RHCSA for Openstack - Vugt

Throughout this, commands complain that they are deprecated and replaced by python-openstackclient Hopefully, this will not be an issue, and we are told to ignore it. Here is the link to more info on that package: https://github.com/openstack/python-openstackclient

Based on a minimal setup for OpenStack
- Set up as virtual machines is recommended

server1: AiO setup initially, controller node later in this course
- 4GB RAM, 10 GB disk, CentOS
- 2 Network interfaces

server2: Used for specific purposes in some labs
- 1GB RAM, 10 GB disk, CentOS
- 2 Network interfaces

workstation (optional): used to control the cloud environment
- 1GB RAM, 10 GB disk, CentOS "Server with GUI" installation

Installing:
- Highlight Install CentOS and hit <tab> to get cmd line option ("full configuration options")
- add "net.ifnames=0 biosdevnames=0" at the end so network hardware will show up as eth0 instead of eno353487
- output what was used to file with "grub2-mkconfig -o /boot/grub2/grub.cfg"

Nova Compute - Part of the OS talking to the hypervisor to decide where a VM will run

Swift Object Storage - physical HDs are tied to physical machines. This isn't.
Elastic alternative to local storage
Creates binary objects that are dispersed/replicated between storage nodes
Binary = easily managed, replicated= "who cares about where they exist?"
If a storage node goes down - who cares, because it is replicaed elsewhere
As long as you have a minimum of 3 physical nodes you are fine
Swift proxy is used by Swift to access filesystems. Swift itself doesnt access filesystems

Glance Images (instances) to deploy VMs from
Users don't want to install and do settings, they just want to spin something up! (think AWS)
Cirros is a 13MB image used for testing
For different Linux OS's, be sure to use ready-to-deploy cloud images when possible
You might need custom images, particularly in a corporate environment
Typically, with the image you would hava a template to specifiy hardware (flavors)

Cinder Volume You booted from an image so you need something to write on- here it is
Otherwise write is local to the machine hosting the image, and the write will be temporary
Things move around - the write cannot be guranteed, and Cinder provides **persistent storage**.
Often uses Swift object store for a backend
"No local storage! It's the last thing you want"

Neutron Networking (SDN, logical switches)
In context, SDN answers two things: tenant traffic isolation and single logical broadcast in multi-router physical infrastructure.
You don't care where VMs exist, you only care that they are on the same network
Overlaying logical network and underlaying physical network

Horizon Web user interface - dashboard, for tenants and administrators
For full control, you still need command line access
In terms of Red Hat exams, you can configure in Horizon, but you will fail since you can't determine if it works well

Keystone Identity - users and roles
Authentication tokens like kerberos
Stores in a database (MariaDB)
Central point to get info about cloud components:
Tools: keystone service-list, keystone user-list, keystone endpoint-list

Message Broker RabbitMQ (default) or Qpid
Is like an smtp server for email, ensures messages between cloud components in an orderly way
Needs to be in an HA configuration so when it goes down another can take it's place

```
[root@server1 ~]# ls
anaconda-ks.cfg  answers.txt  keystonerc_admin  keystonerc_user  newcompute.txt  packstackca
[root@server1 ~]# source keystonerc_admin
[root@server1 ~(keystone_admin)]# keystone service-list
/usr/lib/python2.7/site-packages/keystoneclient/shell.py:65: DeprecationWarning: The keystone CLI is deprecated in favor of py
thon-openstackclient. For a Python library, continue using python-keystoneclient.
  'python-keystoneclient.', DeprecationWarning)
+----------------------------------+-----------+--------------+----------------------------------+
|                id                |    name   |     type     |           description            |
+----------------------------------+-----------+--------------+----------------------------------+
| a2c1b507f30d4273b60fb906557729ff | ceilometer |   metering   |   Openstack Metering Service   |
| b487167641684d108915996b7c59c3e3 |   cinder   |    volume    |        Cinder Service          |
| 20666a6b744142dc94025c68471e6e39 |  cinderv2  |   volumev2   |       Cinder Service v2        |
| b4f0ef7ef4174540b55c6a4c6058eb7c |   glance   |    image     |    OpenStack Image Service     |
| 2853db69d4ad463ea33459caaa6400ec |  keystone  |   identity   |   OpenStack Identity Service   |
| ea5b86e5ffc34979816669e8cd96fd4a |  neutron   |   network    |   Neutron Networking Service   |
| a0cf4e62d5c3494c85fa83cc8cb1c32b |    nova    |   compute    |   Openstack Compute Service    |
| 2eb391c03a5f4429834b2db5112dd9f8 |  nova_ec2  |     ec2      |         EC2 Service            |
| 63b326eb9bf14a55bd6725592cc4f2c6 |   novav3   |   computev3  |  Openstack Compute Service v3  |
| ea0c4c1490c04b28b80deb446d4a826b |   swift    | object-store | Openstack Object-Store Service |
| 61b6b311acbc461dbf7e748ef12644c3 |  swift_s3  |      s3      |     Openstack S3 Service       |
+----------------------------------+-----------+--------------+----------------------------------+
[root@server1 ~(keystone_admin)]# keystone user-list
/usr/lib/python2.7/site-packages/keystoneclient/shell.py:65: DeprecationWarning: The keyst
hon-openstackclient. For a Python library, continue using python-keystoneclient.
  'python-keystoneclient.', DeprecationWarning)
+----------------------------------+------------+---------+----------------------------+
|                id                |    name    | enabled |           email            |
+----------------------------------+------------+---------+----------------------------+
| 87f551f46a0943cb930f96c3db63fef5 |    adm     |   True  | adm@server1.example.com    |
| 4d72def1ea054cd5b3e885c960b68704 | adm-nieuw  |   True  |                            |
| b328b494231640ed9936288a3417255a |   admin    |   True  |      root@localhost        |
| 295b25788cdc4f0db7fddd48413ac2b9 | ceilometer |   True  |   ceilometer@localhost     |
| 3eeda54361584e1d8a21443e11ff88e8 |   cinder   |   True  |     cinder@localhost       |
| 16de98800609484c9d6083eeb91ff438 |   glance   |   True  |     glance@localhost       |
| dc0938f55e144d01895356925704b92c |  neutron   |   True  |     neutron@localhost      |
| 2d5fca5aa9514864a70762617b0fef46 |    nova    |   True  |       nova@localhost       |
| c30e567232884dedbcf04679c71f89ef |   swift    |   True  |      swift@localhost       |
| 73b8d0d6a9bb4a61ba372fe46064bdad |    user    |   True  | user@server1.example.com   |
+----------------------------------+------------+---------+----------------------------+
[root@server1 ~(keystone_admin)]# keystone endpoint-list
+----------------------------------+-----------+-----------------------------------------+----------------------------------+
|                id                |   region  |                publicurl                |                internalurl
|                                  |  adminurl |                                         |          service_id              |
+----------------------------------+-----------+-----------------------------------------+----------------------------------+
| 09c633e70f8446fbaa6382c9b7abf081 | RegionOne |          http://127.0.0.1:8774/v3       |           http://127.0.0.1:
774/v3           |  http://127.0.0.1:8774/v3  | 63b326eb9bf14a55bd6725592cc4f2c6 |
| 0d4a632f6e7d44d8bea1b525f1230fca | RegionOne |   http://192.168.4.10:8776/v2/%(tenant_id)s    |   http://192.168.4.10:8776/v
/%(tenant_id)s    | http://192.168.4.10:8776/v2/%(tenant_id)s | 20666a6b744142dc94025c68471e6e39 |
| 1159a398397c4ea7a09747e014602f38 | RegionOne |            http://192.168.4.10:9696     |           http://192.168.4.
0:9696           |  http://192.168.4.10:9696  | ea5b86e5ffc34979816669e8cd96fd4a |
| 19d535d8882d42d8916242c4182b40b1 | RegionOne |           http://192.168.4.10:8080      |           http://192.168.4.
0:8080           |  http://192.168.4.10:8080  | 61b6b311acbc461dbf7e748ef12644c3 |
| 416fa43282ce4448a50f2b1a4dac9a53 | RegionOne |        http://192.168.4.10:5000/v2.0    |        http://192.168.4.10:
000/v2.0         |  http://192.168.4.10:35357/v2.0 | 2853db69d4ad463ea33459caaa6400ec |
| 425f979275674e8baa3f0835e0aa2d16 | RegionOne | http://192.168.4.10:8080/v1/AUTH_%(tenant_id)s | http://192.168.4.10:8080/v1/
UTH_%(tenant_id)s |  http://192.168.4.10:8080  | ea0c4c1490c04b28b80deb446d4a826b |
| 5252439e0cc745e2abd0d9910c434d83 | RegionOne |           http://192.168.4.10:8777      |           http://192.168.4.
0:8777           |  http://192.168.4.10:8777  | a2c1b507f30d4273b60fb906557729ff |
| 7e047c0517b74ef883abfbc84f31a025 | RegionOne |           http://192.168.4.10:9292      |           http://192.168.4.
0:9292           |  http://192.168.4.10:9292  | b4f0ef7ef4174540b55c6a4c6058eb7c |
| aeaa70ac35f54d33bb70d74ea01a6dcf | RegionOne |   http://192.168.4.10:8776/v1/%(tenant_id)s    |   http://192.168.4.10:8776/v
/%(tenant_id)s    | http://192.168.4.10:8776/v1/%(tenant_id)s | b487167641684d108915996b7c59c3e3 |
| b2dcdf445b8945a58ba7cb95c559d9a3 | RegionOne |    http://192.168.4.10:8773/services/Cloud     |    http://192.168.4.10:8773/
ervices/Cloud     |  http://192.168.4.10:8773/services/Admin  | 2eb391c03a5f4429834b2db5112dd9f8 |
| c944e76f46b64c13af76b40ae6bbadbd | RegionOne |   http://192.168.4.10:8774/v2/%(tenant_id)s    |   http://192.168.4.10:8774/v
/%(tenant_id)s    | http://192.168.4.10:8774/v2/%(tenant_id)s | a0cf4e62d5c3494c85fa83cc8cb1c32b |
+----------------------------------+-----------+-----------------------------------------+----------------------------------+
```

Overall notes on these Openstack components:
- Typically (KVM) , Glance images will be /dev/vda and Cinder volumes will be /dev/vdb, etc
- If running on a different hypervisor, this will be different (perhaps familiar like /dev/sda, sdb)
- Nova used to include networking prior to 2010
- Swift is often used as a storage backend for Glance and Cinder
- If message broker or DB stuff goes down you're screwed

RESTful API and Python
Standardization in components
Oslo project standardized set of Python libraries to help make things uniform.

This allows all of the Openstack commands to have the same structure

Default hypervisor is KVM. but can use Qemu, HyperV, vSphere,

Other services
Ceilometer (telemetry) - which you have to have for measurements for things like billing
Heat (orchestration) - stacks (of instances) uses templates
Ironic (bare metal deployment) - you will need this for Nova and Neutron
Trove (DB) - database as a service
Saharah (data processing) - for big data needs
Oslo (standardization)
Others for DNS as a service etc...

Deployment: Using Packstack

- Uses answer files with Puppet to deploy a state on nodes
- Answer file generated before deployment, contains config settings

Steps:

- yum update –y
- Make sure name resolving to the host itself is setup in /etc/hosts
- yum install –y https://rdoproject.org/repos/rdo-release.rpm
- yum repolist
- systemctl disable NetworkManager; systemctl stop NetworkManager
- add net.ifnames=0 and biosdevname=0 to /etc/default/grub and run grub2-mkconfig to generate new bootloader
- Create a snapshot!
- yum install –y openstack-packstack
- packstack –h | less
- packstack --gen-answer-file /root/answers.txt
- Modify answers.txt parameters (see next slide)
- packstack --answer-file /root/answers.txt
- Wait five minutes to ensure proper start; total procedure takes approx. 15 minutes
- Use openstack-status to verify working

- "Network Manager is not compatible with current setup of Openstack" - doesnt explain what component. I checked and remembered teaming is not compatible with NetworkManager
- After disabling the NetworkManage you might want to check your interfaces with "ip link show"
- Check /etc/default/grub to ensure grub2-mkconfig also wrote network items here (when we ran it post-install)
- Snapshot mentioned is good if we want to roll back to pre-Openstack-installstate

These are some of the useful options to set when you deploy Openstack:

Parameters to be modified in the answers.txt file:

- CONFIG_DEFAULT_PASSWORD=password
- CONFIG_SWIFT_INSTALL=n
- CONFIG_HEAT_INSTALL=y
- CONFIG_NTP_SERVERS=pool.ntp.org
- CONFIG_KEYSTONE_ADMIN_PW=password
- CONFIG_CINDER_VOLUMES_CREATE=y
- CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS=physnet1:br-eth1
- CONFIG_NEUTRON_OVS_BRIDGE_IFACES=br-eth1:eth1
- CONFIG_NEUTRON_OVS_TUNNEL_IF=eth1
- CONFIG_HORIZON_SSL=y
- CONFIG_HEAT_CFN_INSTALL=y
- CONFIG_PROVISION_DEMO=n
- CONFIG_NEUTRON_OVS_TUNNEL_IF=eth1

There is also a Nagios option. Not turned on for this
EXCLUDE_SERVERS is usefull if you set up another testbed and don't want it taling to this one

When you are done, and run "packstack --answer-file /root/answers.txt" it will take about 15 or 20 minutes
In demo, said wait until after configuring ssh keys before walking off.

Errors

After config continued, neutron install failed- said it couldn't work with eth1. Installer quit to cmd line. Running "ip a" revealed that eth1 says it is down! Go to /etc/sysconfig/network-scripts/ then ls, then "cat ifcfg-eth1". ONBOOT=no is the problem. Set it to YES, save/exit, then run "ifup eth1". "ip a" again to check.

After fixing this, we reviewed the progress of the install, seeing no actuall configuration done yet, and merely "adding manifests for" lines (no "applying" Puppet's *.pp files/ lines). Determining that it was safe, run "packstack --answer-file /root/answers.txt" again. If it had gone much futher, we would have needed to roll back to the previous snapshot and proceed from there.

End of install:
- A line tells us that /root/keystonerc_admin was created - we will need that (login credentials)
- The url to Horizon dashboard
- where install logs and manifest were put
- finally, that we will need to reboot

First, run openstack-status

Quick note on current network:
Server 1 is the internet-connected node- the gateway for the cloud.
Internal SDN will use 172.16.0.0/24 between instances.
Server1 has Eth1 and Server2 has Eth0, but we won't be looking at those
For convenience, this allows the physical machines to have direct access to the internet for their repository stuff, but should be seen as unusual in the cloud scenario.
The OVS and BR connections are shown in "ip a" but are not configured yet. Eth0 is associated with OVS with no IP4, a EUI64 IPv6, and is considered not connected externally.

