OpenStack Cloud Deployment on UCS B-Series Servers and UCS Fabric

This Tech Note covers setting up an OpenStack Cloud (Cactus release), comprising a cluster of compute controller and compute nodes running RHEL 6.0. Each node is a Cisco UCS B200 Blade on a UCS B5100 Blade Server. This document lists the steps followed in the Cisco lab and includes observations in bringing up OpenStack on the Cisco Unified Computing System (UCS) platform. It builds on installation instructions described in *OpenStack Compute and Storage Administration Guides*, but is not intended to supercede those documents.

Table of Contents

Table of Figures

Introduction

OpenStack is a collection of open source technologies that provide massively scalable open source cloud computing software. This Tech Note documents our experience in setting up an OpenStack Cloud comprising a cluster of compute controller and compute nodes running RHEL 6.0. Each node is a Cisco UCS B200 Blade on a UCS B5100 Blade Server. This document lists steps followed in our lab and includes observations in bringing up OpenStack on the UCS platform. It builds on the installation instructions described in *OpenStack Compute and Storage Administration Guides*[1], but is a more streamlined method specific to our deployment.

Cisco UCS B5100 Blade Server[2]

The Cisco UCS B5100 is a blade server based on Intel® Xeon® processor 5500 and Xeon 5600 series. These servers work with virtualized and nonvirtualized applications to increase:

- Performance
- Energy efficiency
- Flexibility

Our OpenStack installation is on B200 Blades on the UCS chassis which has the following features:

- Up to two Intel® Xeon® 5500 Series processors, which automatically and intelligently adjust server performance according to application needs, increasing performance when needed and achieving substantial energy savings when not.
- Up to 96 GB of DDR3 memory in a half-width form factor for mainstream workloads, which serves to balance memory capacity and over all density.
- Two optional Small Form Factor (SFF) Serial Attached SCSI (SAS) hard drives available in 73GB 15K RPM and 146GB 10K RPM versions with an LSI Logic 1064e controller and integrated RAID.
- One dual-port mezzanine card for up to 20 Gbps of I/O per blade. Mezzanine card options include either a Cisco UCS VIC M81KR Virtual Interface Card, a converged network adapter (Emulex or QLogic compatible), or a single 10GB Ethernet Adapter.

UCS Fabric Topology

Our deployment consists of a cluster of eight B200 blades on a chassis interconnected by UCS 6120 Fabric Interconnect Switch. One server serves as the OpenStack *Cloud Controller*. The

[1] http://docs.openstack.org/
[2] http://www.cisco.com/en/US/products/ps10279/index.html

other servers are configured as compute nodes. We deployed the OpenStack network with VLAN model, so that the OpenStack management/control network is separate from the data network. (By management/control network, we imply the network which is used to access the servers, and on which the OpenStack processes exchange messages. By data network, we imply the network on which the virtual machines instantiated by OpenStack communicate with each other.) This is achieved by the host side VLAN tagging done at the OpenStack controlled VLAN network configuration. Figure 1 shows the topology.

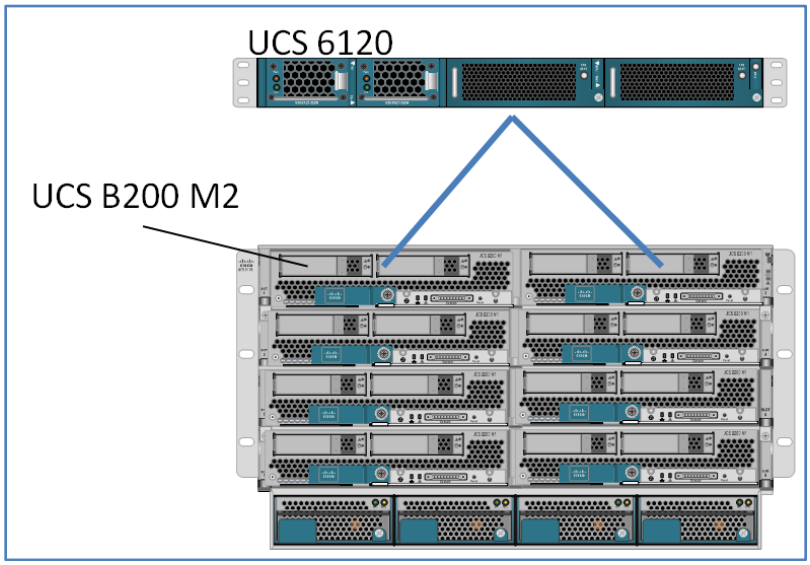


Figure 1: OpenStack Cloud Deployment on a B5100 UCS cluster

Installation on the Cloud Controller

Creating Service Profiles in a UCS Server

The UCS Manager (UCSM) provides facility to create *Service Profiles* which represent a logical view of a single blade server. The profile object contains the server personality, identity and network information. The profile can then be associated with a single blade at a time. Using service profiles, a user can create/configure virtual network interfaces and their properties. Using the same UCSM application, users can also configure and manage network configuration for UCS fabric 6120 that interconnects the chassis.

For more information on creating/managing UCS B Series Blade Servers and UCS fabric, please refer to product literature at http://www.cisco.com/en/US/products/ps10280/index.html

The following figures show screenshots of UCSM and are the important steps for enabling VLAN trunking at the UCS Fabric and UCS Blade service profile to support OpenStack with VLAN network model.

It's assumed that the Service profile is created and configured with all the parameters relevant to the server and basic network properties.

