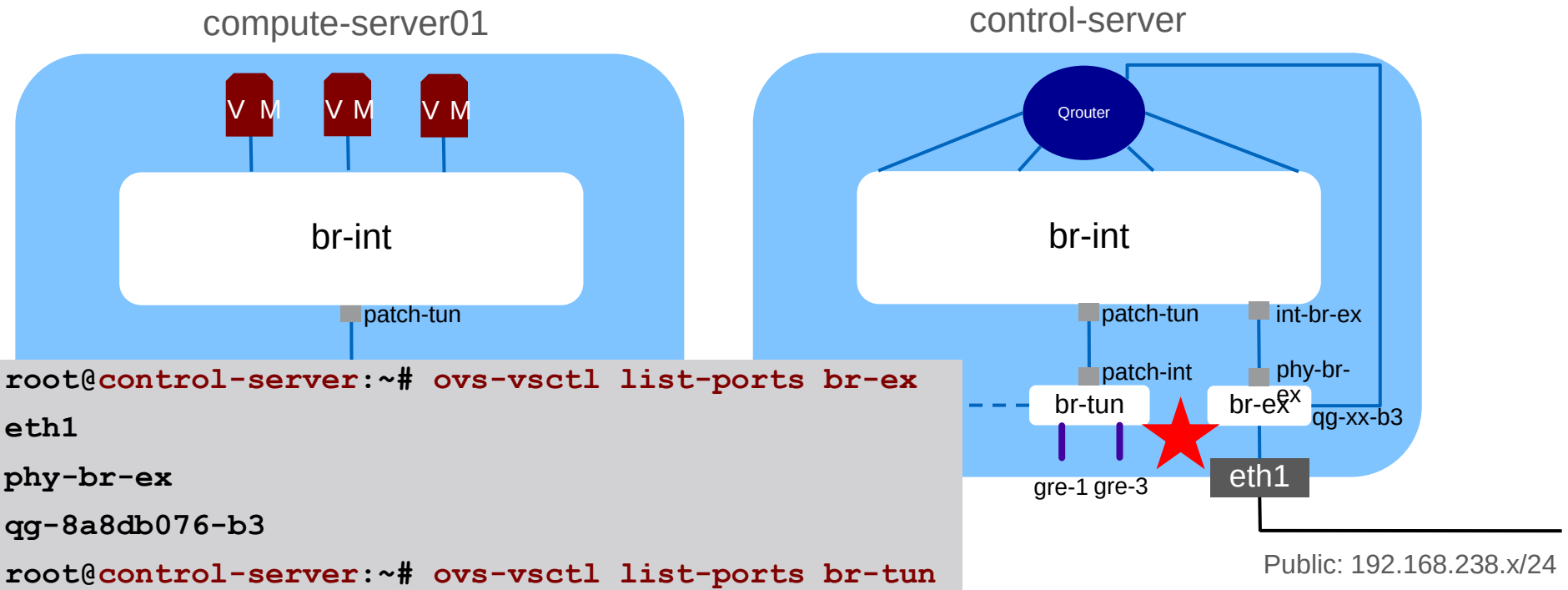
14=10.10.20.2

44=10.10.25.2



example.com – OVS Bridge/Quantum Router

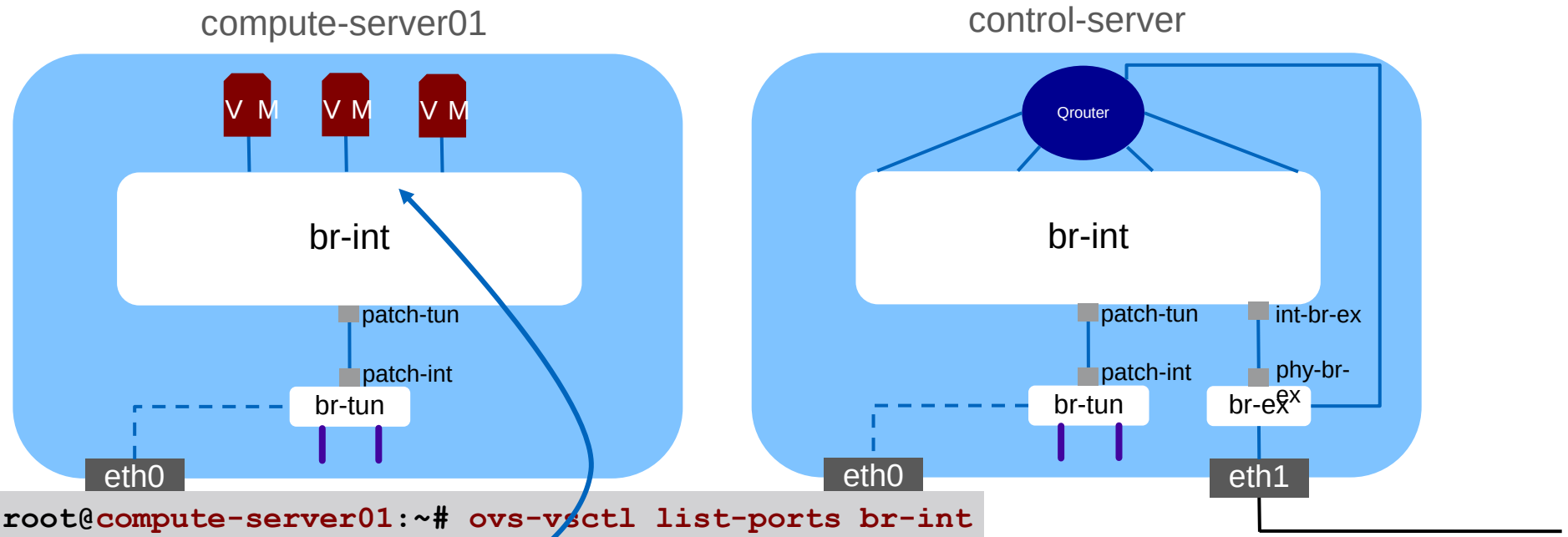
"br-ex" & br-tun view




```
root@control-server:~# ovs-vsctl list-ports br-ex
eth1
phy-br-ex
qg-8a8db076-b3
root@control-server:~# ovs-vsctl list-ports br-tun
gre-1
gre-3
patch-int
```