Adding a Packstack Compute node
Before beginning added CONFIG_COMPUTE_HOSTS 192.168.4.11 (dot10 was there, now there are two)
Note that this says "servers on which to install the Compute service"
You need to add dot10 to EXCLUDE_SERVERS so it doesn't try to reinstall

Creating br-ex, the external bridge
/etc/sysconfig/network-scripts/
cp ifcfg-eth0 ifcfg-br-ex (duplicate a copy for br-ex)
Open **ifcfg-eth0**
Zap most of this, EXCEPT:
BOOTPROTO="none", ONBOOT="yes"
Add DEVICE=eth0, add TYPE="OVSPort", OVS_BRIDGE=br-ex, DEVICETYPE=ovs
You should only have these 6 lines. Save.
For **ifcfg-br-ex:**
TYPE="OVSBridge", DEVICE=br-ex, BOOTPROTO="static", DEVICETYPE=ovs
(notes that there is no difference between BOOTPROTO static and none, but static is more descriptive)
Delete DEFROUTE and IPV4/6 stuff, NAME, UUID, HWADDR
Keep IPADDR0,PREFIX0, DNS1 as-is (same as eth0 was using)
GATEWAY must be called GATEWAY in this version of Openstack (using GATEWAY0 triggers a bug)
add PEERDNS=yes and USRCTL=yes
Save and reboot

Do "ip a" and "ip route" to make sure it got all the stuff it needs, plus check /etc/resolve.conf

Machines in RHCE exam are often set up with DHCP, server will loose it's hostname and some functionality, so you may need to fix it after reboot with hostnamectl set-hostname server1.example.com

Deploying Openstack Behind a Proxy
Make /etc/environment
http_proxy=http://user@someproxy.com:80
https_proxy=https://user@someproxy.com:443
no_proxy= localhost, 127.0..0.1,youdomain.com, your.ip.add.ress
To ensure that your nodes aren't trying to talk to eachother over the proxy there is the no_proxy directive.
Optionally, also include these in the /etc/yum.conf. Yum doesn't use the generic variables just defined.
proxy=
proxy_username=
proxy_password=

Glance Image + Flavor (4GB RAM 4 CPU 4GB storage) + Volume = instance
instance => virtual router => internal network
=> Floating IP => external network
floating IP + security group + ssh key

So if you are working with a internet browser and try to open a session to the same server such as https://server1/dashboard, you may likely get a message that server certificate failed - go to browser settings and clear certificates and servers marked "Openstack". Quit the browser and open to try again.
Forgot user/pass? Go to ~/keystonerc_admin

Creating Projects and Users in Horizon
Opening and logging into Horizon as an admin, we are greeted with the Overview page with a usage summary table with date range, and list of projects including their: VCPUs, disk, RAM, VCPU hours, Disk GB hours, and Memory MB hours.
A sidebar provides us with:
Project tab>Network, Orch, and Compute sections
Admin tab>
Identity tab>Projects and Users sections

A non-admin user still has the sidebar but has a simpler summary view (showing just the main project) and limited options. The overview is under the Computer header in the sidebar where it sits next to instances, volumes, images, and "access & security".

Default Projects (tenants) are "admin" and "services"
The main listing shows name, description, projectID, enabled (or not) and an Actions column providing a "manage members" pull-down menu. Checkboxes on the side let you select multiple projects and create and delete buttons are on the top. Predictably, you add a name and description; users tab allows user add/remove. A quotas tab has many options to set project confines:

Quota Name (default value)	Defines the number of	Service
Metadata Items (128)	Metadata items allowed for each instance.	Compute
VCPUs (20)	Instance cores allowed for each project.	Compute
Instances (10)	Instances allowed for each project.	Compute
Injected Files (5)	Injected files allowed for each project.	Compute
Injected File Content Bytes (10240)	Content bytes allowed for each injected file.	Compute
Volumes (10)	Volumes allowed for each project.	Block Storage
Volume Snapshots (10)	Volume snapshots allowed for each project.	Block Storage
Total Size of Vols (GB) (1000)	Volume gigabytes allowed for each project.	Block Storage
RAM (MB) (51200)	RAM megabytes allowed for each instance.	Compute
Security Groups (10)	Security groups allowed for each project.	Compute
Security Group Rules (100)	Rules allowed for each security group.	Compute
Floating IPs (50)		
Networks (10)		
Ports (50)		
Routers (10)		
Subnets (10)		

Start with Identity: before we can do anything we need to create users.
Service users exist for Glance, Neutron, etc., plus Admin
New users have username, password, email (optional), primary project and role
It makes sense in larger projects to give every tenant it's own administrator, and Red Hat wants that.
Roles (shown) are admin, _member_, heat_stack_owner, heat_stack_user

Images

For this, downloaded Cirros. Google for cirros image download brings us right to Openstack's Get Image page.
Don't download the image- just copy the link
Example logs in as a regular user
The images window lists image name, type, status, public, protected, format, size, and "actions"
There are choices to view by project, shared, or public, and a "create" button.

The create window has few options: name, description, image source (file/location), location, format, architecture. (we paste the url here)
Default image format is QCOW2 for QEMU. There is also Amazon Kernel Image (AKI), Amazon Machine Image (AMI), Amazon Ramdisk Image (ARI), Open Virtual Appliance (OVA), Docker, VDI, VHD, VMDI, ISO, Raw
There are options for Minimum disk (GB), Maximum disk (MB), and checkboxes for public and protected. None of these or architecture is required to specify.
Copy data (to image service) should be checked or else it can end up downloading each use.

These options are the same for admin and regular cloud users, but cloud users can't make HW profiles (flavors) so we have to log out and log back in as admin.
Consider the "cloud user" account for the "tenant administrator" and admin account for the "cloud administrator" with wider authorization and access.

Flavors
Here, we get default options listed below, columns for the listing: name, VCPUs, RAM, Root Disk, Ephemeral disk, Swap disk, ID, Public (yes/no), metadata (yes/no), and edit. Here are those and the default flavor offerings:

Flavor Name	VCPUs	RAM	Root Disk	Ephemeral Disk	Swap Disk	ID	Public	Metadata	Actions
m1.tiny	1	512MB	1GB	0GB	0MB	1	Yes	No	Edit Flavor
m1.small	1	2GB	20GB	0GB	0MB	2	Yes	No	Edit Flavor
m1.medium	2	4GB	40GB	0GB	0MB	3	Yes	No	Edit Flavor
m1.large	4	8GB	80GB	0GB	0MB	4	Yes	No	Edit Flavor
m1.xlarge	8	16GB	160GB	0GB	0MB	5	Yes	No	Edit Flavor

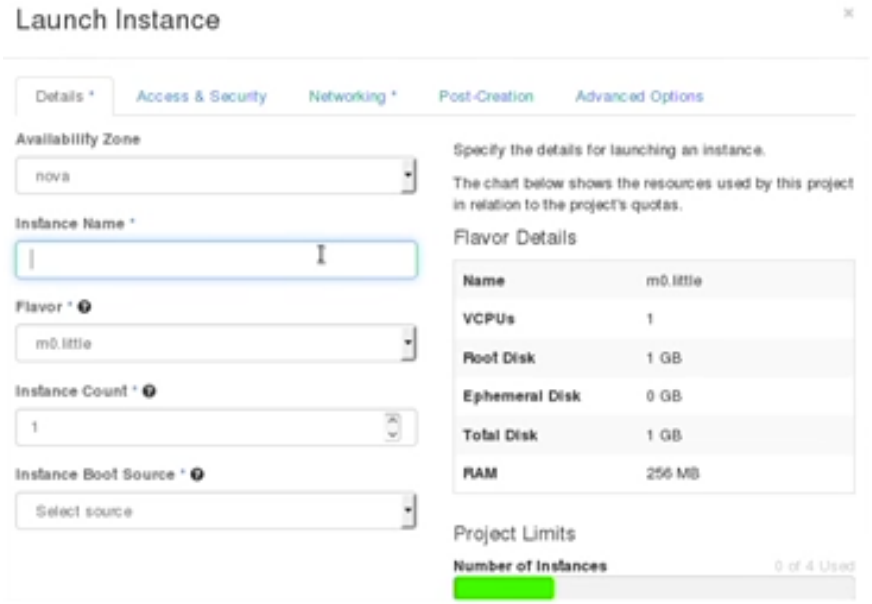
Displaying 5 items

These are pretty big. As an example we made one called "little" that is much like "tiny" (above)

Network
We need to define the network before spinning up instances. This can be done as a cloud user.
Create network as an internal network: give a general name and note "admin state" of up (ignore it). For subnet, we are asked to name the subnet, give IPv4 or v6, network address, gateway. You are just informing of subnets here, so they don't need to relate to virtual network card addresses. Default for GW will be first available IP in subnet, but there can be issues if you leave this blank (this example puts in 10.0.0.254 - also see "router" below). The next subnet details tab asks for dhcp pools (or disable dhcp- pools only need "to address, from address"), name servers (needed with or without DHCP on), host routes (none required)

Also create a network for an external network. This is set for a real interface, GW for internet
Disable DHCP (checkbox) but **add pool anyway for floating IP addresses** to add to nodes later on.
When we finish, we go to network topology in the sidebar (cloud user gets networks, routers, and topology links).
Since the external network is configured to directly-connected physical cards, we have to log out and log in as cloud administrator to access the rest.
When you go to "edit network" in this mode, it allows checking a (previously absent) box that says "external network". All that was needed was checking that box. No need for messing with "Ports" section or anything, and we can re-login as regular cloud admin, and click on "Routers" to add one. We can now give one an external network since we made it available. When we look at the settings (after adding) we can see SNAT applied for us. An interfaces tab lets us add them to either the internal or external network.
Important: the example set the GW for the network to 10.0.0.254 - this should be the IP address for the external network interface!
The network topology should show a router between the 10.0.0.0/24 internal and 192.168.4/24 external network.

Instances
The instance list shows name, image, ip addy, size, key pair, status, zone, task, power state, when created and "actions". We click "New".
Choices: Availability Zone (default is nova), flavor, instance count (1), boot source (image), image name (select)
For key pair, you will need to run ssh commands into terminal (it gives you instructions)
ssh-keygen -t rsa -f cloud.key <----will spit out cloud.key and cloud.key.pub for you. cat and copy contents of cloud.key to past in Horizon. You probably want to clean up by putting keys in ~/.ssh directory: "cp cloud.* .ssh
Set security group to default (checkbox) for now. Under networking, select the internal network we created.
There are also "post-creation" and "advanced" options, we skipped for now. Hit "Launch"
We get error (to troubleshoot later) "no valid host was found... not enough hosts available"



Resuming the Procedure

- Make sure a tenant, tenant user and role exist
- Provide an Image
- Create Flavors
- Provide an external network
- Create an SSH key
- Select the network
- Launch the image
- Allocate a floating IP
- Add block storage

Troubleshooting
openstack-status is the way to authoritatively determine the availability of services

```
[root@server1 ~]# openstack-status
== Nova services ==                    == Cinder services ==
openstack-nova-api:           active   openstack-cinder-api:          active
openstack-nova-cert:          active   openstack-cinder-scheduler:    active
openstack-nova-compute:       active   openstack-cinder-volume:       active
openstack-nova-network:       inactive openstack-cinder-backup:       inactive
openstack-nova-scheduler:     active   == Heat services ==
openstack-nova-conductor:     active   openstack-heat-api:            active
== Glance services ==                  openstack-heat-api-cfn:        inactive
openstack-glance-api:         active   openstack-heat-api-cloudwatch: inactive
openstack-glance-registry:    active   openstack-heat-engine:         active
== Keystone service ==                 == Support services ==
openstack-keystone:           inactive mysqld:                        active
== Horizon service ==                  openvswitch:                   active
openstack-dashboard:          301      dbus:                          active
== neutron services ==                 target:                        active
neutron-server:               active   rabbitmq-server:               active
neutron-dhcp-agent:           active   memcached:                     active
neutron-l3-agent:             active   == Keystone users ==
neutron-metadata-agent:       active   Warning keystonerc not sourced
neutron-openvswitch-agent:    failed   [root@server1 ~]#
```

systemctl status httpd -l (l for long format)

/var/log/horizon
Also /var/log directories for cinder, keystone, glance, heat, mariadb, neutron, nova, openvswitch, rabbitmq
If you are checing things right after restarting, give everything a minute or two on slower machines to initialize
Verifying things, you might also systemctl status rabbitmq-server, mariadb-server, and keystone
Quick notes:
- if RabbitMQ fails management will be impossible but instances won't fail. In service config, contact info must be included. Can be TLS-secured, but if manual install can be skipped. Config is in nova.conf - rabbit_* attributes. RabbitMQ here is fairly simple but you could do a 2-day course on it to cover everything it can do.
- Keystone relies on DB, generally SQL-compatible, and default is MaraDB. Information is supposedly synchronized with the DB, but sometimes it isn't the case, particularly with Neutron stuff (inconsistencies with neutron lists). You may have to dig into it for troubleshooting occasionally.
Storage in Openstack is ephemeral (nothing stored in a permanent way), USUALLY.
- However- while deploying instances, using the settings in a flavor, disk files are written to /var/lib/nova/instances on the Compute node. Once you move around this wont move around with the instance, stays on the same compute node. VM XML file can give details- this is persistent storage, so it may be good to make it available on an NFS (or other shared storage) as a backend. Since generally storage is ephemeral, your should use cinder volumes for truly persistent cloud storage
/var/lib/nova/ contains /buckets, /instances, /keys, /networks, /tmp Inside /instances are directories named for UUIDs These contain console.log, disk, disk.info disk.local disk.swap libvirt.xml - disk being a disk file. all items belong to users and groups of qemu and nova. The xml file is full of the VMs info in KVM libvirt format.
In this way, storage is not necessarily always ephemeral.