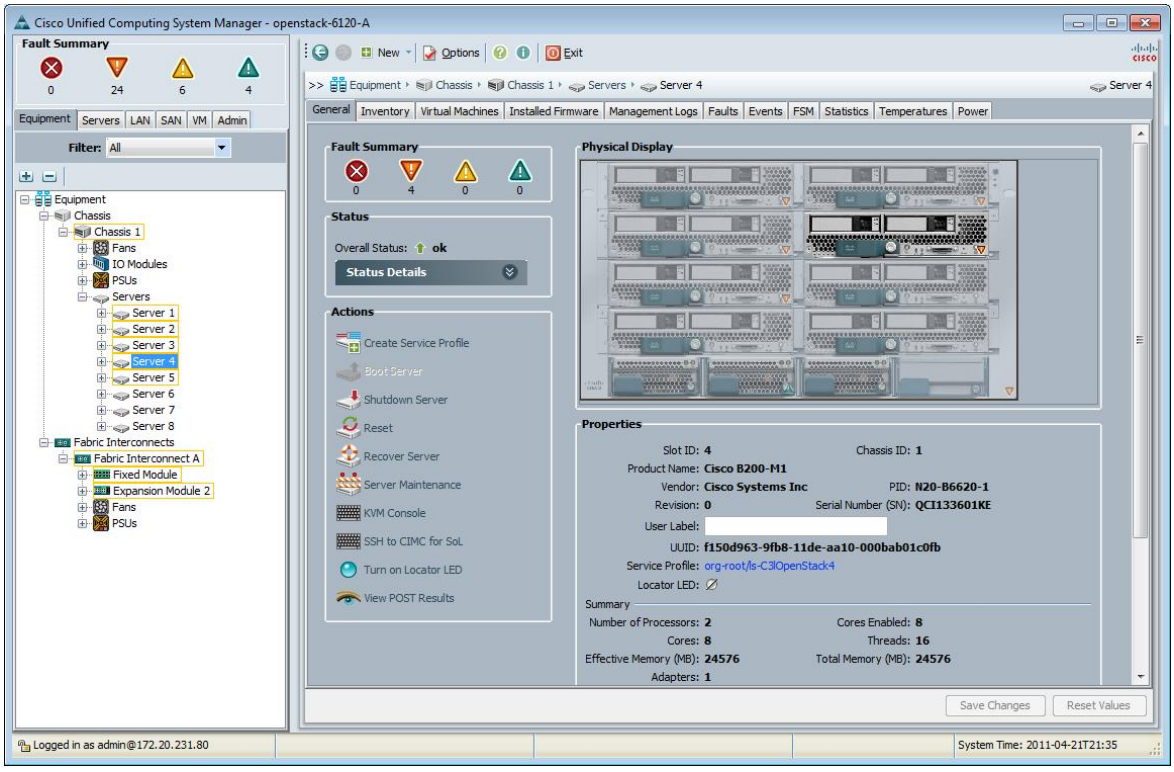


Figure 2: Cisco UCS-M service profile window

Once the service profile is created and associated to a blade in a UCS chassis, open the network tab and enable VLAN trunking as shown below.

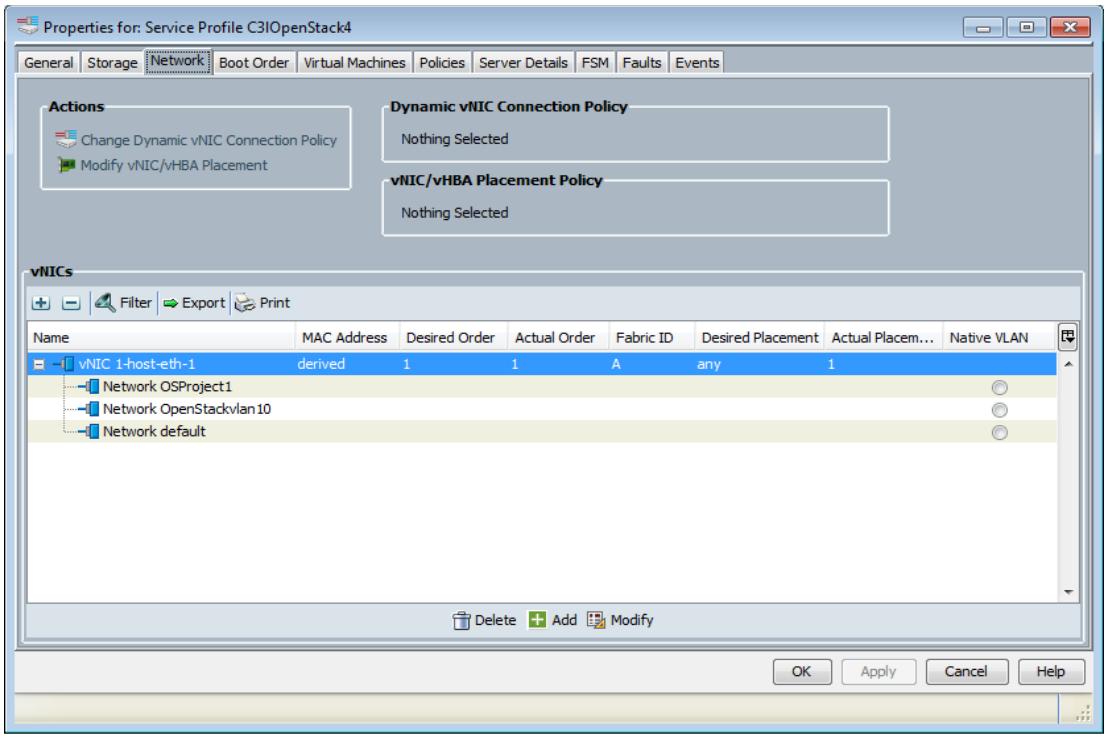


Figure 3: Cisco UCS-MvNIC configuration

Select and open the vNIC configuration window and "enable VLAN" trunking and select all the VLAN groups that are relevant for this vNIC as shown below.