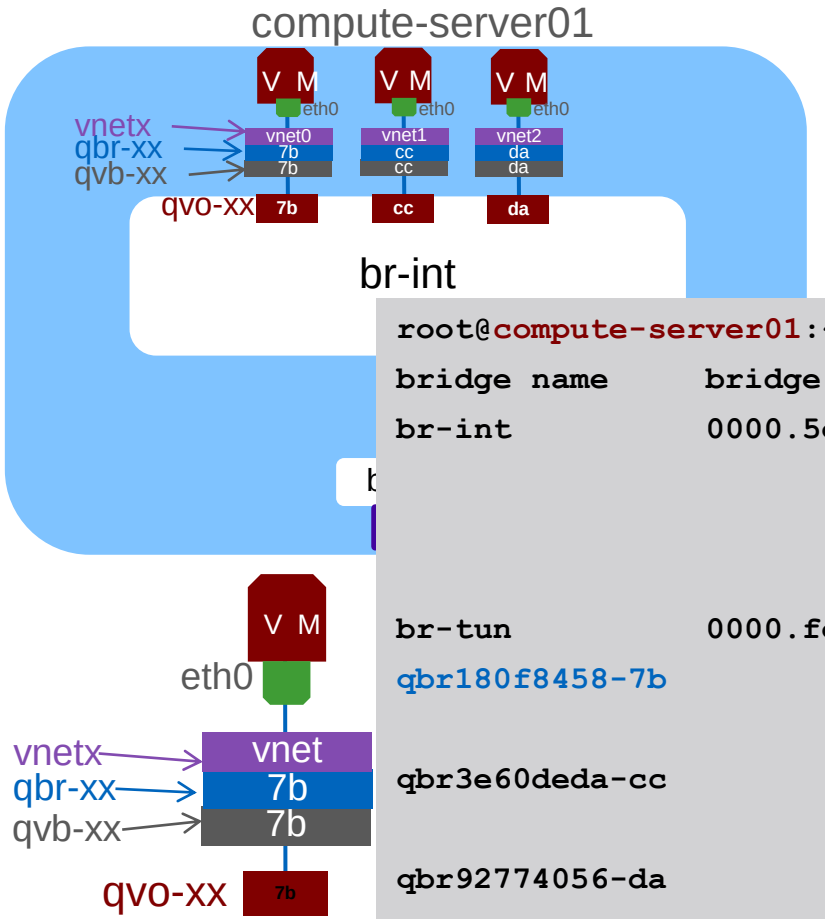
example.com – OVS Bridge/Quantum Router

compute-server01 “br-int” view




```
root@compute-server01:~# ovs-vsctl list-ports br-int
patch-tun
qvo180f8458-7b
qvo3e60deda-cc
qvo92774056-da
```

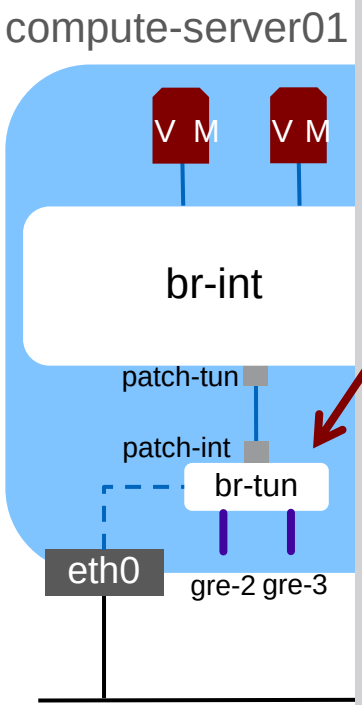
example.com – OVS Bridge/Quantum Router



*Thanks to Etsuji Nakai for the original detailed overview of OVS/Quantum ports : http://www.slideshare.net/enakai/how-quantum-configures-virtual-networks-under-the-hood

```
root@compute-server01:~# brctl show
bridge name     bridge id               STP enabled     interfaces
br-int          0000.5e15d719a548       no              int-br-ex
                                                        qvo180f8458-7b
                                                        qvo3e60deda-cc
                                                        qvo92774056-da
br-tun          0000.febc48d02540       no
qbr180f8458-7b          8000.1a425eeda354       no              qvb180f8458-7b
                                                        vnet0
qbr3e60deda-cc          8000.8a70b498c8ce       no              qvb3e60deda-cc
                                                        vnet2
qbr92774056-da          8000.3e21bdf7dd5b       no              qvb92774056-da
                                                        vnet1
```