Keystone

The backend DB everything is talking to (for authentication and authorization)
These keystone CLI commands are replaced with python-openstackclient.
keystone service-list -services available to users
keystone role-list -roles available to users
keystone user-list
keystone endpoint-list shows the authentication urls to use when digging deeper into Openstack. We need this when working on configuration files manually later on. Whenever configuring services endpoints must be created in Keystone. The endpoint is the url a service uses when connecting to another service

Started on these, but got message "Expecting an auth URL via either --os-auth-url or env[OS_AUTH_URL]
We have to fix this:
~/keystonerc_admin <---Holds the env[ ] variable mentioned with username and pass we set up
Run this file like a script and get a prompt!
source keystonerc_admin
> keystone service-list and it works.
Page two in the beginning of these notes shows the output of most of these commands

keystone --help gives tons of info

```
usage: keystone [--version] [--debug] [--os-username <auth-user-name>]
                [--os-password <auth-password>]
                [--os-tenant-name <auth-tenant-name>]
                [--os-tenant-id <tenant-id>] [--os-auth-url <auth-url>]
                [--os-region-name <region-name>]
                [--os-identity-api-version <identity-api-version>]
                [--os-token <service-token>]
                [--os-endpoint <service-endpoint>] [--os-cache]
                [--force-new-token] [--stale-duration <seconds>] [--insecure]
                [--os-cacert <ca-certificate>] [--os-cert <certificate>]
                [--os-key <key>] [--timeout <seconds>]
                <subcommand> ...

Command-line interface to the OpenStack Identity API.

Positional arguments:
  <subcommand>
    catalog                 List service catalog, possibly filtered by
                            service.
    ec2-credentials-create  Create EC2-compatible credentials for user per
                            tenant.
    ec2-credentials-delete  Delete EC2-compatible credentials.
    ec2-credentials-get     Display EC2-compatible credentials.
    ec2-credentials-list    List EC2-compatible credentials for a user.
    endpoint-create         Create a new endpoint associated with a service.
    endpoint-delete         Delete a service endpoint.
    endpoint-get            Find endpoint filtered by a specific attribute or
                            service type.
    endpoint-list           List configured service endpoints.
    password-update         Update own password.
    role-create             Create new role.
    role-delete             Delete role.
    role-get                Display role details.
    role-list               List all roles.
    service-create          Add service to Service Catalog.
    service-delete          Delete service from Service Catalog.
    service-get             Display service from Service Catalog.
    service-list            List all services in Service Catalog.
    tenant-create           Create new tenant.
    tenant-delete           Delete tenant.
    tenant-get              Display tenant details.
    tenant-list             List all tenants.
    tenant-update           Update tenant name, description, enabled status.
    token-get               Display the current user token.
    user-create             Create new user.
    user-delete             Delete user.
    user-get                Display user details.
    user-list               List users.
    user-password-update    Update user password.
    user-role-add           Add role to user.
    user-role-list          List roles granted to a user.
    user-role-remove        Remove role from user.
    user-update             Update user's name, email, and enabled status.
    discover                Discover Keystone servers, supported API versions
                            and extensions.
    bootstrap               Grants a new role to a new user on a new tenant,
                            after creating each.
    bash-completion         Prints all of the commands and options to stdout.
    help                    Display help about this program or one of its
                            subcommands.

Optional arguments:
  --version                 Shows the client version and exits.
  --debug                   Prints debugging output onto the console, this
                            includes the curl request and response calls.
                            Helpful for debugging and understanding the API
                            calls.
  --os-username <auth-user-name>
                            Name used for authentication with the OpenStack
                            Identity service. Defaults to env[OS_USERNAME].
  --os-password <auth-password>
                            Password used for authentication with the
                            OpenStack Identity service. Defaults to
                            env[OS_PASSWORD].
  --os-tenant-name <auth-tenant-name>
                            Tenant to request authorization on. Defaults to
                            env[OS_TENANT_NAME].
  --os-tenant-id <tenant-id>
                            Tenant to request authorization on. Defaults to
                            env[OS_TENANT_ID].
  --os-auth-url <auth-url>  Specify the Identity endpoint to use for
                            authentication. Defaults to env[OS_AUTH_URL].
  --os-region-name <region-name>
                            Specify the region to use. Defaults to
                            env[OS_REGION_NAME].
  --os-identity-api-version <identity-api-version>
                            Specify Identity API version to use. Defaults to
                            env[OS_IDENTITY_API_VERSION] or 2.0.
  --os-token <service-token>
                            Specify an existing token to use instead of
                            retrieving one via authentication (e.g. with
                            username & password). Defaults to
                            env[OS_SERVICE_TOKEN].
  --os-endpoint <service-endpoint>
                            Specify an endpoint to use instead of retrieving
                            one from the service catalog (via authentication).
                            Defaults to env[OS_SERVICE_ENDPOINT].
  --os-cache                Use the auth token cache. Defaults to
                            env[OS_CACHE].
  --force-new-token         If the keyring is available and in use, token will
                            always be stored and fetched from the keyring
                            until the token has expired. Use this option to
                            request a new token and replace the existing one
                            in the keyring.
  --stale-duration <seconds>
                            Stale duration (in seconds) used to determine
                            whether a token has expired when retrieving it
                            from keyring. This is useful in mitigating process
                            or network delays. Default is 30 seconds.
  --insecure                Explicitly allow client to perform "insecure" TLS
                            (https) requests. The server's certificate will
                            not be verified against any certificate
                            authorities. This option should be used with
                            caution.
  --os-cacert <ca-certificate>
                            Specify a CA bundle file to use in verifying a TLS
                            (https) server certificate. Defaults to
                            env[OS_CACERT].
  --os-cert <certificate>   Defaults to env[OS_CERT].
  --os-key <key>            Defaults to env[OS_KEY].
  --timeout <seconds>       Set request timeout (in seconds).

See "keystone help COMMAND" for help on a specific command.
(END)
```

To simplify remembering:
service-, role-, user-, tenant-, and ec2-credentials-, and endpoint- all prefix create, delete, get, list commands
For user- commands, there is also user-password-update, user-update, and user-role has -add -list -remove variants

openstack-service stautus keystone
cd /usr/lib/systemd/system/; ls open*

```
[root@server1 ~(keystone_admin)]# openstack-service status keystone
openstack-keystone (pid 1450) is active
[root@server1 ~(keystone_admin)]# cd /usr/lib/systemd/system
[root@server1 system(keystone_admin)]# ls open*
openstack-ceilometer-alarm-evaluator.service  openstack-swift-account-reaper@.service
openstack-ceilometer-alarm-notifier.service   openstack-swift-account-replicator.service
openstack-ceilometer-api.service              openstack-swift-account-replicator@.service
openstack-ceilometer-central.service          openstack-swift-account.service
openstack-ceilometer-collector.service        openstack-swift-account@.service
openstack-ceilometer-notification.service     openstack-swift-container-auditor.service
openstack-cinder-api.service                  openstack-swift-container-auditor@.service
openstack-cinder-backup.service               openstack-swift-container-reconciler.service
openstack-cinder-scheduler.service            openstack-swift-container-replicator.service
openstack-cinder-volume.service               openstack-swift-container-replicator@.service
openstack-glance-api.service                  openstack-swift-container.service
openstack-glance-registry.service             openstack-swift-container@.service
openstack-glance-scrubber.service             openstack-swift-container-updater.service
openstack-keystone.service                    openstack-swift-container-updater@.service
openstack-losetup.service                     openstack-swift-object-auditor.service
openstack-nova-api.service                    openstack-swift-object-auditor@.service
openstack-nova-cert.service                   openstack-swift-object-expirer.service
openstack-nova-conductor.service              openstack-swift-object-replicator.service
openstack-nova-consoleauth.service            openstack-swift-object-replicator@.service
openstack-nova-console.service                openstack-swift-object.service
openstack-nova-metadata-api.service           openstack-swift-object@.service
openstack-nova-novncproxy.service             openstack-swift-object-updater.service
openstack-nova-scheduler.service              openstack-swift-object-updater@.service
openstack-nova-xvpvncproxy.service            openstack-swift-proxy.service
openstack-swift-account-auditor.service       openvswitch-nonetwork.service
openstack-swift-account-auditor@.service      openvswitch.service
openstack-swift-account-reaper.service
```

systemctl status openstack-keystone (of course) gives a lot more than the previous non-systemd command
less /etc/keystone/keystone.conf
You can also do grep -v '^#' /etc/keystone/keystone.conf
- to see all of the lines that aren't commented out

mysql -e 'use keystone; show tables;'

```
+------------------------+
| Tables_in_keystone     |
+------------------------+
| access_token           |
| assignment             |
| consumer               |
| credential             |
| domain                 |
| endpoint               |
| endpoint_group         |
| federation_protocol    |
| group                  |
| id_mapping             |
| identity_provider      |
| idp_remote_ids         |
| mapping                |
| migrate_version        |
| policy                 |
| policy_association     |
| project                |
| project_endpoint       |
| project_endpoint_group |
| region                 |
| request_token          |
| revocation_event       |
| role                   |
| sensitive_config       |
| service                |
| service_provider       |
| token                  |
| trust                  |
| trust_role             |
| user                   |
| user_group_membership  |
| whitelisted_config     |
+------------------------+
```

CLI User Administration

The openstack CLI command uses many of the similar subcommands as the keystone command, without the "-". It is a newer command so you won't find it in older versions of Openstack.
One other difference is it uses "project" instead of "tenant"
Here are it's user subcommands:

```
[root@server1 system(keystone_admin)]# openstack --help | grep user
                [--os-auth-type <auth-type>] [--os-username <auth-username>]
                [--os-user-domain-id <auth-user-domain-id>]
                [--os-user-domain-name <auth-user-domain-name>]
                [--os-user-id <auth-user-id>] [--os-password <auth-password>]
                      selected based on --os-username/--os-token (Env:
  --os-username <auth-username>
  --os-user-domain-id <auth-user-domain-id>
  --os-user-domain-name <auth-user-domain-name>
  --os-user-id <auth-user-id>
  role add       Add role to project:user
  role remove    Remove role from project : user
  server lock    Lock a server. A non-admin user will not be able to execute actions
  user create    Create new user
  user delete    Delete user(s)
  user list      List users
  user role list  List user-role assignments
  user set       Set user properties
  user show      Display user details
[root@server1 system(keystone_admin)]# openstack user list
+----------------------------------+------------+
```

That having been said, back to the keystone commands:
keystone user-create --name list --password password
- creates the user, spits out theusername and id number
keystone user-role-add --user-id lisa role-id admin --tenant-id project
keystone user-create --name lisa --pass password
keystone user-list
keystone user-delete *USERID*
keystone tenant-create --name *tenant-name*
keystone tenant-list
keystone role-create --name somerole
keystone user-role-add --user-id *user* --role-id *role* --tenant-id *tenant*

Recognizing Keystone problems

- All requests arriving to services, have to go through keystone
- Requests contain an authorization URL (endpoint)
- The SERVICE_TOKEN and SERVICE_ENDPOINT from keystonerc_admin are used for this

If authentication doesn't work (look for "invalid openstack credentials"), export these

- export SERVICE_TOKEN=$(crudini --get /etc/keystone/keystone.conf DEFAULT admin_token)
- export SERVICE_ENDPOINT=http://server1.example.com:35357/v2.0

When you are looking at the logs in /var/log/keystone, you might want to run a grep exclude on INFO-level messages (there are a lot) by using "grep -v INFO /var/log/keystone/keystone.log"
Make use of keystone endpoint-list
Check keystonerc_admin or other relevant keystonerc file (keystonerc_user
Did you source it the way you should have for credentialing in?
Differences between user and admin versions of keystonerc files

```
[root@server1 ~(keystone_admin)]# cat keystonerc_admin
unset OS_SERVICE_TOKEN
export OS_USERNAME=admin
export OS_PASSWORD=password
export OS_AUTH_URL=http://192.168.4.10:5000/v2.0
export PS1='[\u@\h \W(keystone_admin)]\$ '

export OS_TENANT_NAME=admin
export OS_REGION_NAME=RegionOne
[root@server1 ~(keystone_admin)]# cat keystonerc_user
unset SERVICE_TOKEN SERVICE_ENDPOINT
export OS_USERNAME=user
export OS_TENANT_NAME=workshop
export OS_PASSWORD=password
export OS_AUTH_URL=http://server1.example.com:35357/v2.0/
export PS1='[\u@\h \W(keystone_user)]\$ '
[root@server1 ~(keystone_admin)]#
```

Swift Object storage:
We want to work with objects- not files.
Disk-based is not elastic, and a DFS isn't elastic enough.
Swift takes care of replication, availability in a model where data is written to multiple binary objects, dispersed in a replicated way over many nodes. Metadata is similarly dispersed and replicated.
The application is required to make (RESTful) API calls to read and write, through a Swift proxy.
The entire structure is made of many nodes, many binary objects.

[Security note to self is that Swift data must be deabstracted through a Swift proxy of some sort to be exploited or broken into. Ideally, the existing framework would be used to accomplish this rather than obtaining files and reconstructing binaries, metadata from the entire mess of a recon grab.]

So Swift object storage also provides a flexible backend framework to support cinder volumes and glance images.

Manually Installing Swift - Knowledge for Troubleshooting not in RHCE
Here is the procedure (outline):

- yum install –y openstack-swift-proxy openstack-swift-object openstack-swift-container openstack-swift-account python-swiftclient memcached
- source /root/keystonerc_admin
- keystone user-create --name swift –pass password --tenant services
- keystone role-create --name SwiftOperator
- keystone user-role-add --role admin --tenant services --user swift
- Look for object storage service: keystone service-list | grep swift
- If necessary: keystone service-create --name swift --type object-store --description "OpenStack Object Storage"
- Create the endpoints: keystone endpoint-create --service swift --region RegionOne --publicurl "http://192.168.4.10:8080/v1/AUTH_%(tenant_id)s" --adminurl "http://192.168.4.10:8080/v1/AUTH_%(tenant_id)s" --internalurl "http://192.168.4.10:8080/v1/AUTH_%(tenant_id)s"

The install: (we omitted this from the Packstack batch install)
openstack-swift -proxy, -object, -container, -account, THEN python-swiftclient and memcached

Creating the user, example really does use name swift, and tenant services- these aren't there for special reasons

Operator role allows assigning operator permissions to storage administrators in swift

Region- just like in AWS. Can make things region specific later.