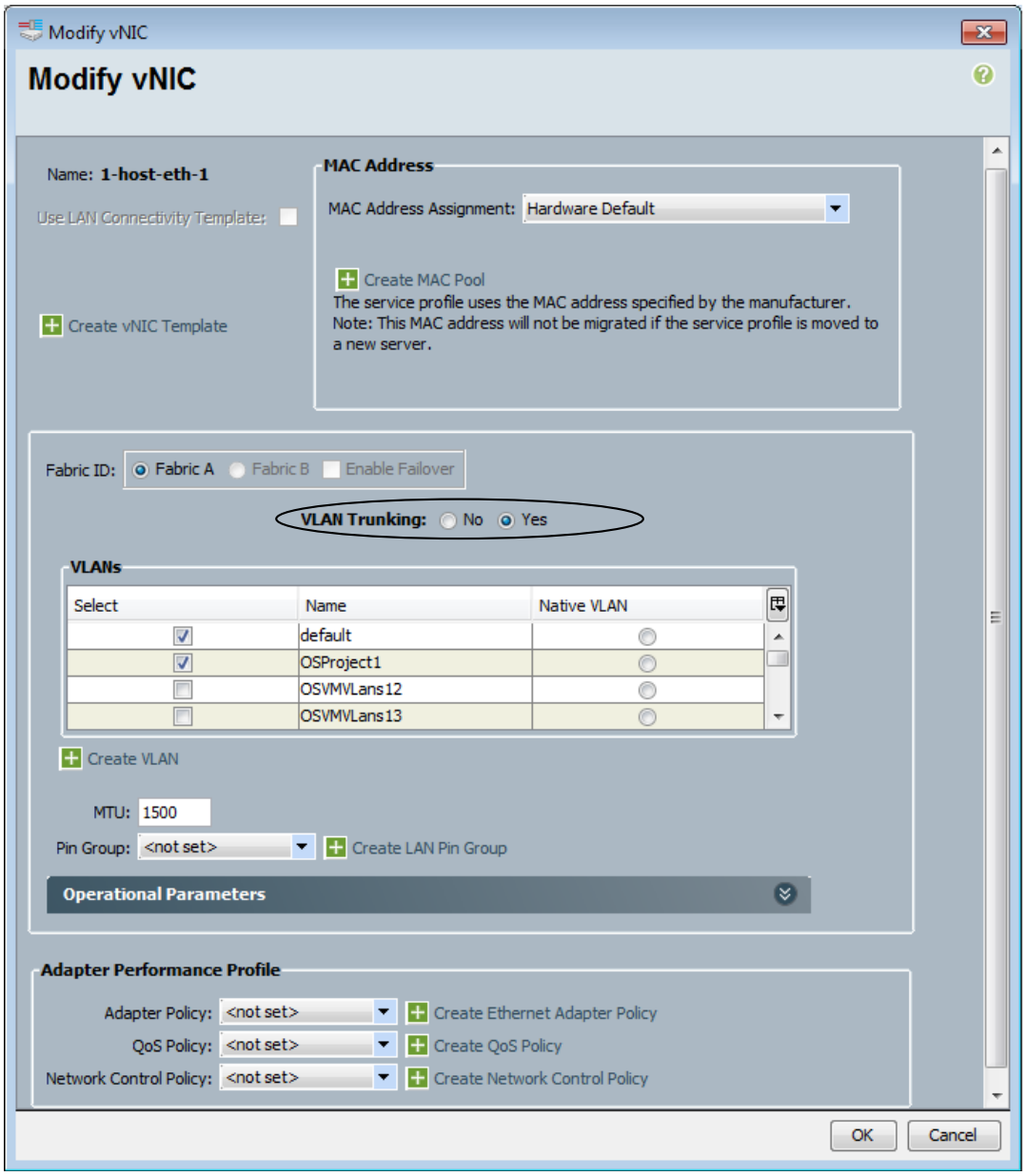


Figure 4: Enabling VLAN Trunking for the vNIC

Now the UCS blade is ready to support VLAN tagging at the host side for the Openstack installation.

Installing OpenStack Components

The installation of OpenStack components from RHEL repo as shown in the Openstack documentation at http://docs.openstack.org/cactus/openstack-compute/admin/content/inst

alling-openstack-compute-on-rhel6.html works well on both the Cloud Controller and also on the other compute nodes. We will follow that approach for the installation. In our installation, we will run all the services on the Cloud Controller, and only the *nova-compute* service on the compute nodes. Note that in this setup, the Cloud Controller also serves as one of the compute nodes. We suggest this approach since you can get started running and testing virtual machine instances even with installing just the Cloud Controller, and add one or more compute nodes later as required.

Installing OpenStack Nova-compute

After updating the /etc/yum.repos.d/openstack.repo as above mentioned in the openstack wiki, please use the following commands to install OpenStack components in on UCS Blade Server with RHEL 6.0

```
[root@c3l-openstack4 /]# yum install openstack-nova-compute openstack-nova-compute-config
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
openstack-nova-deps                                       | 1.3 kB     00:00
Setting up Install Process
Package openstack-nova-compute-2011.1.1-4.noarch already installed and latest version
Resolving Dependencies
--> Running transaction check
---> Package openstack-nova-compute-config.noarch 0:2011.1.1-1 set to be updated
--> Processing Conflict: openstack-nova-cc-config-2011.1.1-1.noarch conflicts openstack-nova-
compute-config = 2011.1.1
--> Processing Conflict: openstack-nova-compute-config-2011.1.1-1.noarch conflicts openstack-
nova-cc-config = 2011.1.1
--> Finished Dependency Resolution
```

<clipped>

```
Installed:
  openstack-nova-compute.noarch 0:2011.1.1-4
```

Follow the same method to install other OpenStack components:

```
sudo yum install euca2ools openstack-nova-{api,network,objectstore,scheduler,volume openstack-
glance
```

The *nova-objectstore* installation screen output is shown below:

```
Resolving Dependencies
--> Running transaction check
---> Package openstack-nova-objectstore.noarch 0:2011.1.1-4 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                    Arch      Version        Repository          Size
================================================================================
Installing:
 openstack-nova-objectstore    noarch    2011.1.1-4     openstack-nova     10 k
```