example.com – OVS Bridge/Quantum Router



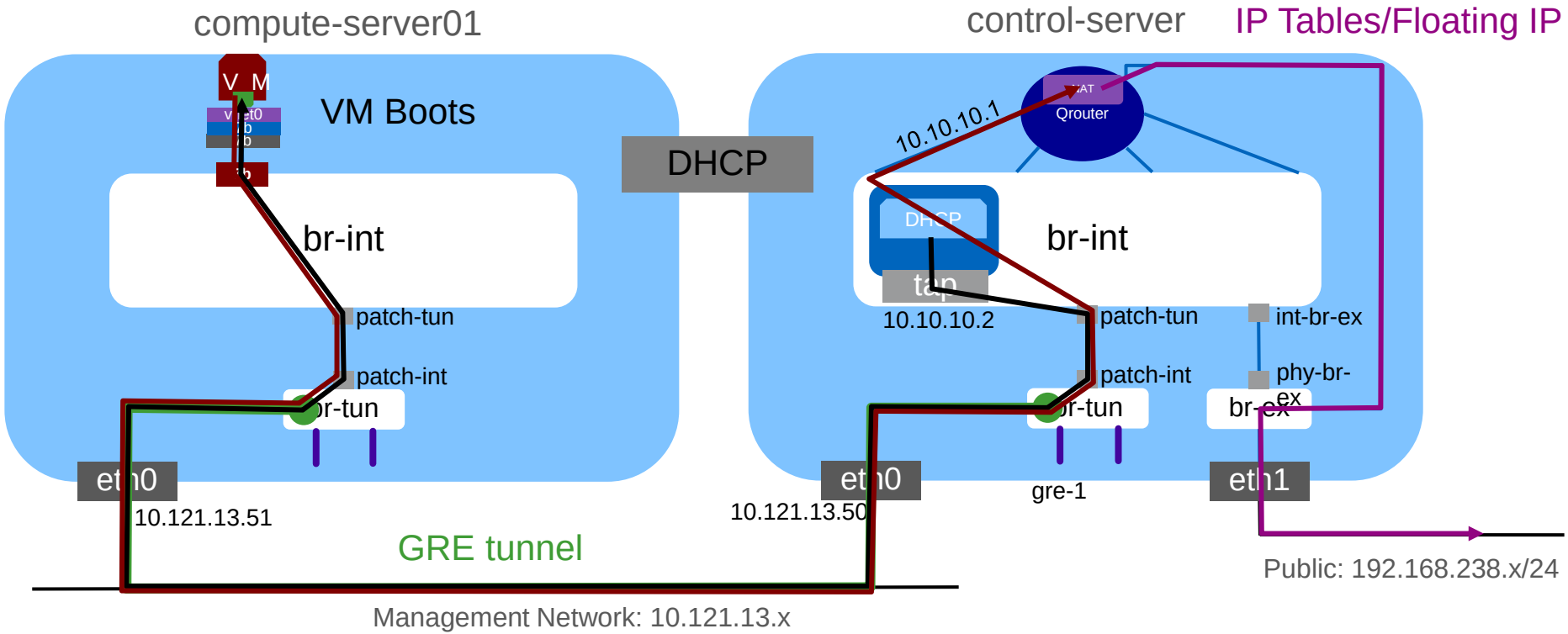

```
root@compute-server01:~# ovs-vsctl show
ac44a899-5f10-4ff9-8dad-902fa7c10e5e
...
Bridge br-tun
        Port "gre-2"
            Interface "gre-2"
                type: gre
                options: {in_key=flow, out_key=flow, remote_ip="10.121.13.50"}
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}
        Port "gre-3"
            Interface "gre-3"
                type: gre
                options: {in_key=flow, out_key=flow, remote_ip="10.121.13.52"}
        Port br-tun
            Interface br-tun
                type: internal
```

control-server

compute-server02

example.com – Basic VM Traffic Flow

High-Level Walk-Through






Running Applications

Multiple Paths to Managing Images/Apps

- Docker:
  - http://www.docker.io/
  - https://wiki.openstack.org/wiki/Docker
- VMBuilder:
  - http://docwiki.cisco.com/wiki/OpenStack:VM_Build
  - https://launchpad.net/vmbuilder
  - https://help.ubuntu.com/12.04/serverguide/jeos-and-vmbuilder.html
- Disk Image Builder:
  - https://github.com/stackforge/diskimage-builder
- Heat – Template based orchestration engine :
  - https://wiki.openstack.org/wiki/Heat
  - https://github.com/openstack/heat
- Salt Cloud
  - https://github.com/saltstack/salt-cloud
- Baseline images + automated application deployment (scripts, Puppet, Chef)
- Template images – Prebuilt with apps installed and deployed from Glance

VMBuilder

Reference

- https://help.ubuntu.com/12.04/serverguide/jeos-and-vmbuilder.html
- Build images from KVM installed machine
- Create a configuration file and run vmbuilder:

```
# vmbuilder kvm ubuntu --hostname=base2 \
> --destdir=/var/lib/libvirt/images/base2
```