Endpoints: notice the 3 urls are the same- public, admin, or tenant will populate the "tenantID appropriately when actually used

Deploying Swift Storage
Since this is just a test environment to see the procedure, we are only going to work with a single node here

- Add 2 1 GB disks to server1, create a single partition on them and format them with the XFS file system. The names should be /dev/sdb1 and /dev/sdc1.
- mkdir –p /srv/node/z{1,2}d1
- Add following to /etc/fstab
  - /dev/sdb1 /srv/node/z1d1 xfs defaults 0 0
  - /dev/sdc1 /srv/node/z2d1 xfs defaults 0 0
  - mount –a
- chown –R swift:swift /srv/node
- restorecon –rv /srv
- crudini --set /etc/swift/swift.conf swift-hash swift_hash_path_prefix $(openssl rand –hex 10)
- crudini --set /etc/swift/swift.conf swift-hash swift_hash_path_suffix $(openssl rand –hex 10)
- crudini --set /etc/swift/account-server.conf DEFAULT bind_ip 192.168.4.10
- crudini --set /etc/swift/container-server.conf DEFAULT bind_ip 192.168.4.10
- crudini --set /etc/swift/object-server.conf DEFAULT bind_ip 192.168.4.10

crudini is utility for modifying config files (is scriptable
Account, container, and server lines are for those "rings"

cat /proc/partitions
fdisk /dev/sdb and sdc - they are already both 1GB each, so no partitioning needed.
mkfs.xfs them both as well

```
mkdir -p /srv/node/z{1,2}d1
ls reveals z1d1, z2d1 - z for zone
vim /etc/fstab
/dev/sdb1    /srv/node/z1d1    xfs   defaults    0   0
/dev/sdc1    /srv/node/z2d1    xfs   defaults    0   0
mount -a
```

Configuring Swift Rings

Rings determine where data is stored in the cluster
Three rings are required:

- object
- container
- account services

Storage devices are divided in partitions

- each partition is a directory on the XFS file system

While creating ring files, three parameters are used:

- Partition power (2 ^partition power = # of partitions)
- Replica count
- Min_part_hours

Object ring are the binary objects themselves; container ring is about how these are stored, accounts ring is auth and access

Min_part_hours is the minimum time before partition synch happens- indirectly ensures that only one partition's replica is moved at a time. Moving multiple copies at the same time risks making data inaccessible. 24 is a recommended default, in a production environment. Checking logs can reveal time needed (to optimize setting).

Partition power, 4096 in this case (2^12). This is the amount of partitions needed to provision for all of Swift's operations: "maximum number of drives you expect your cluster to have at it's largest size. In general, it is considered good to have about 100 partitions per drive in large clusters." is how OReilly's Openstack Swift: Administering..." puts it, and the best definition I found. Calculate with [log2 (100 x max disks)] - e.g. is large deployment of 1800 drives that will grow to a total of 18,000 drives - part power = [log2 (100 x 18000)] = 20.77 = 21 allows for 100 partitions per disk.

source /root/keystonerc_admin
Create the rings:

- swift-ring-builder /etc/swift/account.builder create 12 2 1
- swift-ring-builder /etc/swift/container.builder create 12 2 1
- swift-ring-builder /etc/swift/object.builder create 12 2 1

Add devices to the rings

- for i in 1 2; do swift-ring-builder /etc/swift/account.builder add z${i}-192.168.4.10:6202/z${i}d1 100; done
- for i in 1 2; do swift-ring-builder /etc/swift/container.builder add z${i}-192.168.4.10:6201/z${i}d1 100; done
- for i in 1 2; do swift-ring-builder /etc/swift/object.builder add z${i}-192.168.4.10:6200/z${i}d1 100; done
- NOTE: Depending your SELinux configuration, ports used can be different. Verify ports in case it doesn't work!

When running for "i in..." you may get a message saying region not specified, using region 1. This is normal.
In the for statement, "i" refers to our partition names we used (1 and 2).
The meaningful output for each of these is similar to this:
Device d0r1z1-192.168.4.10:6202R192.168.4.10:6202/z1d1_"" with 100.0 weight got id0
Device d1r1z2-192.168.4.10:6202R192.168.4.10:6202/z2d1_"" with 100.0 weight got id1

After adding devices, distribute the partitions:

- swift-ring-builder /etc/swift/account.builder rebalance
- swift-ring-builder /etc/swift/container.builder rebalance
- swift-ring-builder /etc/swift/object.builder rebalance

Verify the rings have been built successfully

- ls /etc/swift/*gz
- Check the configuration in the ring files, using an editor

Start and enable services:

- systemctl enable openstack-swift-account
- systemctl enable openstack-swift-container
- systemctl enable openstack-swift-object
- openstack-service start swift

Set permissions and verify working:

- chown –R root:swift /etc/swift
- tail /var/log/messages

Rebalance distributes the binary objects over the different partitions that are available in the configuration
The output of the rebalance commands looks like this:
Reassigned 4096 (100.00%) partitions. Balance is now 0.00. Dispersion is now 0.00.
As noted, the configuration files in /etc/swift are indeed gz'd

```
[root@server1 ~(keystone_admin)]# cd /etc/swift/
[root@server1 swift(keystone_admin)]# ls
account.builder  account-server.conf  container-reconciler.conf  container-server.conf  object.ring.gz      proxy-server
account.ring.gz  backups              container.ring.gz          object.builder         object-server       proxy-server.conf
account-server   container.builder    container-server           object-expirer.conf    object-server.conf  swift.conf
```

Later, when verifying, you need to use netstat -ntulpe

Deploying a Swift Proxy
Proxy determines to which nodes gets and puts are directed (remember JSON/RESTful?)
To avoid SPOF (single point of failure) use multiple in production env
Employ a load balancer of round-robin DNS to distribute data among proxies
From server1, configure the proxy
Update the proxy server configuration file

- crudini --set /etc/swift/proxy-server.conf filter:authtoken admin_tenant_name services
- crudini --set /etc/swift/proxy-server.conf filter:authtoken auth_host_name 192.168.4.10
- crudini --set /etc/swift/proxy-server.conf filter:authtoken auth_uri http://192.168.4.10:5000
- crudini --set /etc/swift/proxy-server.conf filter:authtoken admin_user swift
- crudini --set /etc/swift/proxy-server.conf filter:authtoken admin_password password

Start and enable all required services

- systemctl start memcached
- systemctl start openstack-swift-proxy
- systemctl enable memcached
- systemctl enable openstack-swift-proxy

Auth host is where keystone lives.

Managing Objects in the Swift Obj Store
Files in the store are binary objects- not accessible as files. The Swift API and swift command allow access, and the CLI command is much like an FTP command
First we have to make these checks, that everything is in order.

- Check port bindings: **cd /etc/swift; grep bind_port *.conf**
- Verify existence of swift directories: **ls –al /srv/node/**
- Use swift-ring-builder to inspect the services rings:
  - **swift-ring-builder /etc/swift/account.builder**
  - **swift-ring-builder /etc/swift/container.builder**
  - **swift-ring-builder /etc/swift/object.builder**
- Apply some changes to /etc/swift/proxy-server.conf and make sure it includes the following settings:
  - [DEFAULT]
    bind_port = 8080
  - [pipeline:main]
    pipeline = healthcheck cache authtoken keystone proxy-logging proxy-server
  - [app:proxy-server]
    allow_account_management = true
    account_autocreate = true
  - [filter:keystone]
    operator_roles = admin, SwiftOperator
  - [filter:authtoken]
    paste.filter_factory = keystonemiddleware.auth_token:filter_factory
    admin_tenant_name = services
    admin_user = swift
    admin_password = password
    auth_host = 192.168.4.10
    auth_port = 35357
    auth_protocol = http
    signing_dir = /tmp/keystone-signing-swift
    auth_uri = http://192.168.4.10:5000
- Restart swift using **openstack-service restart swift**

ls -al /srv/node/ shows z1d1 and z2d1, which normally would be on different servers

```
[root@server1 swift(keystone_admin)]# swift-ring-builder /etc/swift/account.builder
/etc/swift/account.builder, build version 2
4096 partitions, 2.000000 replicas, 1 regions, 2 zones, 2 devices, 0.00 balance, 0.00 dispersion
The minimum number of hours before a partition can be reassigned is 1
The overload factor is 0.00% (0.000000)
Devices:    id  region  zone      ip address  port  replication ip  replication port      name weight partitions balance meta
             0       1     1    192.168.4.10  6202    192.168.4.10              6202      z1d1 100.00       4096    0.00
             1       1     2    192.168.4.10  6202    192.168.4.10              6202      z2d1 100.00       4096    0.00
```

Verifying correct operation of Swift:

- source /root/keystonerc_admin
- keystone user-get swift
- keystone role-list
- keystone service-get swift
- keystone catalog --service object-store

Verifying through use example:

- swift list
- swift post c1
- swift post c2
- echo hi > my.file
- swift upload c1 my.file
- echo hello >my2.file
- swift upload c1 my2.file
- head –c 1024 /dev/urandom > my3.file
- swift upload c2 my3.file
- swift list
- swift list c1
- swift list c2
- Look in the /srv/node mount points to see how the files are stored as binary objects

swift list should just come back like it did nothing. That's because there is nothing to display yet
swift post c1 and c2 are creating two directories (2 partitions) in the Swift object store
Here is a demo of the output when traversing directories
It shows how the binary data is distributed around

```
[root@server1 swift(keystone_admin)]# swift list c1
my.file
my2.file
[root@server1 swift(keystone_admin)]# swift list c2
my3.file
[root@server1 swift(keystone_admin)]# cd /srv/node/
[root@server1 node(keystone_admin)]# ls
z1d1  z2d1
[root@server1 node(keystone_admin)]# cd z1d1/
[root@server1 z1d1(keystone_admin)]# ls
accounts  containers  objects  tmp
[root@server1 z1d1(keystone_admin)]# cd objects/
[root@server1 objects(keystone_admin)]# ls
1031  2090  2768
[root@server1 objects(keystone_admin)]# cd 1031/
[root@server1 1031(keystone_admin)]# ls
2cf
[root@server1 1031(keystone_admin)]# cd 2cf/
[root@server1 2cf(keystone_admin)]# ls
407cef7d876d9cc1793ac59c5940c2cf
[root@server1 2cf(keystone_admin)]# cd 407cef7d876d9cc1793ac59c5940c2cf/
[root@server1 407cef7d876d9cc1793ac59c5940c2cf(keystone_admin)]# ls
1447362868.10401.data
[root@server1 407cef7d876d9cc1793ac59c5940c2cf(keystone_admin)]# cat 1447362868.10401.data
hi
```

Beyond Swift: Ceph
Improves on Swift - more scalable (more nodes too) and more than just API access
Ceph file system/block device lets you work with like a physical HD (same addressing), with a logical object store backend

Verifying Glance Image Services

- Glance has been installed through Packstack. This procedure verifies that it is operational.
- Check what Packstack has configured:
  - **grep GLANCE /root/answers.txt**
- Verify that Glance processes are operational
  - **ps axf | grep glance**
- Check the bind_port setting:
  - **grep bind_port /etc/glance/glance-api.conf**
  - **ss -nlp | egrep 9292**
- In glance-api.conf, check the following:
  - The **Notification System Options** contain connectivity information for RabbitMQ
  - The **Database Options** show how to connect to mariadb
  - And **keystone_authtoken** options show how to connect to keystone
  - Also check that **flavor** is set to keystone
  - Read the **filesystem_store_datadir** setting, that specifies where image files are stored
- Glance registry is used to store metadata about images. glance-registry.conf contains the relevant settings
  - Check the **[database]** section that specifies how to connect to mariadb
  - Check the **[keystone_authtoken]** section that specifies how to connect to keystone
- Check the keystone configuration:
  - **source /root/keystonerc_admin**
  - **keystone service-get glance**
  - **keystone catalog --service image**
  - **keystone user-get glance**
- Use the **glance** command to list the contents of the image store for all tenants:
  - **glance image-list --all-tenants**

Uploading System Images

- A system image is an image that has been prepared and downloaded from some Internet location
- Cirros is a recommended image as it is tiny
- source the admin credentials before starting
- Type **glance image-create --name small --is-public True --disk-format qcow2 --container-format bare --copy-from https://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img**
- Verify using **glance image-list**
- Type **glance image-show small** to get more information about the image

glance-image --help
Juno or Kilo this works, Liberty it stops working. We need Openstack command
It wont tell you about deprecation! Just says "glance: error: unrecognized arguments:"
It appears the openstack command, as said in the manpage, is to "provide a common CLI to Openstack APIs... generally equiv to CLIs provided by Openstack project libraries"
man openstack - look for "create new image"
openstack image create --disk format=qcow2 --container-format=bare --public --copy-from https..... small
- it complains that copy-from (which was obtained from the example it gave us!) is "an Image v1 option no longer supported in Image v2"
Go to Openstack website for Liberty version- they tell us to download the image and then load- USING THE OLD COMMAND THEY TOLD US NOT TO USE!
wget http.....image.img --no-check-certificate
glance image-create --file cirros-0.3.4....disk.img
FINALLY FULL CIRCLE BACK TO GLANCE IMAGE-CREATE, EDITED:
glance image-create --name small --visibility public --disk-format qcow2 --container-format bare --file cirros-0.3....img
And it works.