<clipped>

```
Running Transaction
  Installing     : openstack-nova-objectstore-2011.1.1-4.noarch             1/1

Installed:
```

```
  openstack-nova-objectstore.noarch 0:2011.1.1-4

Complete!
```

Bringing Up Openstack Components

The below screen capture shows the MySQL initialization:

```
[root@c3l-openstack4 /]# chkconfig mysqld
[root@c3l-openstack4 /]# service mysqld status
mysqld is stopped
[root@c3l-openstack4 /]# service mysqld start
Initializing MySQL database:  Installing MySQL system tables...
OK
Filling help tables...
OK

To start mysqld at boot time you have to copy
support-files/mysql.server to the right place for your system

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !
To do so, start the server, then issue the following commands:

/usr/bin/mysqladmin -u root password 'new-password'
/usr/bin/mysqladmin -u root -h c3l-openstack4 password 'new-password'

Alternatively you can run:
/usr/bin/mysql_secure_installation

which will also give you the option of removing the test
databases and anonymous user created by default.  This is
strongly recommended for production servers.

See the manual for more instructions.

You can start the MySQL daemon with:
cd /usr ; /usr/bin/mysqld_safe &

You can test the MySQL daemon with mysql-test-run.pl
cd /usr/mysql-test ; perl mysql-test-run.pl

Please report any problems with the /usr/bin/mysqlbug script!

                                                           [  OK  ]
Starting mysqld:                                           [  OK  ]
[root@c3l-openstack4 /]# chkconfig mysqld on
```

Next, configure mysqld and set neccssary user permissions. Create a mysqlamin.sh shell script (as shown in the OpenStack wiki page referenced above)

```
#!/bin/bash

DB_NAME=nova
DB_USER=nova
DB_PASS=nova
PWD=nova

CC_HOST= "172.20.231.73"
HOSTS='' # compute nodes list

mysqladmin -uroot -p$PWD -f drop nova
mysqladmin -uroot -p$PWD create nova

for h in $HOSTS localhost; do
        echo "GRANT ALL PRIVILEGES ON $DB_NAME.* TO '$DB_USER'@'$h' IDENTIFIED BY '$DB_PASS';" |
mysql -uroot -p$DB_PASS mysql
done
```

```
echo "GRANT ALL PRIVILEGES ON $DB_NAME.* TO $DB_USER IDENTIFIED BY '$DB_PASS';" | mysql -uroot -
p$DB_PASS mysql
echo "GRANT ALL PRIVILEGES ON $DB_NAME.* TO root IDENTIFIED BY '$DB_PASS';" | mysql -uroot -
p$DB_PASS mysql

nova-manage db sync
```

Executing the above script sets up *mysqld* for OpenStack components use.

```
[root@c3l-openstack4 openstack]# ./mysqladmin.sh
./mysqladmin.sh: line 8: 172.20.231.73: command not found
mysqladmin: DROP DATABASE nova failed;
error: 'Can't drop database 'nova'; database doesn't exist'
2011-03-09 09:35:50,704 migrate.versioning.api: 0 -> 1...
2011-03-09 09:35:51,227 migrate.versioning.api: done
2011-03-09 09:35:51,227 migrate.versioning.api: 1 -> 2...
2011-03-09 09:35:51,518 migrate.versioning.api: done
```

Now create project user and required credentials

```
[root@c3l-openstack4 ~]# nova-manage user admin root
2011-03-09 12:22:37,106 nova.auth.manager: Created user root (admin: True)
export EC2_ACCESS_KEY=cfc59c93-47a3-4a91-9222-e26721ddf49e
export EC2_SECRET_KEY=8dc6fcdb-ea26-4679-a6c8-dc3a8bb9ac56
[root@c3l-openstack4 ~]#
```

Make a note of the username and the project name that you enter here.

Currently only one network is supported per project.

```
[root@c3l-openstack4 ~]# nova-manage project create cisco1 root
2011-03-09 12:23:54,526 nova.auth.manager: Created project cisco1 with manager root
[root@c3l-openstack4 ~]#
-----------------
```

Now create a network for the project.

```
[root@c3l-openstack4 ~]# nova-manage network create 192.168.0.0/24 1 255
```

Generate nova credential file.

```
[root@c3l-openstack4 ~]# cd /root/openstack/
[root@c3l-openstack4 openstack]# mkdir -p creds
[root@c3l-openstack4 openstack]# cd creds
[root@c3l-openstack4 creds]# nova-manage project zipfile cisco1 root ./novacreds.zip


[root@c3l-openstack4 creds]# unzip novacreds.zip -d .
Archive:  novacreds.zip
 extracting: ./novarc
 extracting: ./pk.pem
 extracting: ./cert.pem
 extracting: ./cacert.pem
[root@c3l-openstack4 creds]# ls
cacert.pem  cert.pem  novacreds.zip  novarc  pk.pem
[root@c3l-openstack4 creds]# source ./novarc
[root@c3l-openstack4 creds]# . ./novarc
```

Now restart all the OpenStack components at the Cloud Controller:

```
service libvirt-bin restart; service nova-network restart; service nova-compute restart; service
nova-api restart; service nova-objectstore restart; service nova-scheduler restart
```

You can verify the mysqld setup by issuing

```
[root@c3l-openstack4 openstack]# mysql -uroot -pnova
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.1.47 Source distribution

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases
    -> ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| nova               |
| test               |
+--------------------+
4 rows in set (0.00 sec)

mysql> use nova;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables
    -> ;
+-------------------------------------+
| Tables_in_nova                      |
+-------------------------------------+
| auth_tokens                         |
| certificates                        |
| console_pools                       |
| consoles                            |
| export_devices                      |
| fixed_ips                           |
| floating_ips                        |
| instance_actions                    |
| instances                           |
| iscsi_targets                       |
| key_pairs                           |
| migrate_version                     |
| networks                            |
| projects                            |
| quotas                              |
| security_group_instance_association |
| security_group_rules                |
| security_groups                     |
| services                            |
| user_project_association            |
| user_project_role_association       |
| user_role_association               |
| users                               |
| volumes                             |
+-------------------------------------+
24 rows in set (0.01 sec)

mysql>
mysql>
```

Post OpenStack Installation Configuration

Once the installation has completed successfully, you will see that a /root/creds/novarc file has been created.