- Or via full CLI with options:

```
vmbuilder kvm ubuntu --suite precise --flavour virtual \
--arch amd64 -o --libvirt qemu:///system --ip 10.121.13.77 \
--hostname base2 --part vmbuilder.partition \
--user localadmin --name localadmin --pass ubuntu \
-m 2048 --cpus 1 --addpkg unattended-upgrades \
--addpkg openssh-server --addpkg puppet --addpkg git
```

```
root@builder:~# more /etc/vmbuilder.cfg
[DEFAULT]
tmpfs = suid,dev,size=2G
arch    = amd64
domain = example.com
ip      = 10.121.13.77
mask    = 255.255.255.0
net     = 10.121.13.0
bcast   = 10.121.13.255
gw      = 10.121.13.1
dns     = 10.121.12.10
user    = localadmin
name    = localadmin
pass    = ubuntu
firstboot = /etc/vmbuilder/firstscripts/firstboot.sh
[kvm]
libvirt = qemu:///system
bridge = virbr0
virtio_net = true
mem     = 2048
cpus    = 2
[ubuntu]
proxy = http://10.129.16.14:8080
suite = precise
flavour = virtual
#install-mirror = http://10.121.13.17:3142/
components = main,universe
addpkg = openssh-server, unattended-upgrades, git, vim, puppet
```

Puppet on Baseline Instances

- Puppet is installed via baseline image or manually installed
- Puppet master or local puppet (masterless) is built and manifests defined
  - Use same PM as the OpenStack build used or your production PM(s) for apps
- Puppet agent runs (or local puppet apply) and apps for that instance are installed and configured
  - Alternatively, install via puppet modules: http://forge.puppetlabs.com/
- Test apps

Puppet Agent Run – Example w/LAMP

Reference

- On Puppet Master - /etc/puppet/manifests/site.pp

```
# Nodes for web server instances
node 'sales-web-01' {
    include lamp
}
```

- LAMP layout on PM:

```
root@build-server:~# tree /etc/puppet/modules/lamp/
/etc/puppet/modules/lamp/
├── files
│   ├── apache2.conf
│   ├── index.php
│   └── php5.conf
└── manifests
    └── init.pp
```

- Puppet Agent run on instance:

```
Info: Applying configuration version '1363712915'
Debug: /Stage[main]/Lamp/Exec[mysqlpasswd]/require: requires Package[mysql-server]
Debug: /Stage[main]/Lamp/Exec[mysqlpasswd]/require: requires Package[apache2]
Debug: /Stage[main]/Lamp/Exec[mysqlpasswd]/notify: subscribes to Service[mysql]
Debug: /Stage[main]/Lamp/Exec[mysqlpasswd]/notify: subscribes to Service[apache2]
Debug: /Stage[main]/Lamp/Service[apache2]/require: requires Package[apache2]
Debug: /Stage[main]/Lamp/Exec[userdir]/require: requires Package[apache2]
```




Monitoring

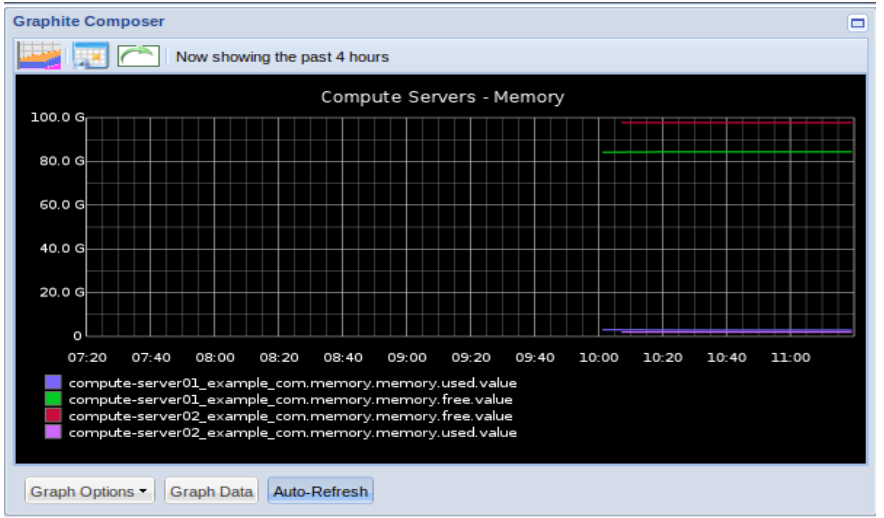
Basic Monitoring is Available

Nagios/Graphite/Collectd

- http://<build-server>/nagios3 - Health monitoring of OpenStack nodes

Host	Status	Last Check	Duration	Status Information
compute-server01	UP	2013-03-19 12:03:47	4d 1h 29m 7s	PING OK - Packet loss = 0%, RTA = 0.27 ms
compute-server02	UP	2013-03-19 12:04:57	4d 1h 28m 57s	PING OK - Packet loss = 0%, RTA = 0.31 ms
control-server	UP	2013-03-19 12:05:47	0d 1h 9m 8s	PING OK - Packet loss = 0%, RTA = 0.30 ms