Building Custom Images for Glance (not in RH exam)
virt-sysprep to remove persistent mac addresses user accounts and ssh keys from image
qemu-img info /imagefile ----if you need more info on the image type
glance image-create --file
--location to copy from remote --copy-from is an alternative
--is-public
NOTE look at last example for what WORKED - these Glance image-create options have problems we encountered, but you use the same line
glance image-list (you might also try with option --all-tenants

Swift as a Glance backend - not needed on exam
Local file location doesn't scaale

swift_store_create_contaner_on_put True --- may need to be manually inserted into config

- **source /root/keystonerc_admin**
- **glance image-list**
- Delete all images, using **glance image-delete**
- Edit settings in /etc/glance/glance-api.conf
  - **crudini --set /etc/glance/glance-api.conf DEFAULT default_store swift**
  - **crudini --set /etc/glance/glance-api.conf glance_store stores glance.store.swift.Store**
  - **crudini --set /etc/glance/glance-api.conf glance_store swift_store_auth_address http://192.168.4.10:5000/v2.0/**
  - **crudini --set /etc/glance/glance-api.conf glance_store swift_store_user services:swift**
  - **crudini --set /etc/glance/glance-api.conf glance_store swift_store_key password**
  - **crudini --set /etc/glance/glance-api.conf glance_store swift_store_create_container_on_put True**
- Packstack does not provide glance-registry-paste.ini, nor glance-api-paste.ini, so they need to be copied over:
  - **cp /usr/share/glance/glance-api-dist-paste.ini /etc/glance/glance-api-paste.ini**
  - **cp /usr/share/glance/glance-registry-dist-paste.ini /etc/glance/glance-registry-paste.ini**
- Restart Glance: **openstack-service restart glance**
- Use **wget** to download the cirros image:
  - **wget https://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img**
- Upload the image and verify
  - **glance image-create --name small --disk-format qcow2 --container-format bare --file cirros-0.3.4-x86_64-disk.img**
  - Use **glance image-list** to verify

Troubleshooting Glance

Ensure glance-api and glance-registry processes are running
Check disk space, auth credentials, that correct ports are available
/var/log/glance/ for registry and api ..
You will probably have to use grep verboten on INFO and/or to make useful

Openstack Storage Needs

Cinder review:
By default, an instance spins of with ephemeral storage
- That is: storage that is non-persistent
- It cannot be persistent, because instances move around

Different approaches exist to make storage persistent
Cinder Block Storage is the most common method
Cinder Block Storage provides volumes that can be attached to virtual machines

- On server1, type **grep CINDER /root/answers.txt** to see what has been done from the packstack file
- Use **ps axf | grep cinder** to cerify that cinder services are running
- Verify the configuration of the default cinder port: **grep listen_port /etc/cinder/cinder.conf**
- Type **ss –nlp | egrep 8776** to verify the default cinder port is listening
- Type **ls /etc/cinder/** to list cinder config files
- Type **grep rabbit /etc/cinder/cinder.conf** to verify RabbitMQ connectivity
- To store snapshots in glance, cinder needs to know where to find the glance server:
  - **grep glance /etc/cinder/cinder.conf**
- Open /etc/cinder/cinder.conf and look up the **volume_driver** parameter to check the storage backend
- Look up the **[filter:authtoken]** section in /etc/cinder/api-paste.ini to check authentication settings
- Source the keystone credentials to verify the keystone configuration part
- Type **keystone service-get cinder** to check cinder details
- Get the catalog information, using **keystone catalog --service volume**
- Type **keystone user-get cinder**
- And finally, type **cinder list**. This shouldn't give anything so far.

```
[root@server1 glance(keystone_admin)]# grep CINDER /root/answers.txt
CONFIG_CINDER_INSTALL=y
CONFIG_CINDER_DB_PW=6a8ba0280d504a6f
CONFIG_CINDER_DB_PURGE_ENABLE=True
CONFIG_CINDER_KS_PW=7c6903e0281e43fc
CONFIG_CINDER_BACKEND=lvm
CONFIG_CINDER_VOLUMES_CREATE=y
CONFIG_CINDER_VOLUMES_SIZE=20G
CONFIG_CINDER_GLUSTER_MOUNTS=
CONFIG_CINDER_NFS_MOUNTS=
CONFIG_CINDER_NETAPP_LOGIN=
# the CONFIG_CINDER_NETAPP_LOGIN parameter.
CONFIG_CINDER_NETAPP_PASSWORD=
CONFIG_CINDER_NETAPP_HOSTNAME=
CONFIG_CINDER_NETAPP_SERVER_PORT=80
CONFIG_CINDER_NETAPP_STORAGE_FAMILY=ontap_cluster
CONFIG_CINDER_NETAPP_TRANSPORT_TYPE=http
CONFIG_CINDER_NETAPP_STORAGE_PROTOCOL=nfs
CONFIG_CINDER_NETAPP_SIZE_MULTIPLIER=1.0
CONFIG_CINDER_NETAPP_EXPIRY_THRES_MINUTES=720
CONFIG_CINDER_NETAPP_THRES_AVL_SIZE_PERC_START=20
# CONFIG_CINDER_NETAPP_EXPIRY_THRES_MINUTES parameter. Defaults to 60.
CONFIG_CINDER_NETAPP_THRES_AVL_SIZE_PERC_STOP=60
CONFIG_CINDER_NETAPP_NFS_SHARES=
CONFIG_CINDER_NETAPP_NFS_SHARES_CONFIG=/etc/cinder/shares.conf
CONFIG_CINDER_NETAPP_VOLUME_LIST=
CONFIG_CINDER_NETAPP_VFILER=
CONFIG_CINDER_NETAPP_PARTNER_BACKEND_NAME=
CONFIG_CINDER_NETAPP_VSERVER=
CONFIG_CINDER_NETAPP_CONTROLLER_IPS=
CONFIG_CINDER_NETAPP_SA_PASSWORD=
CONFIG_CINDER_NETAPP_ESERIES_HOST_TYPE=linux_dm_mp
# CONFIG_CINDER_NETAPP_TRANSPORT_TYPE, CONFIG_CINDER_NETAPP_HOSTNAME,
# and CONFIG_CINDER_NETAPP_HOSTNAME options to create the URL used by
CONFIG_CINDER_NETAPP_WEBSERVICE_PATH=/devmgr/v2
CONFIG_CINDER_NETAPP_STORAGE_POOLS=
[root@server1 glance(keystone_admin)]#
```

Most of these have to do with being able to integrate with Netapp SAN, and the important ones for us generally are at the top. LVM=y means you should see volumes with vgs (that it is operational). This jumps to volume_driver line. in our list (it's LVM)

Managing Cinder Volumes

Creation of volumes is typically a cloud user task and not a cloud administrator task so you can log in as such for this
cinder create --display-name my-volume1
cinder list
vgs to also view stuff
lvs to see created Cinder volume
grep backup_driver /etc/cinder/cinder.conf
systemctl status openstack-cinder-backup (start and enable)
- To create backups, you have to add the user as SwiftOperator, which has to be done by admin credentials
source/root/keystonerc_admin
keystone user-role --role SwiftOperator --user myuser --tenant myopenstacktenant
-- log back in as user:
source /root/keystonrc_myuser
cinder backup-create vol1 --display-name vol1-backup
cinder backup-list

```
[root@server1 ~(keystone_admin)]# cinder create help
usage: cinder create [--consisgroup-id <consistencygroup-id>]
                     [--snapshot-id <snapshot-id>]
                     [--source-volid <source-volid>]
                     [--source-replica <source-replica>]
                     [--image-id <image-id>] [--image <image>] [--name <name>]
                     [--description <description>]
                     [--volume-type <volume-type>]
                     [--availability-zone <availability-zone>]
                     [--metadata [<key=value> [<key=value> ...]]]
                     [--hint <key=value>] [--allow-multiattach]
                     [<size>]
```

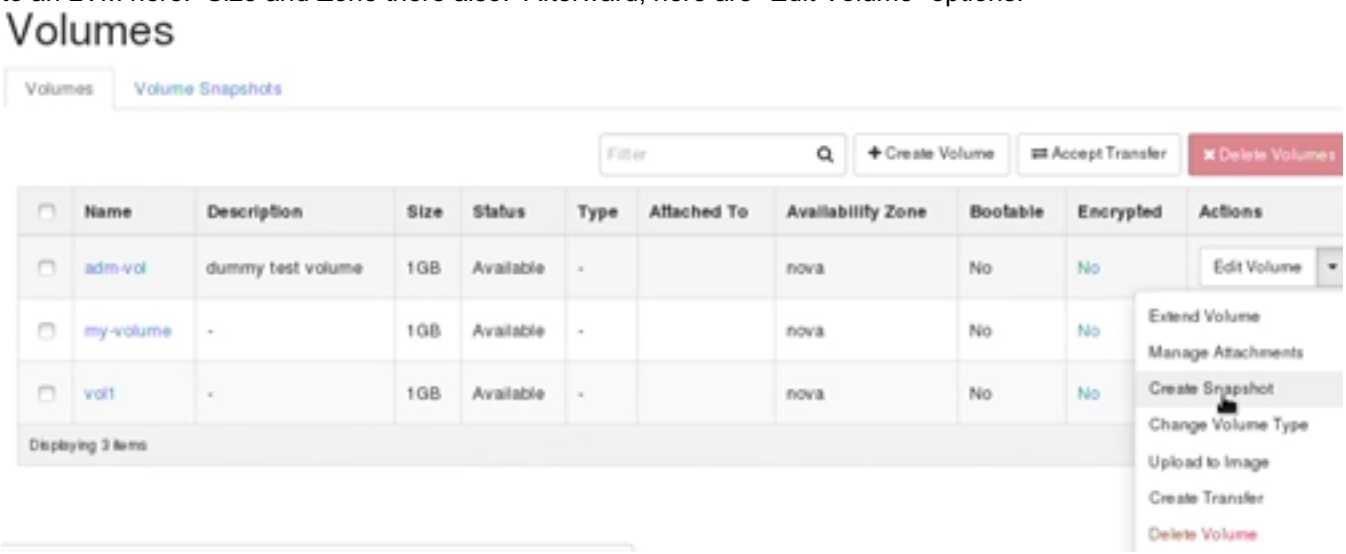
Note the the help test is incomplete, and doesn't list --display-name
On the exam you only need to know how to do this in Horizon

```
[root@server1 ~(keystone_admin)]# cinder create --display-name my-volume 1
+---------------------------------------+--------------------------------------+
|                Property               |                Value                 |
+---------------------------------------+--------------------------------------+
|              attachments              |                  []                  |
|           availability_zone           |                 nova                 |
|                bootable               |                false                 |
|          consistencygroup_id          |                 None                 |
|               created_at              |      2015-11-12T23:57:05.000000      |
|              description              |                 None                 |
|               encrypted               |                False                 |
|                   id                  | 567c142b-d155-4262-a053-9713b05abcbb |
|               metadata                |                  {}                  |
|           migration_status            |                 None                 |
|              multiattach              |                False                 |
|                  name                 |               my-volume              |
|        os-vol-host-attr:host          |                 None                 |
|    os-vol-mig-status-attr:migstat     |                 None                 |
|    os-vol-mig-status-attr:name_id     |                 None                 |
|     os-vol-tenant-attr:tenant_id      |   1e1ab59958d54616a6174bdf74d24384   |
|  os-volume-replication:driver_data    |                 None                 |
| os-volume-replication:extended_status |                 None                 |
|          replication_status           |               disabled               |
|                  size                 |                  1                   |
|              snapshot_id              |                 None                 |
|              source_volid             |                 None                 |
|                 status                |               creating               |
|               user_id                 |   20dbef3a7b1d48738e4f2eb23165879a   |
|              volume_type              |                 None                 |
+---------------------------------------+--------------------------------------+
[root@server1 ~(keystone_admin)]# cinder list
+--------------------------------------+-----------+------------------+-----------+------+-------------+----------+-------------
-+--------------+
|                  ID                  |   Status  | Migration Status |    Name   | Size | Volume Type | Bootable | Multiattach
 | Attached to |
+--------------------------------------+-----------+------------------+-----------+------+-------------+----------+-------------
-+--------------+
| 567c142b-d155-4262-a053-9713b05abcbb | available |        -         | my-volume |  1   |      -      |  false   |    False
 |              |
| e704df8b-d11c-4fc9-806a-a8e4e71ee2e5 | available |        -         |    vol1   |  1   |      -      |  false   |    False
 |              |
+--------------------------------------+-----------+------------------+-----------+------+-------------+----------+-------------
```

```
[root@server1 ~(keystone_admin)]# lvs
  LV                                          VG             Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Con
ert
  root                                        centos         -wi-ao---- 10.31g

  swap                                        centos         -wi-ao----  1.20g

  volume-567c142b-d155-4262-a053-9713b05abcbb cinder-volumes -wi-a-----  1.00g

  volume-e704df8b-d11c-4fc9-806a-a8e4e71ee2e5 cinder-volumes -wi-a-----  1.00g
```

Managing Block Storage from Horizon

In Horizon, in "Volumes" we can see the volumes just created and say in "cinder list"
Hit "create"; call it adm-vol; under Volume Source, we go with default "No source, empty volume". We could attach it to an LVM here. Size and Zone there also. Afterward, here are "Edit Volume" options:



- Asssigning a vol to an instance, use "manage attachments"
- In troubleshooting, tail /var/log/cinder/volume.log and api.log, vgs, and cinder list commands
- Packstack provisions VGs for you but you might have to do vgcreate, etc to do it manually

SDN and Neutron

In the definition of SDN is something that is easily forgotten - that the data plane and HW is a separate entity:
It says that the control plane resides in the software, (true we knew that) but that the data plane is "implemented in commodity network equipment". That last part is arguable since traffic channels are often virtualized
- Think of tenants. Network traffic needs to stay separated.
- BC domains defined in SW, need to be maintained across routers (VLANs assumed)
- For some reason, it is stated that traffic load and patterns is unpredictable.
- Emphasis is placed on need to be scalable and accommodate increased BW needs on-demand.
(scale out easily)

Illustrated (as if a great mystery solved by SDN) are 3 compute nodes separated by routers, and that each of their individuated VMs are in the same BC domain. "Network Decoupling" term used for SDN doing this.
- Control plane defined as a network service, Typically in Neutron configured in ml2_conf.ini
    - ml2 is a pluggable layer, allows plugins to define the overlay networking
- The differentiation is made (finally)
    VLAN is defined here as a "legacy method tough to configure because of physical dependencies"
    (My critique is it isn't tough to insert 802.1q fields in frames and you don't need HW!)
    VXLAN based on IP multicast to enable auto-discovery of the overlay MAC addresses
    GRE P2P or multipoint handled from a central router to connect endpoints. (remember no encryption)