The novarc file will look like this:

```
[root@c3l-openstack4 creds]# cat novarc
NOVA_KEY_DIR=$(pushd $(dirname $BASH_SOURCE)>/dev/null; pwd; popd>/dev/null)
export EC2_ACCESS_KEY="cfc59c93-47a3-4a92-9222-e26721ddf49e:cisco1"
export EC2_SECRET_KEY="8dc6fcdb-ea26-4679-a6c8-dc3a8bb9ac56"
export EC2_URL="http://172.20.xxx.yyy:8773/services/Cloud"
export S3_URL="http://172.20.aaa.qqq:3333"
export EC2_USER_ID=42 # nova does not use user id, but bundling requires it
export EC2_PRIVATE_KEY=${NOVA_KEY_DIR}/pk.pem
export EC2_CERT=${NOVA_KEY_DIR}/cert.pem
export NOVA_CERT=${NOVA_KEY_DIR}/cacert.pem
export EUCALYPTUS_CERT=${NOVA_CERT} # euca-bundle-image seems to require this set
alias ec2-bundle-image="ec2-bundle-image --cert ${EC2_CERT} --privatekey ${EC2_PRIVATE_KEY} --
user 42 --ec2cert ${NOVA_CERT}"
alias ec2-upload-bundle="ec2-upload-bundle -a ${EC2_ACCESS_KEY} -s ${EC2_SECRET_KEY} --url
${S3_URL} --ec2cert ${NOVA_CERT}"
export CLOUD_SERVERS_API_KEY="cfc59c93-47a3-4a91-9222-e26721ddf49e"
export CLOUD_SERVERS_USERNAME="root"
export CLOUD_SERVERS_URL="http://172.20.231.73:8774/v1.0/"
```

Append the contents of this file to your profile file (eg: ~/.bashrc) and source it for this session.

```
cat /root/creds/novarc >> ~/.bashrc
source ~/.bashrc
```

You will also find some .pem files in the /root/creds/ directory. These .pem files have to be copied to the $NOVA_KEY_DIR path. (You will see these .pem files being referenced in the novarc file at that path).

Stop any running instances of dnsmasq

```
#ps -eaf | grep dns
nobody   24784     1  0 Apr17 ?        00:00:00 dnsmasq --strict-order --bind-interfaces --conf-
file= --domain=novalocal --pid-file=/var/lib/nova/networks/nova-br100.pid --listen-
address=10.0.0.1 --except-interface=lo --dhcp-range=10.0.0.11,static,120s --dhcp-
hostsfile=/var/lib/nova/networks/nova-br100.conf --dhcp-script=/usr/bin/nova-dhcpbridge --
leasefile-ro
root     24785 24784  0 Apr17 ?        00:00:00 dnsmasq --strict-order --bind-interfaces --conf-
file= --domain=novalocal --pid-file=/var/lib/nova/networks/nova-br100.pid --listen-
address=10.0.0.1 --except-interface=lo --dhcp-range=10.0.0.11,static,120s --dhcp-
hostsfile=/var/lib/nova/networks/nova-br100.conf --dhcp-script=/usr/bin/nova-dhcpbridge --
leasefile-ro
#kill -9 24784 24785
```

Restart all the services.

```
libvirtd restart; restart nova-network; restart nova-compute; restart nova-api; restart nova-
objectstore; restart nova-scheduler
```

Verify all the services registration and status at mysqld by issuing

```
[root@c3l-openstack4 ram]# mysql -uroot -pnova nova -e 'select * from services;'
+---------------------+---------------------+------------+---------+----+----------------+-------
---------+-----------+--------------+----------+-------------------+
| created_at          | updated_at          | deleted_at | deleted | id | host           | binary
| topic     | report_count | disabled | availability_zone |
+---------------------+---------------------+------------+---------+----+----------------+-------
---------+-----------+--------------+----------+-------------------+
| 2011-03-09 20:30:50 | 2011-03-10 19:50:53 | NULL       |       0 |  1 | c3l-openstack4 | nova-
network   | network   |         8370 |        0 |                   |
| 2011-03-09 20:31:16 | 2011-03-10 19:50:59 | NULL       |       0 |  2 | c3l-openstack4 | nova-
compute   | compute   |         8366 |        0 |                   |
| 2011-03-09 20:32:12 | 2011-03-10 19:50:53 | NULL       |       0 |  3 | c3l-openstack4 | nova-
scheduler | scheduler |         8375 |        0 |                   |
+---------------------+---------------------+------------+---------+----+----------------+-------
---------+-----------+--------------+----------+-------------------+
```

Use the ‘euca-authorize’ command to enable ping and ssh access to all the VMs.

```
euca-authorize -P icmp -t -1:-1 default
euca-authorize -P tcp -p 22 default
```

Testing the Installation by Publishing and Starting an Image

Once you have an installation, you want to get images that you can use in your Compute cloud. Download a sample image, and then use the following steps to publish:

```
[root@c3l-openstack4 creds]# wget http://c2477062.cdn.cloudfiles.rackspacecloud.com/images.tgz
--2011-03-10 12:15:40--  http://c2477062.cdn.cloudfiles.rackspacecloud.com/images.tgz
Resolving proxy-sjc-2.cisco.com... 128.107.241.170
Connecting to proxy-sjc-2.cisco.com|128.107.241.170|:80... connected.
Proxy request sent, awaiting response... 200 OK
Length: 58520278 (56M) [appication/x-gzip]
Saving to: “images.tgz”

100%[======================================>] 58,520,278  6.35M/s   in 7.9s

2011-03-10 12:15:49 (7.05 MB/s) - “images.tgz” saved [58520278/58520278]

[root@c3l-openstack4 creds]# mkdir -p ../downloadedimages
[root@c3l-openstack4 creds]# mv images.tgz ../downloadedimages/
[root@c3l-openstack4 creds]# cd ../downloadedimages/
[root@c3l-openstack4 downloadedimages]# ls
images.tgz
[root@c3l-openstack4 downloadedimages]# tar xzvf images.tgz
images/
images/aki-lucid/
images/ami-tiny/
images/ari-lucid/
images/ari-lucid/image
images/ari-lucid/info.json
images/ami-tiny/image
images/ami-tiny/info.json
images/aki-lucid/image
images/aki-lucid/info.json
```