- http://<build-server>:8190 – Main Graphite performance console
- http://<build-server>:8190/dashboard/ - User/Self-service performance console



- http://www.nagios.org/
- http://graphite.wikidot.com/
- http://collectd.org/

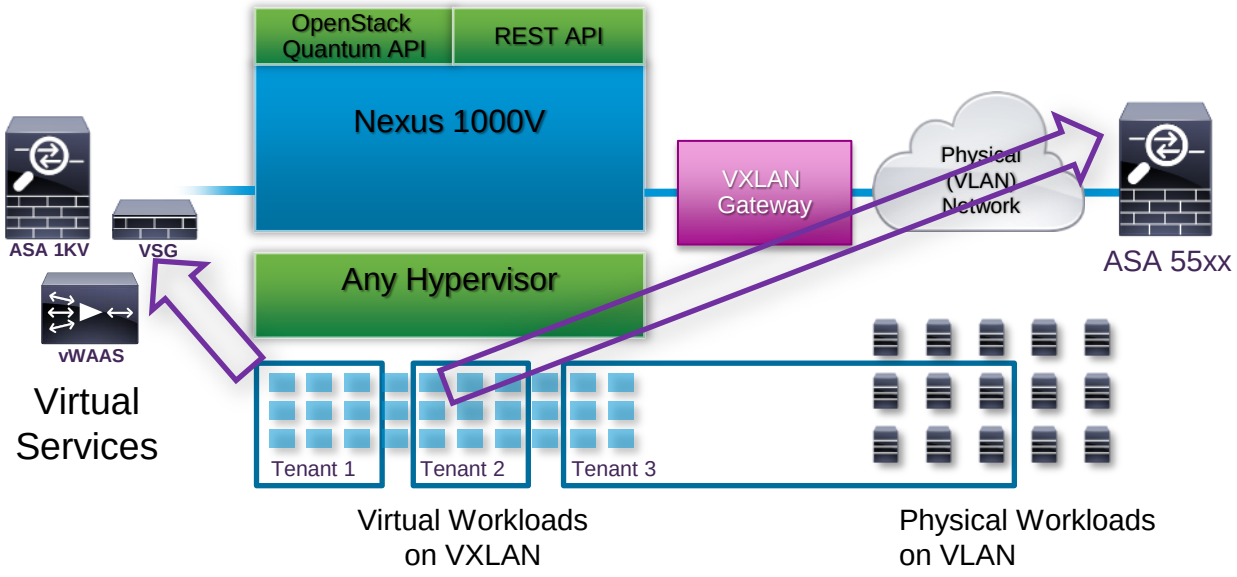



Cisco Product Integration - Nexus

Virtual Overlay Networking Cloud Solution

Building OpenStack based Clouds with Nexus 1000V

- **Scalable Multi-tenancy**
  - Tens of thousands of virtual ports, L2 networks
  - Hundreds of Hosts
  - Scalable segmentation: VXLAN
- **Common APIs**
  - Incl. OpenStack Quantum API's for cloud automation/orchestration
- **Virtual Services**
  - vPath for traffic steering / service chaining
  - VSG, ASA 1000V (cloud-ready security), vWAAS (application acceleration)
  - CSR 1000V (cloud router)
- **Multi-hypervisor**
  - ESX, Hyper-V, OpenSource Hypervisors (KVM/Xen)
- **Hybrid Use Cases** (Physical and Virtual)
  - VXLAN to VLAN GW