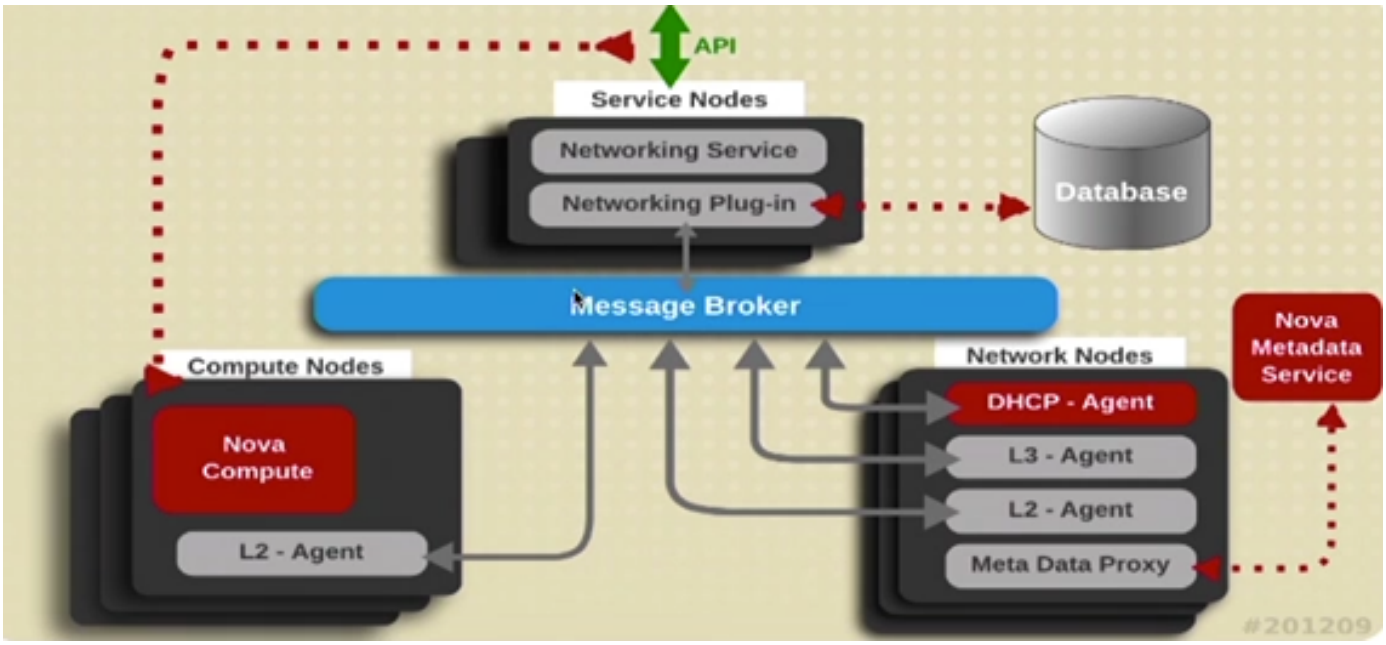
VXLAN has become standard, default.
VLAN is considered antiquated and confined to the stupid definition presented above

Cloud admin provides routers, subnets, external networks
Tenant admin can have multiple private networks to plug into those through Openstack services
- decides IPs to assign networks, create and connect to services

Public networks on other side of router for external connectivity include floating IPs
Plugins not only define the overlay, but are provided to connect to specific physical networks (underlay)

Compares floating IPs to source-NAT - expose an instance with a publically available to the internet on inside of the private network

- Tenant user creates internal network
  - Set DHCP on the subnet
  - Set default gateway to something on the subnet
- Tenant user creates external network, matching to a network that physically exists
  - Assign gateway
  - Do not use DHCP
  - Set floating IP addresses
- Tenant user creates the router
- Tenant admin marks the external network as being external
- Tenant user can now connect this router to the external network
- Tenant user assigns the internal network as an interface to the router
- Tenant user assigns some floating IPs to the network from the access and security tab



Compute nodes, running the instances,

- L2 OVS-agent / server component on node defines the broadcast domain in the overlay network (and the way networking is going to work in the environment)
- L2 OVS-agent on compute node talks to L2 OVS-agent on Network node
- L2 OVS-agent on the network node terminates SDN BC domain

Network node has everything it needs to define the control plane: DHCP agent, L3 stuff (routing), etc.
Often service and network node will be running on same machine
Service node:

- Networking plugin defines interactions with PHYS network, uses DB to keep it straight
- The networking service talks to the API, message broker glues all these different nodes together

The compute node runs neutron-server offering what the service node has (service and plugins) through it's OVS agent- the server component

Analyzing Packet Flow on a Neutron Network

- a VM boots, queries the neutron server, which is the software control plane
- the control plane uses OpenFlow to give info required to connect to the physical switch
- It is OpenFlow that shapes the SDN framework so the VM will know paths to the switch
- says "the switch itself is reduced to a dumb device" because the OpenFlow intelligence is instead in SW

InterVM Comms

VMs of the same tenant group might be on different hosts but need to be in the same subnet.
Routing all traffic through the network node to do this sucks and creates a SPOF
Instead, OpenFlow allows the VMs to talk to eachother directly

So OpenFlow generates a switching structure of a virtualized network
- Is not managed directly, but by Open vSwitch or other agents
- Replaces the control plane on switches, and works with managed and unmanaged switches
- Defines how forwarding planes should work and how they should be controlled from within the cloud
- Layer 2 based virtual interfaces for virtual machines,
- Bridges sit between the virtual and physical interfaces
- Open vSwitch in the BG calls OpenFlow to see how packets are to be dealt with
- OpenFlow rules govern forwarding between virtual machines, even on different physical nodes

Separating traffic between tenants
Different VMs need a different approach
VMs may belong to different customers. separation is a big deal
It should be separated enough that customers can use the same IP addresses without conflicting- which they can!

Linux Network Namespaces:
- are a container-based virtualization technology present in the Linux kernel stack (like chroot for networks)
- define separate areas and allow to distinguish the traffic between them.
- have their own network devices (i.e. virtual interfaces that are bridged into the physical device on the host) and separate IP addresses
Ultimately,
- Things in namespace A need routing to communicate with stuff in namespace B - even on the same host!
- Traffic generated in a namespace is visible only in that namespace

Command Line Tools
ovsdb-server - maintains the internal configuration database for Open vSwitch
ovs-vswitchd - switching daemon doing the network configuration
ovs-vsctl - manage Open vSwitch
ovs-ofctl - manage and monitor OpenFlow - to do deep inspection
ip netns show - show namespaces available
ip netns exec <namespace-identifier> COMMAND --- runs that command inside that namespace

```
[root@server1 ~]# ip netns show
qdhcp-e3043210-acf1-4fd1-9d3a-50d7832589eb
qdhcp-713d4e4a-84d5-4e61-98f1-b47353f330bd
qrouter-2995c2d5-2073-4608-bd1f-11e3a29aba2a
[root@server1 ~]# ip netns exec qrouter-2995c2d5-2073-4608-bd1f-11e3a29aba2a ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
11: qr-6d2046d4-71: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
    link/ether fa:16:3e:03:0a:b9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.25/24 brd 192.168.0.255 scope global qr-6d2046d4-71
       valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe03:ab9/64 scope link
       valid_lft forever preferred_lft forever
12: qr-9065fb6a-23: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
    link/ether fa:16:3e:fa:14:92 brd ff:ff:ff:ff:ff:ff
    inet 192.168.33.1/24 brd 192.168.33.255 scope global qr-9065fb6a-23
       valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fefa:1492/64 scope link
       valid_lft forever preferred_lft forever
13: qg-71b594dc-6f: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
    link/ether fa:16:3e:32:ff:de brd ff:ff:ff:ff:ff:ff
    inet 192.168.4.200/24 brd 192.168.4.255 scope global qg-71b594dc-6f
       valid_lft forever preferred_lft forever
    inet 192.168.4.201/32 brd 192.168.4.201 scope global qg-71b594dc-6f
       valid_lft forever preferred_lft forever
    inet 192.168.4.202/32 brd 192.168.4.202 scope global qg-71b594dc-6f
       valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe32:ffde/64 scope link
       valid_lft forever preferred_lft forever
```

Notice the multiple addresses in 13- those are floating IPs.

Interfaces
br-int (integration bridge) to integrate everything
br-tun (tunnel interface) define the network between the nodes
br-ex 0 external bridge - for external devices
eth0/1/2 your physical interfaces - on the networking node you need one for the external network and one for the different physical machines in the cloud environment (I am just reporting what it said here- this doesn't sound quite right, but see below).

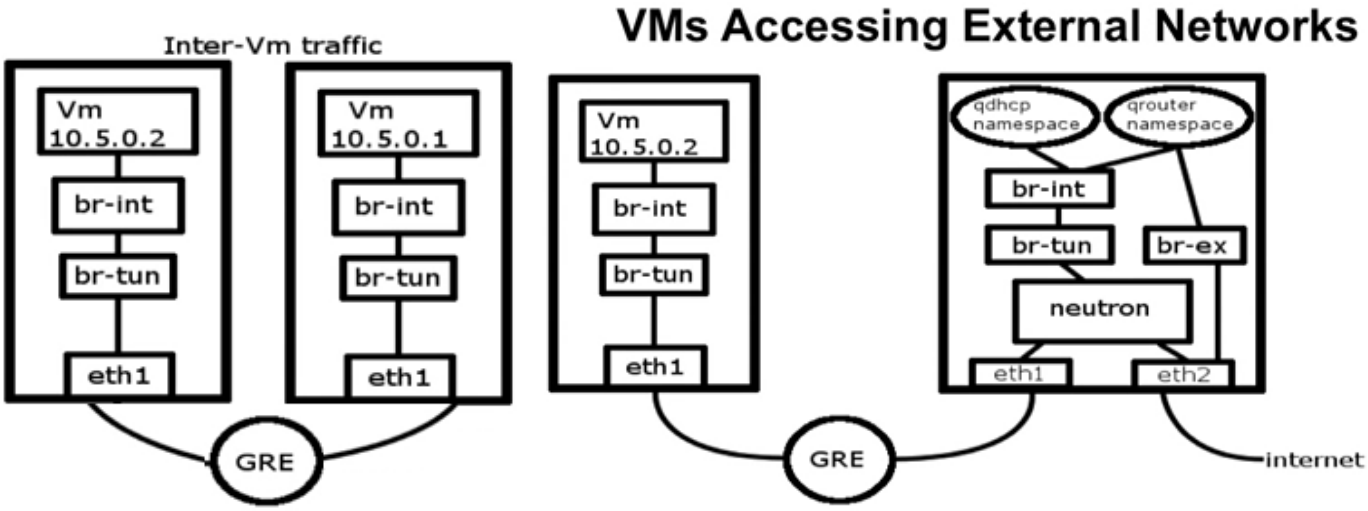
Typical network configuration:

Hypervisor node:

br-int - what the VMs are connected to
br-tun - used for tunneling
eth1 - internal physical interface used by br-tun
GRE tunnel - present on all hypervisor and network nodes for tunneling

Network node:

br-int / br-tun - similar to the hypervisor
br -ex - connected to eth0 for external comms
qrouter - one for every external network for cconnections to outside
qdhcp - one for every tenant, exists within GRE tunnel



ovs-vsctl show on compute node shows two bridges:

```
[root@compute ~]# ovs-vsctl show
09d585bc-18ff-4b42-b000-fb9736f318aa
    Bridge br-tun
        Port "gre-0a000115"
            Interface "gre-0a000115"
```

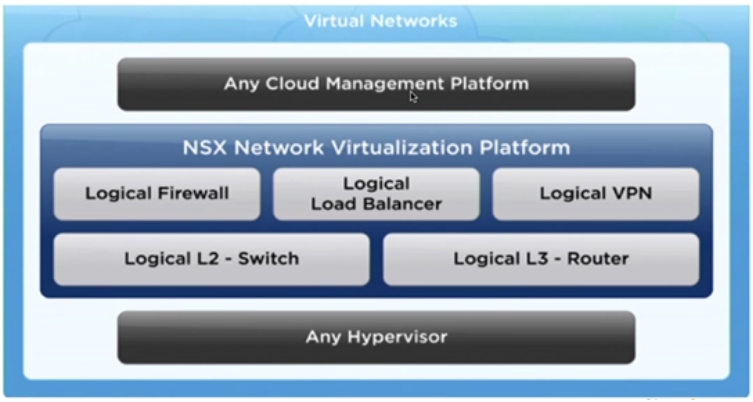
```
Bridge br-tun
        Port "gre-0a000115"
            Interface "gre-0a000115"
                type: gre
                options: {df_default="true", in_key=flow,
                local_ip="10.0.1.31", out_key=flow, remote_ip="10.0.1.21"}
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}
        Port br-tun
            Interface br-tun
                type: internal
```

```
Bridge br-int
        fail_mode: secure
        Port br-int
            Interface br-int
                type: internal
        Port patch-tun
            Interface patch-tun
                type: patch
                options: {peer=patch-int}
        Port "qvo2fbe1117-b4"
            tag: 1
            Interface "qvo2fbe1117-b4"
    ovs_version: "2.1.3"
```

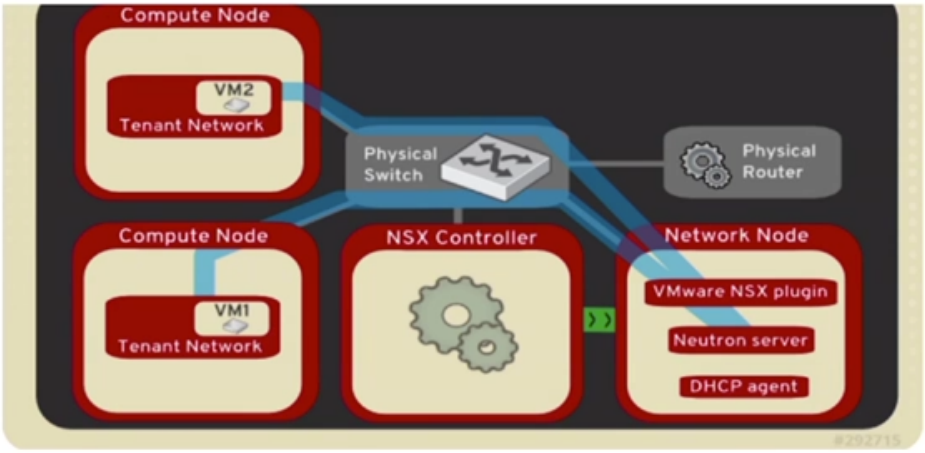
On br-tun:
Patch-int port shows the connection to patch-tun (which connects br-int to br-tun)
The gre interface shows which host the tunnel is connected to on the opposite end - the network node.
On br-int:
"tag: 1" Is the OpenFlow ID assigned to packet entering from qvo port
The patch-tun port shows it's connected to peer port patch-int (on br-tun)

The "ovs-ctl show" output can get really long/ messy on the neutron node. This is obviously split for easy viewing.

Understanding Backends
OpenStack + Open vSwitch the usual solution
Other backends might be provided by HW vendors or other open source solutions
Neutron just gives an API, so you can slap another onto it
Common Plugins:
Cisco UCS/ Nexus
Linux Bridge (outdated)
VMware NSX vCloud example



NSX offers:
VXLAN (Virtual eXtensible LAN)
STT- Stateless Transport Tunneling
NSX gateway for comms with non-NSX networks
NSX and RHEL sample layout, where th NSX plugin stands in for OVS plugin:



https://wiki.openstack.org/wiki/Cisco-neutron
https://wiki.openstack.org/wiki/Neutron_Plugins_and_Drivers
The second page says it is deprecated and does not refer to a new one; searching wiki also is "go fish"

Verifying Neutron Networking Services
Below, consider server1 commands for a node providing network services (Neutron node).
Server2 is just a compute node using network services (compute node)
On Server1's /etc/neutron/neutron.conf the ml2 entries defines which plugin to use

Log in to server1 and use the following to verify network controller services

- **systemctl status neutron-openvswitch-agent**
- **systemctl status neutron-dhcp-agent**
- **systemctl status neutron-l3-agent**
- **systemctl status neutron-ovs-cleanup**

On server2, type **systemctl status neutron-openvswitch-agent**

On server1, type **neutron agent-list**

- **keystone user-list --tenant services**
- **keystone user-get neutron**
- **keystone service-get neutron**
- **keystone endpoint-get --service network**

Check contents of **/etc/neutron/neutron.conf**

- **crudini --get /etc/neutron/neutron.conf DEFAULT**
- **crudini --get /etc/neutron/plugin.ini ml2 mechanism_drivers**
- **crudini --get /etc/neutron/plugin.ini ml2 tenant_network_types**

Check bridge configuration on server1:

- **cat /etc/sysconfig/network-scripts/ifcfg-br-ex**
- **ovs-vsctl show**

You should see br-ex and br-int. Br-ex is for external connectivity and contains the floating IPs, br-int is what the nodes internally will connect to.