The next set of commands show examples of bundling the images and registering with OpenStack compute:

```
root@c3l-openstack4 downloadedimages]# euca-bundle-image -i images/aki-lucid/image -p kernel --
kernel true
Checking image
Tarring image
Encrypting image
Splitting image...
Part: kernel.part.0
Generating manifest /tmp/kernel.manifest.xml
[root@c3l-openstack4 downloadedimages]#

------------------------------

root@c3l-openstack4 downloadedimages]# euca-bundle-image -i images/ari-lucid/image -p ramdisk --
ramdisk true
Checking image
Tarring image
Encrypting image
Splitting image...
Part: ramdisk.part.0
Generating manifest /tmp/ramdisk.manifest.xml
[root@c3l-openstack4 downloadedimages]#
```

After bundling the images, use *euca-upload-bundle* for uploading and registering the image bundles:

```
[root@c3l-openstack4 downloadedimages]# euca-upload-bundle -m /tmp/kernel.manifest.xml -b
mybucket
Checking bucket: mybucket
Creating bucket: mybucket
Uploading manifest file
Uploading part: kernel.part.0
Uploaded image as mybucket/kernel.manifest.xml
[root@c3l-openstack4 downloadedimages]# euca-upload-bundle -m /tmp/ramdisk.manifest.xml -b
mybucket
Checking bucket: mybucket
Uploading manifest file
Uploading part: ramdisk.part.0
Uploaded image as mybucket/ramdisk.manifest.xml
[root@c3l-openstack4 downloadedimages]#
------------------------------
[root@c3l-openstack4 downloadedimages]# euca-register mybucket/kernel.manifest.xml
IMAGE   ami-la6ojwyz
[root@c3l-openstack4 downloadedimages]# euca-register mybucket/ramdisk.manifest.xml
IMAGE   ami-fgx2gb9m
--------------
root@c3l-openstack4 downloadedimages]# euca-bundle-image -i images/ami-tiny/image -p machine  --
kernel ami-la6ojwyz --ramdisk ami-fgx2gb9m
Checking image
Tarring image
Encrypting image
Splitting image...
Part: machine.part.0
Part: machine.part.1
Part: machine.part.2
Part: machine.part.3
Part: machine.part.4
Generating manifest /tmp/machine.manifest.xml
[root@c3l-openstack4 downloadedimages]#
---------------------------
root@c3l-openstack4 downloadedimages]# euca-upload-bundle -m /tmp/machine.manifest.xml -b
mybucket
Checking bucket: mybucket
Uploading manifest file
Uploading part: machine.part.0
Uploading part: machine.part.1
Uploading part: machine.part.2
Uploading part: machine.part.3
Uploading part: machine.part.4
Uploaded image as mybucket/machine.manifest.xml
```

Now register the image bundle with OpenStack:

```
[root@c3l-openstack4 downloadedimages]# euca-register mybucket/machine.manifest.xml
IMAGE   ami-xra6y0wp
```

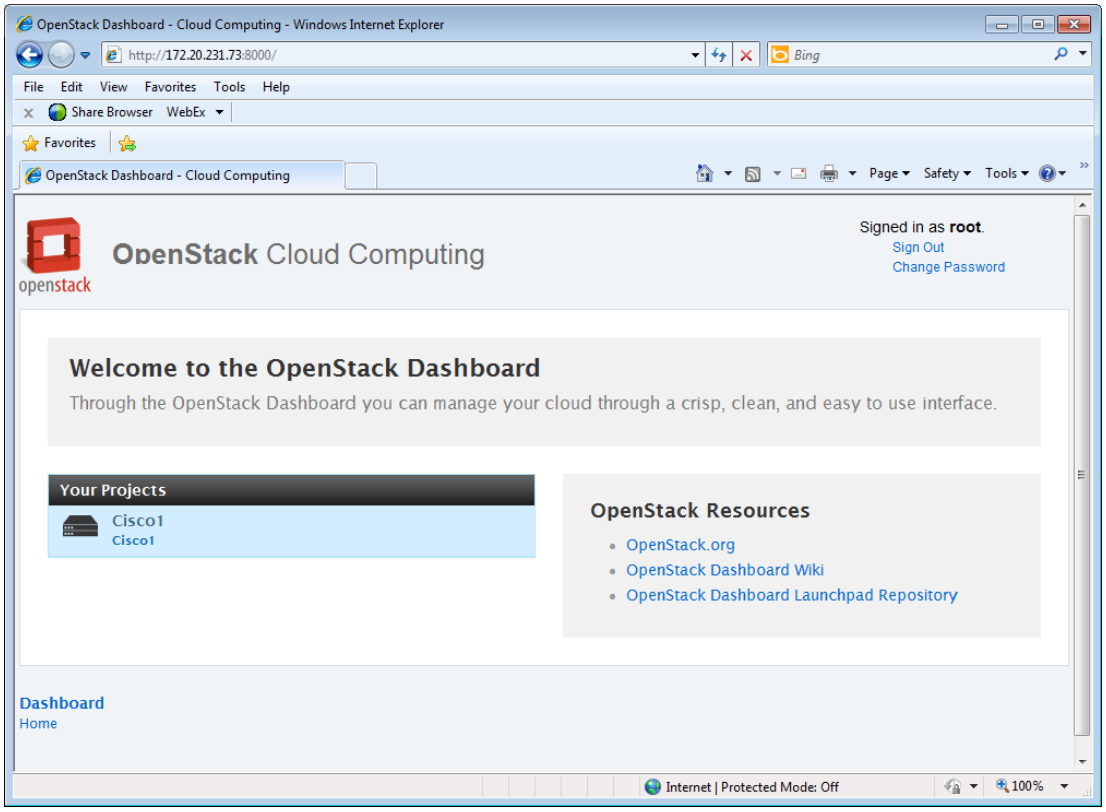
Now you can schedule, launch and connect to the instance, which you do with tools from the Euca2ools on the command line or a Openstack Dashboard application;