Tenant 1: virtual workloads protected by virtual firewall

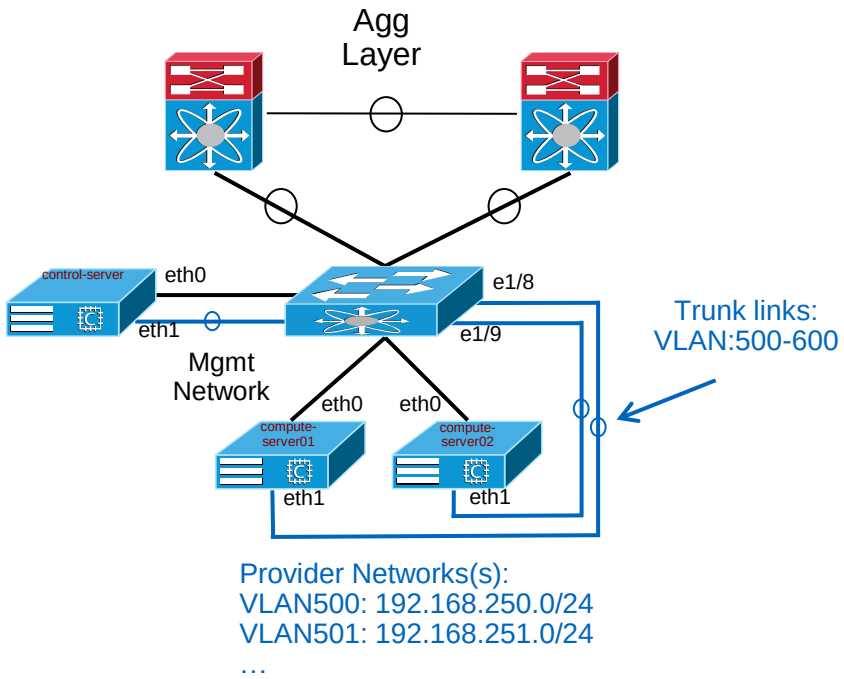
Tenant 2: virtual workloads protected by physical firewall (via VXLAN GW)

Tenant 3: virtual & physical workloads in same L2 domain (via VXLAN GW)

Nexus – Initial Support for OpenStack

- Nexus 1000
  - Based on Grizzly release
  - Red Hat and Ubuntu - KVM
  - 512 servers per VSM and scaling to future with federations
  - VLAN - 4096, VXLAN – 16000 segments, 32000 ports, 300+ veths/vem
  - Enhanced VXLAN – No multicast requirement in a VSM and in future across VSMs
  - VSM on any hypervisor or Nexus1010
  - CSR as the tenant router – integrated into OpenStack (VXLAN aware)
  - NAT is supported/overlapping IP support
- Nexus 3000 and Higher
  - http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps11541/data_sheet_c78-727737.html
- Cisco OpenStack Installer with Nexus Plugin: http://docwiki.cisco.com/wiki/OpenStack:Grizzly-Nexus-Plugin

Nexus Plugin Example Topology



- Stuff we care about in the site.pp relevant to the diagram:
  - Switch ports that connect to the eth1 on each compute node
  - That the appropriate interface on the controller is configured to trunk all of the same VLANs that will be used by instances (attached to eth1 on compute nodes)
  - That the uplinks from ToR to Agg layer switches has all of the trunks/VLANs configured ahead of time
- Multiple ToR switches and host FEX setups are supported

Nexus Plugin – site.pp Specifics

```
$public_interface           = 'eth0'
$external_interface         = 'eth1'
$ovs_vlan_ranges            = 'physnet1:500:600'
$ovs_bridge_uplinks         = ['br-eth1:eth1']
$ovs_bridge_mappings        = ['physnet1:br-eth1']
$quantum_core_plugin        = 'cisco'
$cisco_vswitch_plugin       = 'ovs'
$cisco_nexus_plugin         = 'nexus'
$nexus_config = {
                  '10.121.10.39' => {
                                      'compute-server01' => '1/8',
                                      'compute-server02' => '1/9'
                                    }
                }
$nexus_credentials = ['10.121.10.39/my_username/my_password']
$tenant_network_type = 'vlan'
```