Use **openstack-status** and verify the availability of the neutron services

The "neutron" service command has many options. Before you use it, you need to "source keystonerc_admin" in. Below we see that Server 2 and 3 are showing down (they aren't on)

```
[root@server1 ~]# source keystonerc_admin
[root@server1 ~(keystone_admin)]# neutron agent-list
+--------------------------------------+--------------------+--------------------+-------+----------------+---------------------------
--------+
| id                                   | agent_type         | host               | alive | admin_state_up | binary
       |
+--------------------------------------+--------------------+--------------------+-------+----------------+---------------------------
--------+
| 121f9fe9-9a6d-4d53-a5db-32f69eb0f400 | Open vSwitch agent | server2.example.com | xxx   | True           | neutron-openvswit
h-agent |
| 40148695-4b5e-4fd4-87ef-4695655be721 | L3 agent           | server1.example.com | :-)   | True           | neutron-l3-agent
       |
| 5aec8705-9a35-4518-9030-a709d4c44ce2 | Open vSwitch agent | server1.example.com | :-)   | True           | neutron-openvswit
h-agent |
| 71c21a9d-1675-4808-8b77-98d14c58704d | Open vSwitch agent | server3.example.com | xxx   | True           | neutron-openvswit
h-agent |
| e4444774-13ab-42df-8b93-997cb7afc173 | Metadata agent     | server1.example.com | :-)   | True           | neutron-metadata-
gent    |
| f4eb298f-2f67-48ce-8f47-8449f5b607c1 | DHCP agent         | server1.example.com | :-)   | True           | neutron-dhcp-agen
       |
+--------------------------------------+--------------------+--------------------+-------+----------------+---------------------------
--------+
```

We already know what the keystone commands will show us.
One interesting item is "keystone endpoint-get --service network". Using regular "keystone endpoint-list" we don't have this service specifically identified (network.publicURL on :9696) service-get wasn't demoed

Note the use of crudini to get rather than set values. Unfortunately, a key-value pair is not returned. The query for DEFAULT just spit out an unlabeled list of values. Specific queries such as "ml2 mechanism drivers" simply dumps out the specified value.

Configuring Tenant Networking in Horizon

This lesson was after the command line lesson in the following notes.
We are going to do this as a tenant administrator
Had to log in as cloud admin first, go to Identity>Projects>Edit to change the max network and floating IP quotas
Logged in as tenant admin again, go to Network>Networks>Create Network
Name arbitrary, network also arbitrary unused 192.168.200.0/24, put in a gateway IP you know***
In subnet details, allocation pools for the floating IPs range_start,range_end
Set a name server 8.8.8.8
For an external network, do the same an arbitrary unused IP block, make sure the gateway is correct for the external routing. Turn off DHCP on external.
Log in as cloud administrator to mark this as an external network (!) Tenant users must ask the cloud admin!
Click network, edit network, click external, save!
Log in as tenant admin again, click on routers, new, choose a external network we just approved.
Assign interfaces with interfaces tab, add your internal network, and you are done
Network topology should show all is good.

Configuring Tenant Networking at the command line

A public network has already been provisioned by the cloud administrator- configured on server1, the neutron node.
The tenant administrator/ cloud user sets up the rest.

The commands:

- Source the keystonerc_myuser credentials: **source /root/keystonerc_myuser**
- **neutron router-create router1**
- **neutron net-create private**
- **neutron subnet-create --name privatesub private 192.168.0.0/24**
- **neutron router-interface-add router1 privatesub**
- **neutron port-list**
- **neutron router-gateway-set router1 public**
- **neutron floatingip-create public**
- **neutron floatingip-list**

The procedure, in brief:

1. Create virtual router
2. Create private network
3. Create a subnet in that private network
4. Attach the subnet to the router
5. List the port configuration and get the ID of the public network ("public")
6. Establish a gateway to that public network
7. Make your floating IPs you'll need later and list them

Notes on the following demo:

- In the step after listing ports, it looks like we don't have a public network to add floating IPs to
- We need to log in as admin to do this:

neutron net-create public --tenant-id services --router:external

- It also needs a subnet so:

neutron subnet-create --tenant-id services --allocation-pool start=192.168.104.25,end=192.168.104.99 --
192.168.104.2 --disable-dhcp --name subdonotuse public 192.168.104..0/24

- Set the gateway

neutron router-gateway-set router2 public

- Switch to reg user and continue:

neutron floatingip-create public
neutron floatingip-list.

```
[root@server1 network-scripts(keystone_user)]# neutron router-list
+--------------------------------------+---------+--------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------+
| id                                   | name    | external_gateway_info
                                                                                                           |
+--------------------------------------+---------+--------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------+
| 2995c2d5-2073-4608-bd1f-11e3a29aba2a | router1 | {"network_id": "6a34a620-a2e6-4d82-878f-914a1808951d", "enable_snat": true,
"external_fixed_ips": [{"subnet_id": "a070846f-c09a-4286-8e4d-4ea10f9cf852", "ip_address": "192.168.4.200"}]} |
+--------------------------------------+---------+--------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------+
[root@server1 network-scripts(keystone_user)]# neutron router-create router2
Created a new router:
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| admin_state_up        | True                                 |
| external_gateway_info |                                      |
| id                    | a5b0c779-8d19-4ce7-8077-6f5e4b283f16 |
| name                  | router2                              |
| routes                |                                      |
| status                | ACTIVE                               |
| tenant_id             | 749116ea397e40a6b8664d2599764e17     |
+-----------------------+--------------------------------------+
[root@server1 network-scripts(keystone_user)]# neutron net-list
+--------------------------------------+-------------+-------------------------------------------------------+
| id                                   | name        | subnets                                               |
+--------------------------------------+-------------+-------------------------------------------------------+
| 6a34a620-a2e6-4d82-878f-914a1808951d | ext         | a070846f-c09a-4286-8e4d-4ea10f9cf852 192.168.4.0/24   |
| 713d4e4a-84d5-4e61-98f1-b47353f330bd | int         | 7197fe46-fbb5-449e-9e1e-82e295bb7b35 192.168.0.0/24   |
| e3043210-acf1-4fd1-9d3a-50d7832589eb | int-gooiweg | 7b0f1dc1-6a0f-4444-aafc-1fc79e42b164 192.168.33.0/24  |
+--------------------------------------+-------------+-------------------------------------------------------+
[root@server1 network-scripts(keystone_user)]# neutron net-create private
Created a new network:
+-----------------+--------------------------------------+
| Field           | Value                                |
+-----------------+--------------------------------------+
| admin_state_up  | True                                 |
| id              | ea49c625-6c3e-4ef6-863c-4a895d07f30d |
| mtu             | 0                                    |
| name            | private                              |
| router:external | False                                |
| shared          | False                                |
| status          | ACTIVE                               |
| subnets         |                                      |
| tenant_id       | 749116ea397e40a6b8664d2599764e17     |
+-----------------+--------------------------------------+
[root@server1 network-scripts(keystone_user)]# neutron subnet-create --name subpriv private 192.168.44.0/24
Created a new subnet:
+-------------------+----------------------------------------------------+
| Field             | Value                                              |
+-------------------+----------------------------------------------------+
| allocation_pools  | {"start": "192.168.44.2", "end": "192.168.44.254"} |
| cidr              | 192.168.44.0/24                                    |
| dns_nameservers   |                                                    |
| enable_dhcp       | True                                               |
| gateway_ip        | 192.168.44.1                                       |
| host_routes       |                                                    |
| id                | cbc0791a-24dc-48f3-a1a8-2799690daff6               |
| ip_version        | 4                                                  |
| ipv6_address_mode |                                                    |
| ipv6_ra_mode      |                                                    |
| name              | subpriv                                            |
| network_id        | ea49c625-6c3e-4ef6-863c-4a895d07f30d               |
| subnetpool_id     |                                                    |
| tenant_id         | 749116ea397e40a6b8664d2599764e17                   |
+-------------------+----------------------------------------------------+
[root@server1 network-scripts(keystone_user)]# neutron router-interface-add router2 subpriv
Added interface 7356b4e6-dcfc-42dc-8da8-30e342f60362 to router router2.
[root@server1 network-scripts(keystone_user)]# neutron port-list
+--------------------------------------+------+-------------------+--------------------------------------------------------------------------------
-------------------------+
| id                                   | name | mac_address       | fixed_ips
                         |
+--------------------------------------+------+-------------------+--------------------------------------------------------------------------------
-------------------------+
| 2102bafd-5d36-40eb-96c8-b33e9989c891 |      | fa:16:3e:a9:81:4c | {"subnet_id": "7197fe46-fbb5-449e-9e1e-82e295bb7b35", "ip_a
ddress": "192.168.0.200"}  |
| 34edde60-5695-473c-a583-c79253115cce |      | fa:16:3e:4d:33:4a | {"subnet_id": "7197fe46-fbb5-449e-9e1e-82e295bb7b35", "ip_a
ddress": "192.168.0.202"}  |
| 65d06d86-7015-4bf7-a101-b3912178a467 |      | fa:16:3e:71:d3:40 | {"subnet_id": "cbc0791a-24dc-48f3-a1a8-2799690daff6", "ip_a
ddress": "192.168.44.2"}   |
| 6d2046d4-71a9-46de-93f2-ec605acaafd3 |      | fa:16:3e:03:0a:b9 | {"subnet_id": "7197fe46-fbb5-449e-9e1e-82e295bb7b35", "ip_a
ddress": "192.168.0.25"}   |
| 7356b4e6-dcfc-42dc-8da8-30e342f60362 |      | fa:16:3e:b0:0d:84 | {"subnet_id": "cbc0791a-24dc-48f3-a1a8-2799690daff6", "ip_a
ddress": "192.168.44.1"}   |
| 9065fb6a-238a-41a2-ac6e-c0aa5c8fb3f3 |      | fa:16:3e:fa:14:92 | {"subnet_id": "7b0f1dc1-6a0f-4444-aafc-1fc79e42b164", "ip_a
ddress": "192.168.33.1"}   |
| 9efcdb8e-10db-4ad1-ac77-f544f7f1d7e1 |      | fa:16:3e:01:53:e8 | {"subnet_id": "7197fe46-fbb5-449e-9e1e-82e295bb7b35", "ip_a
ddress": "192.168.0.201"}  |
| a5fc4698-d719-4713-ac05-849f4e91d0a5 |      | fa:16:3e:70:a6:40 | {"subnet_id": "7197fe46-fbb5-449e-9e1e-82e295bb7b35", "ip_a
```