OpenStack Dashboard

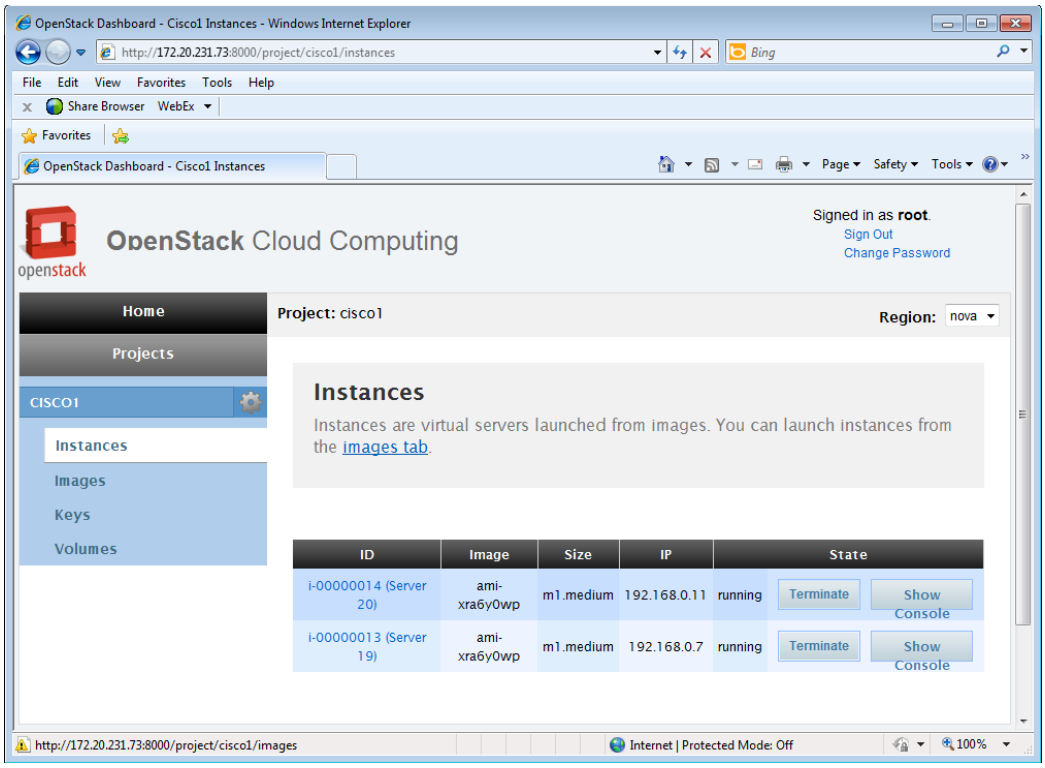
OpenStack Dashboard provides a *django* based GUI for managing OpenStack Compute VM resources

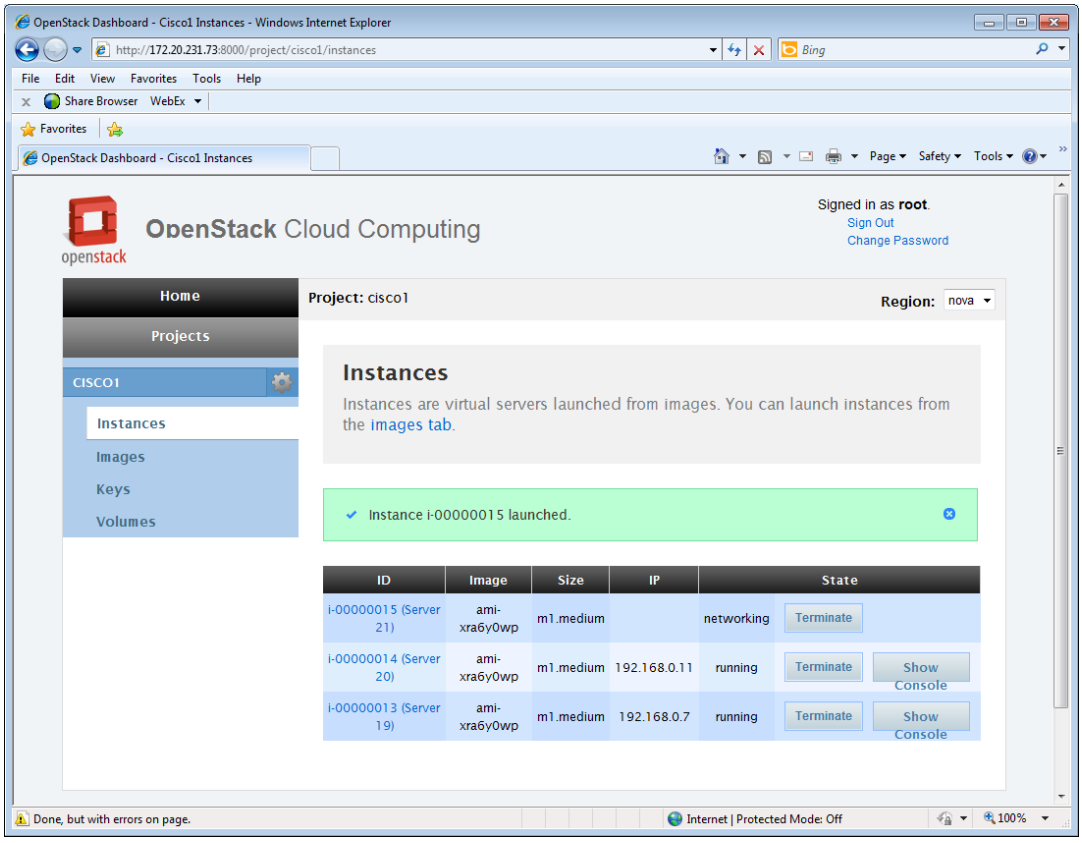
For more information on installing and administering using Openstack Dashboard, please refer to http://wiki.openstack.org/OpenStackDashboard