Example Nexus Config – Pre First Instance Boot

```
interface Ethernet1/1
  description to N7k-agg-1
  switchport mode trunk
  switchport trunk allowed vlan 13,500-600

interface Ethernet1/2
  description to N7k-agg-2
  switchport mode trunk
  switchport trunk allowed vlan 13,500-600

interface Ethernet1/5
  description to control-server Management
  switchport mode trunk
  switchport trunk allowed vlan 13
  speed 1000

interface Ethernet1/6
  description to control-server Data
  switchport mode trunk
  switchport trunk allowed vlan 13,500-600
  speed 1000
```

```
interface Ethernet1/7
  description to compute-server01 Management
  switchport mode trunk
  switchport trunk allowed vlan 13
  speed 1000

interface Ethernet1/8
  description to compute-server01 Data
  switchport mode trunk
  speed 1000

interface mgmt0
  ip address 10.121.10.39/24
```

Quantum/Neutron Setup

```
# Source the "openrc" file to export authentication/tenant info

root@control-server:~# source openrc

# Create a Quantum/Neutron provider network

root@control-server:~# quantum net-create vlan500 --provider:network_type vlan --
provider:physical_network physnet1 --provider:segmentation_id 500 --shared --
router:external=True

# Create a Quantum/Neutron subnet for the provider network. Set starting address to be higher
than upstream DC agg-layer HSRP addresses (.1, .2, .3)

root@control-server:~# quantum subnet-create --name subnet500 --allocation-pool
start=192.168.250.10,end=192.168.250.250 vlan500 192.168.250.0/24 --dns_nameservers list=true
10.121.12.10
```

Before/After of Nexus Configuration

Before First Instance Launch:

```
interface Ethernet1/7
  description to compute-server01 Management
  switchport mode trunk
  switchport trunk allowed vlan 13
  speed 1000

interface Ethernet1/8
  description to compute-server01 Data
  switchport mode trunk
  speed 1000

interface mgmt0
  ip address 10.121.10.39/24
```

After First Instance Launch:

```
vlan 500
  name q-500

interface Ethernet1/7
  description to compute-server01 Management
  switchport mode trunk
  switchport trunk allowed vlan 13
  speed 1000

interface Ethernet1/8
  description to compute-server01 Data
  switchport mode trunk
  switchport trunk allowed vlan 500
  speed 1000
```




Havana

Cisco OpenStack Installer for Havana

- A much simpler process for Havana – Using the Puppet Hiera Data Model: http://docwiki.cisco.com/wiki/Openstack:Havana-Openstack-Installer

- A basic AIO model is as easy as:

```
cd /root && git clone -b havana https://github.com/CiscoSystems/puppet_openstack_builder
&& cd puppet_openstack_builder && git checkout h.0
export vendor=cisco
export scenario=all_in_one
cd puppet_openstack_builder/install-scripts
./install.sh
```

- Havana HA Model: http://docwiki.cisco.com/wiki/OpenStack_Havana_Release:_High-Availability_Manual_Deployment_Guide

DEMO

Conclusion

- OpenStack is for real and maturing at a rapid pace
- Many different players involved and it is evolving rapidly
- Many sources of information on the stuff we did not cover (Storage, Security, HA, etc.)
- Align yourself with market leaders who have strong partnerships
- There is a massive focus on ‘getting it installed’ but the real work starts after that
  - Scale
  - HA
  - Application deployment
  - Worst of all – Upgrades
- Start now by deploying it small scale and learning the parts
- Get involved in the community – open source enjoys the major advantage of feature velocity




Q & A

Complete Your Online Session Evaluation

Give us your feedback and receive a Cisco Live 2014 Polo Shirt!

Complete your Overall Event Survey and 5 Session Evaluations.

- Directly from your mobile device on the Cisco Live Mobile App
- By visiting the Cisco Live Mobile Site www.ciscoliveaustralia.com/mobile
- Visit any Cisco Live Internet Station located throughout the venue

Polo Shirts can be collected in the World of Solutions on Friday 21 March 12:00pm - 2:00pm

Learn online with Cisco Live!

Visit us online after the conference for full access to session videos and presentations.
www.CiscoLiveAPAC.com


CISCO
TM