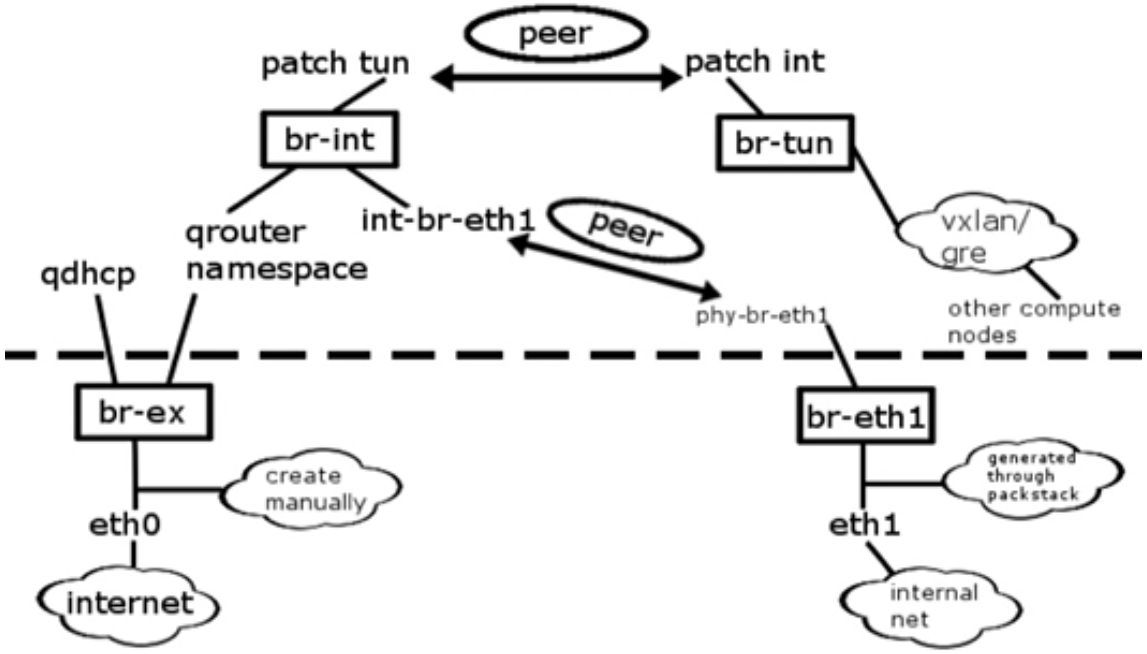
```
ddress": "192.168.0.201"}  |
| a5fc4698-d719-4713-ac05-849f4e91d0a5 |      | fa:16:3e:70:a6:40 | {"subnet_id": "7197fe46-fbb5-449e-9e1e-82e295bb7b35", "ip_a
ddress": "192.168.0.203"}  |
| a9c2592a-9f8f-4734-8c38-5ce306749c40 |      | fa:16:3e:99:d1:e4 | {"subnet_id": "7b0f1dc1-6a0f-4444-aafc-1fc79e42b164", "ip_a
ddress": "192.168.33.100"} |
+--------------------------------------+------+-------------------+-------------------------------------------------------------------------------------
--------------------------+
[root@server1 network-scripts(keystone_user)]# source /root/keystonerc_admin
[root@server1 network-scripts(keystone_admin)]# neutron net-create public --tenant-id services --router:external
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 1e32552e-a2b4-449b-962a-f68928bcdae2 |
| mtu                       | 0                                    |
| name                      | public                               |
| provider:network_type     | vxlan                                |
| provider:physical_network |                                      |
| provider:segmentation_id  | 76                                   |
| router:external           | True                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | services                             |
+---------------------------+--------------------------------------+
[root@server1 network-scripts(keystone_admin)]# neutron subnet-list
+--------------------------------------+-----------------+-----------------+-------------------------------------------------------------------------------------
---+
| id                                   | name            | cidr            | allocation_pools
   |
+--------------------------------------+-----------------+-----------------+-------------------------------------------------------------------------------------
---+
| a070846f-c09a-4286-8e4d-4ea10f9cf852 | subext          | 192.168.4.0/24  | {"start": "192.168.4.200", "end": "192.168.4.250"}
   |
| 7197fe46-fbb5-449e-9e1e-82e295bb7b35 | subint          | 192.168.0.0/24  | {"start": "192.168.0.200", "end": "192.168.0.250"}
   |
| 7b0f1dc1-6a0f-4444-aafc-1fc79e42b164 | int-gooiweg-sub | 192.168.33.0/24 | {"start": "192.168.33.100", "end": "192.168.33.150
"} |
| cbc0791a-24dc-48f3-a1a8-2799690daff6 | subpriv         | 192.168.44.0/24 | {"start": "192.168.44.2", "end": "192.168.44.254"}
   |
+--------------------------------------+-----------------+-----------------+-------------------------------------------------------------------------------------
---+
[root@server1 network-scripts(keystone_admin)]# neutron subnet-create --tenant-id services --allocation-pool start=192.168.104.
25,end=192.168.104.99 --gateway 192.168.104.2 --disable-dhcp --name subdonotuse public 192.168.104.0/24
Created a new subnet:
+-------------------+------------------------------------------------------+
| Field             | Value                                                |
+-------------------+------------------------------------------------------+
| allocation_pools  | {"start": "192.168.104.25", "end": "192.168.104.99"} |
| cidr              | 192.168.104.0/24                                     |
| dns_nameservers   |                                                      |
| enable_dhcp       | False                                                |
| gateway_ip        | 192.168.104.2                                        |
| host_routes       |                                                      |
| id                | bc4ef426-1973-4562-92e4-630e04afbc21                 |
| ip_version        | 4                                                    |
| ipv6_address_mode |                                                      |
| ipv6_ra_mode      |                                                      |
| name              | subdonotuse                                          |
| network_id        | 1e32552e-a2b4-449b-962a-f68928bcdae2                 |
| subnetpool_id     |                                                      |
| tenant_id         | services                                             |
+-------------------+------------------------------------------------------+
[root@server1 network-scripts(keystone_admin)]# neutron router-gateway-set router2 public
Set gateway for router router2
[root@server1 network-scripts(keystone_admin)]# source /root/keystonerc_user
[root@server1 network-scripts(keystone_user)]# neutron floatingip-create public
Created a new floatingip:
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    |                                      |
| floating_ip_address | 192.168.104.26                       |
| floating_network_id | 1e32552e-a2b4-449b-962a-f68928bcdae2 |
| id                  | 9695bc2b-f412-493c-b2f3-24bb732b524e |
| port_id             |                                      |
| router_id           |                                      |
| status              | DOWN                                 |
| tenant_id           | 749116ea397e40a6b8664d2599764e17     |
+---------------------+--------------------------------------+
[root@server1 network-scripts(keystone_user)]# neutron floatingip-list
+--------------------------------------+------------------+---------------------+--------------------------------------+
| id                                   | fixed_ip_address | floating_ip_address | port_id                              |
+--------------------------------------+------------------+---------------------+--------------------------------------+
| 377876bc-6964-441a-8590-b7d0a1a0fd0c | 192.168.0.201    | 192.168.4.201       | 9efcdb8e-10db-4ad1-ac77-f544f7f1d7e1 |
| 4f174420-590f-4e9f-ab3d-7c41b7558960 |                  | 192.168.4.204       |                                      |
| 9695bc2b-f412-493c-b2f3-24bb732b524e |                  | 192.168.104.26      |                                      |
| bb07bf4c-d33e-4ec3-bb3f-64b10aa281e1 | 192.168.0.202    | 192.168.4.202       | 34edde60-5695-473c-a583-c79253115cce |
+--------------------------------------+------------------+---------------------+--------------------------------------+
[root@server1 network-scripts(keystone_user)]#
```

Troubleshooting Neutron Networking

- Pinging the instance
- openstack-service status neutron
- tail /var/log/neutron/*.log
- As a tenant admin, neutron net-list, subnet-list, and router-list
- Make sure the router is connected and has a working external gateway

Above the line is the overlay virtual network
Below is the physical network

Remember- the br-int is the integration bridge- integrating a connection into the namespace.




```
[root@server1 ~]# ip netns show
qrouter-f989330f-b76e-4153-9223-90ab97963007
[root@server1 ~]# ip netns exec qrouter-f989330f-b76e-4153-9223-90ab97963007 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
```

ip netns show "show me the namespaces!"
ipnetns exec qrouter -.... ip addr show "whats up?"
this dispaly is showing nothing is up. Router has an internal address but nothing is going on.

11.2 config tenant networking, next 2 also long - networking
The last 10 get really long esp configuring packstack... these are part of lesson14- an exam lab, and reviews just about everything configuration-wise

Deploying an instance from the command line

source keystonerc_admin
nova list (it's blank)
nova list --all-tenants --shows instances on all tentants (not jus the admin)
nova boot --flavor m1.tiny --image web --poll test

```
+--------------------------------------+--------------------------------------------------+
| Property                             | Value                                            |
+--------------------------------------+--------------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                           |
| OS-EXT-AZ:availability_zone          |                                                  |
| OS-EXT-SRV-ATTR:host                 | -                                                |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | -                                                |
| OS-EXT-SRV-ATTR:instance_name        | instance-00000003                                |
| OS-EXT-STS:power_state               | 0                                                |
| OS-EXT-STS:task_state                | scheduling                                       |
| OS-EXT-STS:vm_state                  | building                                         |
| OS-SRV-USG:launched_at               | -                                                |
| OS-SRV-USG:terminated_at             | -                                                |
| accessIPv4                           |                                                  |
| accessIPv6                           |                                                  |
| adminPass                            | C6R4w4BbHCco                                     |
| config_drive                         |                                                  |
| created                              | 2015-11-13T22:46:22Z                             |
| flavor                               | m1.tiny (1)                                      |
| hostId                               |                                                  |
| id                                   | 3bcfeab8-30e5-4cec-a14e-cbe9601e3582             |
| image                                | web (c23594e0-69c8-49ae-8d9b-2e64eb01b7cf)       |
| key_name                             | -                                                |
| metadata                             | {}                                               |
| name                                 | test                                             |
| os-extended-volumes:volumes_attached | []                                               |
| progress                             | 0                                                |
| security_groups                      | default                                          |
| status                               | BUILD                                            |
| tenant_id                            | 1e1ab59958d54616a6174bdf74d24384                 |
| updated                              | 2015-11-13T22:46:23Z                             |
| user_id                              | 20dbef3a7b1d48738e4f2eb23165879a                 |
+--------------------------------------+--------------------------------------------------+

Server building... 100% complete
Finished
```

And that's it! Off it goes!

```
[root@server1 ~(keystone_admin)]# nova list --all-tenants
+--------------------------------------+------+----------------------------------+--------+------------+-------------
-+
| ID                                   | Name | Tenant ID                        | Status | Task State | Power State
 |
+--------------------------------------+------+----------------------------------+--------+------------+-------------
-+
| 3bcfeab8-30e5-4cec-a14e-cbe9601e3582 | test | 1e1ab59958d54616a6174bdf74d24384 | ACTIVE | -          | Running
 |
| 90e44ebf-3e2e-41c4-a88e-5aa7ff8dacca | web1 | ee4bf296ccf24dd8a964ffec8bb33f1b | ERROR  | -          | NOSTATE
 |
| 6a3ac87d-94f5-46fe-8b5e-20095aa82bc6 | web2 | ee4bf296ccf24dd8a964ffec8bb33f1b | ERROR  | -          | NOSTATE
 |
+--------------------------------------+------+----------------------------------+--------+------------+-------------
```

Troubleshooting Nova and Instances

http://virtual2privatecloud.com/troubleshooting-openstack-nova/ |---site down- saved WBM copy
/var/log/nova/nova-compute.log
/var/log/nova/instances
/var/log/libvirt/
Check networking
mysql -u root -p
    use nova;
    show tables;
    show columns from instances;
    select * from instances;
grep MARIA /root/answers.txt
- login info for DB incl passwd
Node comes up but can't reach it?
Might have a floating IP or routing issue.
ip netns list
ip netns exec <id> ip a
ip netns exec <id> ip ping <address> (can you ping it from inside the namespace?

Adding and Removing Compute Nodes
nova list --all-tenants --- check to see if it's running

nova delete <vm-id> -- to stop it
nova service-disable --reason "needstogo" server2.example.com nova-compute
 - disable the compute service on Server2
nova service-list
-- verify + get the ID of the Server2 instance
nova service-delete 6; nova service-list   - still shows but should be disabled
 - Delete the server2 compute service ID from the list using the ID you got
- It will come back on reboot, although it is said nova service-disable
This part didn't really make things clear - like "nova delete <vm-id>" vs "nova service-delete"
THIS SHOULD BE REVISITED AND EDITED FOR ACCURACY

Adding a Compute Node
Check the instructions at the beginning of the packet, and adding the Openstack repo
Install openstack-nova-compute package
Configure /etc/nova/nova.conf to have it listen on 172.24.101.0/24

my_ip=172.24.101.20
vncserver_proxyclient_address=$my_ip
vncserver_listen= 0.0.0.0
glance_host=172.24.101.10

Managing Instances
As tenant user - source keystonerc_user
nova list; nova show small; nova console-log small
nova hypervisor-list; source keystonerc_admin and retry if needed
nova migrate web or live-migrate -- moves instance "web" to another location
/etc/nova/policy.json   --says has more ineresting stuff
        look for admin_actions:migrate
        change rule:admin_api to rule:admin_or_owner
nova migrate (try again)
nova --help   TONS - ignore network comands and use Neutron for those

Overview review
Essentials: compute (hypervisor), controller, object storage, networking
Controller essentials - MQ, DB, Keystone.
Obj storage - with Swift at least 3 diff servers
A typical cloud at least 10 different hosts
Bigger it gets more specific the hosts can be, more machines
HA - each host redundant - especially essentials
Security
 - pub accessible and administrative parts clearly distinguished
 - Sure VMs need to be accessible, but it doesn't mean every hypervisor node needs to be accessible
 - Neutron using specific network nodes for external connectivity
 - No public access for backend services
 - API endpoints never need to be in public space - they need to be kept in a demarc'd private area
 - Instances DO usually need to be accessible to be functional
 - Think about cloud purpose.
        Compute focus? Service orientation like AWS?
        Storage focus with lots of nodes, particularly dozens of Ceph nodes?
        Network focused?
        How big a deal is multisite?
                Is it for redundancy, regional access, both?
Deployment / Automation
 - Puppet, Chef
 - Triple-O (Openstack on Openstack) a common method
        Nova, neutron, heat to automate management
        Ironic for deployment, using PXE and IPMI; pluggable drivers, Ironic-specific glance images
 - Red Hat Foreman, SUSE Crowbar
Requirements:
 - Physical installation of nodes - PXE (boot) and TFTP (image delivery + other repo)
 - drive setup, net configs, PXE working

IPMI (usually?) lives on a separate board with miniOS to do it's thing

FINAL Part is comprised of the last videos, 92-99.mp4

Final Exam Prep lab: server1 tasks

- Reset your servers to their original state. Configure server1.example.com as follows:
- Use pool.ntp.org as NTP server for all machines
- Set the eth0 interface for internet connectivity. On the eth1 interface, configure internal networking, using the 172.16.0.0/24 IP address range.
- Create a private overlay network with the name int and a subnet with the name subint, using the 192.168.0.0/24 network
- Create a public network named ext and a subnet subext, using your externally assigned IP address range. Set the gateway for this network, and provide a pool of floating IP addresses in the range 192.168.4.200 to 192.168.4.250
- Use VXLAN as the networking type
- On server1, configure br-ex as the external bridge. Configure br-eth1 as the network bridge for networking between server1 and server2. br-eth1 must be attached to eth1 on both server1 as server2.
- Configure Horizon to accept SSL connections
- Use the cinder-volumes group for the cinder volume service
- Create a project with the name workshop, using the description Project for workshop. Set quota to 4 instances, v VCPUs, 4096MB RAM and 12 floating IP addresses
- Create a new flavor with the name m2.tiny, which includes one VCPU, 1024 MB RAM, a 2GB root disk, a 1GB ephemeral disk and a 256 MB swap disk
- Create a tenant user with the name user, with the email address user@server1.example.com. Set the password to "password" and include this user as a member of the workshop project
- Create a tenant user with the name adm, with the email address admin@server1.example.com. Set the password to "password" and include this user as a member of the workshop project. Give this user administrator privileges
- In the workshop project, create a new image with the name small, using Cirros 64 bit Qcow image from the cirros web site. Set no minimum disk and minimum RAM to 512 MB.
- In the workshop project, create a new image with the name web, using the Cirros 64 bit Qcow image from the Cirros web site. Set no minimum disk and minimum RAM to 1024 MB
- Allocate two floating IP addresses to the project: 192.168.4.201 and 192.168.4.202
- Create a new security group **sec1** with the description Web and SSH. Allow TCP/22, TCP/443 and ALL ICMP from all networks, and HTTP/80 only from within the sec1 security group.
- Create an SSH key pair named key1. Save the private key to /home/user/Downloads/key1.pem on workstation.example.com
- In workshop, launch a new instance named small using the small image and the m2.tiny flavor. Include the key1 key pair and the sec1 security group. Associate the 192.168.4.201 floating IP address
- Also launch a new instance named web, using the web image and the m2.tiny flavor. Include the key1 key pair and the sec1 security group. Associate the 192.168.4.202 floating IP address
- In the workshop project, create a new 1 GB volume named vol1 and attach this to the web instance.
- Deploy swift, ceilometer and nagios.