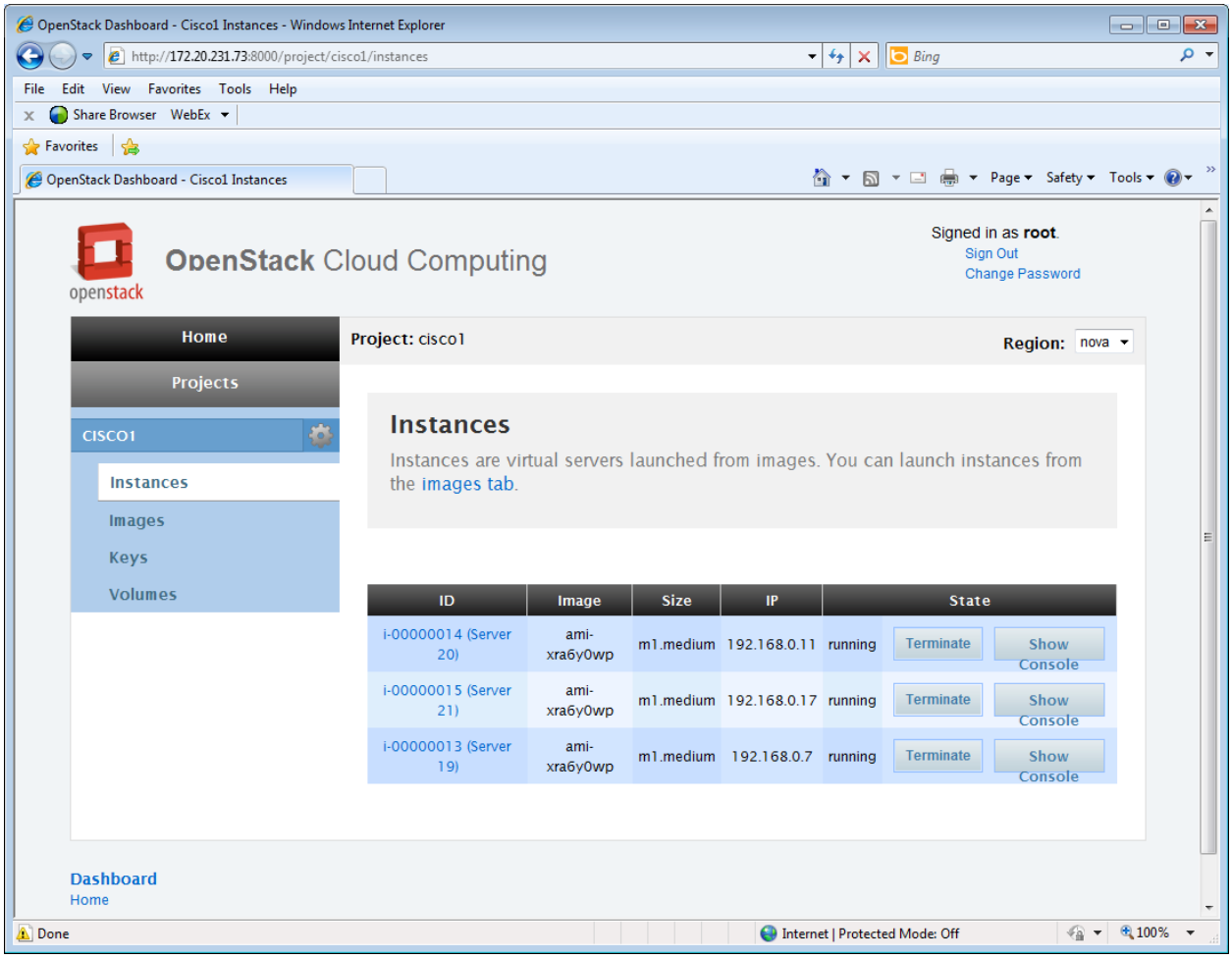
The below screen captures show aspects of the OpenStack Dashboard while launching a VM:



Click on project Cisco1 to list all the available resources. The following screen captures show the current VM instances for project Cisco1.







To launch a new VM, select images option on the left screen and select launch.

Depending on the image that you're using, you need a public key to connect to it. Some images have built-in accounts already created. Images can be shared by many users, so it is dangerous to put passwords into the images. Nova therefore supports injecting ssh keys into instances before they are booted. This allows a user to login to the instances that he or she creates securely. Generally the first thing that a user does when using the system is to create a key pair. Key pairs provide secure authentication to your instances. As part of the first boot of a virtual image, the private key of your key pair is added to root's authorized_keys file. Nova generates a public and private key pair, and sends the private key to the user. The public key is stored so that it can be injected into instances.

Key pairs are created through the api and you use them as a parameter when launching an instance. They can be created on the command line using the euca2ools script *euca-add-keypair*. Refer to the main page for the available options. Example usage:

```
euca-add-keypair test > test.pem
chmod 600 test.pem
```

Installing Compute Nodes

Once the Cloud Controller is installed successfully you can add more compute nodes to the cluster. (Note that if you have followed the instructions above for the installation, you have already installed one compute node on the Cloud Controller itself.)

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices.**