

# OpenStack Compute

## Starter Guide

cactus (Sep 27, 2011)



## OpenStack Compute Starter Guide

cactus (2011-09-27)

Copyright © 2011 CSS Corp Private Limited ( <http://www.csscorp.com/> ) Some rights reserved.

This is a tutorial style beginner's guide for OpenStack™ on Ubuntu 11.04, Natty Narwhal. The aim is to help the reader in setting up a minimal installation of OpenStack.



Except where otherwise noted, this document is licensed under  
**Creative Commons Attribution NonCommercial ShareAlike 3.0 License.**  
<http://creativecommons.org/licenses/by-nc-sa/3.0/legalcode>

## List of Tables

2.1. Configuration .....	7
7.1. Role representation .....	40

# 1. Introduction to OpenStack and Its Components

## Table of Contents

Cloud Computing .....	1
OpenStack .....	1
Open Stack Compute Infrastructure ( Nova ) .....	2
OpenStack Storage Infrastructure ( Swift ) .....	4
OpenStack Imaging Service ( Glance ) .....	5

## Cloud Computing

Cloud computing is a computing model, where resources such as computing power, storage, network and software are abstracted and provided as services on the Internet in a remotely accessible fashion. Billing models for these services are generally similar to the ones adopted for public utilities. On-demand availability, ease of provisioning, dynamic and virtually infinite scalability are some of the key attributes of cloud computing.

An infrastructure setup using the cloud computing model is generally referred to as the "cloud". The following are the broad categories of services available on the cloud:

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

Amazon Web Services (AWS) is one of the major players providing IaaS. They have two popular services - Elastic Compute Cloud (EC2) and Simple Storage Service (S3). These services are available through web services.

## OpenStack

OpenStack is a collection of open source software projects that enterprises/service providers can use to setup and run their cloud compute and storage infrastructure. Rackspace and NASA are the key initial contributors to the stack. Rackspace contributed their "Cloud Files" platform to power the Object Storage part of the OpenStack, while NASA contributed their "Nebula" code to power the Compute part. OpenStack consortium has managed to have more than 100 members including Canonical, Dell, Citrix etc. in less than a year.

OpenStack makes its services available through Amazon EC2/S3 compatible APIs and hence the client tools written for AWS can be used with OpenStack as well.

---

There are 3 main service families under OpenStack

- Compute Infrastructure (Nova)
- Storage Infrastructure (Swift)
- Imaging Service (Glance)

## Open Stack Compute Infrastructure ( Nova )

Nova is the underlying cloud computing fabric controller for the OpenStack cloud. All activities needed to support the life cycle of instances within the OpenStack cloud are handled by Nova. It manages all the compute resources, networking, authorization, and scalability needs of the OpenStack cloud. Nova is a management platform and does not provide any virtualization capabilities by itself; instead, it uses libvirt APIs to interact with the supported hypervisors. Nova exposes its capabilities through a web services API that is compatible with that of EC2 of Amazon Web Services.

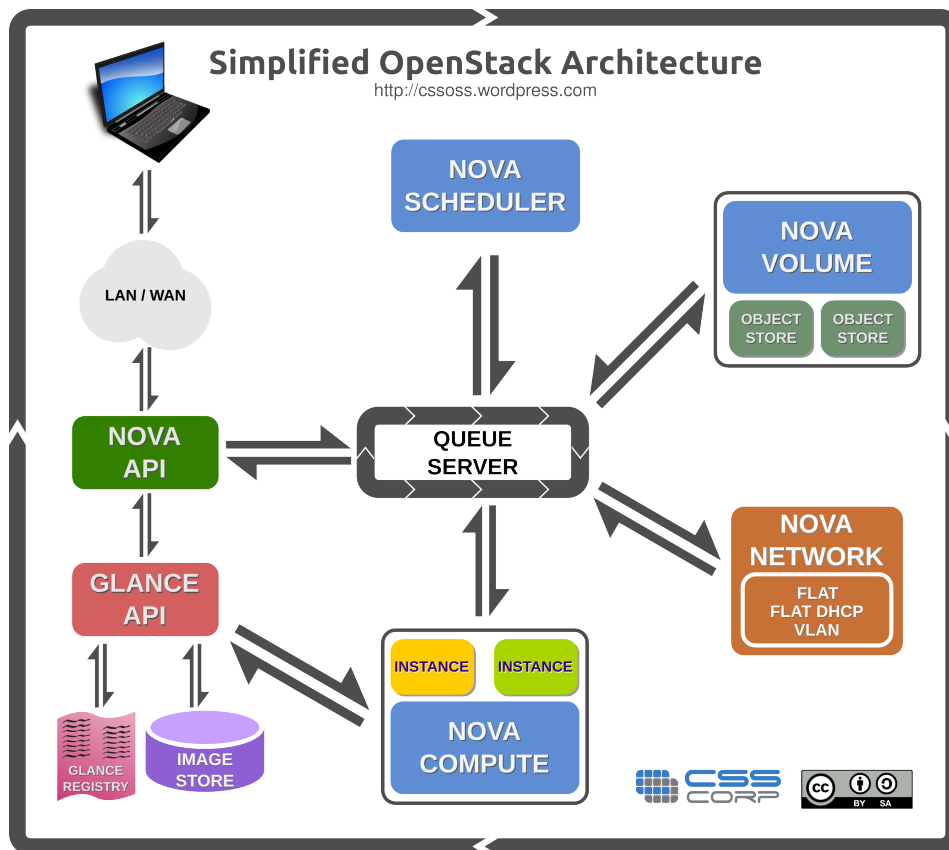
### Functions and Features:

- Instance life cycle management
- Management of compute resources
- Networking and Authorization
- REST-based API
- Asynchronous eventually consistent communication
- Hypervisor agnostic : support for Xen, XenServer/XCP, KVM, UML, VMware vSphere and Hyper-V

### Components of OpenStack Compute

Nova Cloud Fabric is composed of the following major components:

- API Server ( nova-api )
- Message Queue ( rabbit-mq server )
- Compute Workers ( nova-compute )
- Network Controller ( nova-network )
- Volume Worker ( nova-volume )
- Scheduler ( nova-scheduler )



### API Server ( nova-api )

The API Server provides an interface to the outside world to interact with the cloud infrastructure. API server is the only component that the outside world uses to manage the infrastructure. The management is done through web services calls using EC2 API. The API Server then, in turn, communicates with the relevant components of the cloud infrastructure through the Message Queue. Also there is an OpenStack API as well now, as an alternative to the EC2 API.

### Message Queue ( Rabbit MQ Server )

The OpenStack Cloud Controller communicates with other nova components such as the Scheduler, Network Controller, and Volume Controller via AMQP(Advanced Message Queue Protocol). Nova uses asynchronous calls for request response, with a call-back that gets triggered once a response is received. Since asynchronous communication is used, none of the end points get stuck for long in a waiting state. This is especially true since many actions expected by the API calls such as launching an instance or uploading an image are time consuming.

### Compute Worker ( nova-compute )

Compute workers deal with instance management life cycle. they receive the requests for life cycle management via the Message Queue and carry out operations. There are several Compute Workers in a typical production cloud deployment. An instance is deployed on any of the available compute worker based on the scheduling algorithm used.

## Network Controller ( nova-network )

The Network Controller deals with the network configuration of host machines. It does operations like allocating IP addresses, configuring VLANs for projects, implementing security groups and configuring networks for compute nodes.

## Volume Workers ( nova-volume )

Volume workers are used for the management of LVM-based instance volumes. Volume Workers perform volume related functions such as creation, deletion, attaching a volume to an instance, and detaching a volume from an instance. Volumes provide a way of providing persistent storage for use by instances, as the main disk attached to an instance is non-persistent and any changes made to it are lost when the volume is detached or the instance is terminated. When a volume is detached from an instance or when an instance, to which the volume is attached, is terminated, it retains the data that was stored on it when it was attached to an instance earlier. This data can be accessed by reattaching the volume to the same instance or by attaching it to another instances.

Any valuable data that gets accumulated during the life cycle of an instance should be written to a volume, so that it can be accessed later. This typically applies to the storage needs of database servers etc.

## Scheduler ( nova-scheduler )

The scheduler maps the nova-API calls to the appropriate openstack components. It runs as a daemon named nova-schedule and picks up a compute/network/volume server from a pool of available resources depending upon the scheduling algorithm in place. A scheduler can base its decisions on various factors such as load, memory, physical distance of the availability zone, CPU architecture, etc. The nova scheduler implements a pluggable architecture.

Currently the nova-scheduler implements a few basic scheduling algorithms:

- chance: In this method, a compute host is chosen randomly across availability zones.
- availability zone: Similar to chance, but the compute host is chosen randomly from within a specified availability zone.
- simple: In this method, hosts whose load is least are chosen to run the instance. The load information may be fetched from a load balancer.

## OpenStack Storage Infrastructure ( Swift )

Swift is an object store that stores a large number of objects distributed across commodity hardware. Swift has built-in redundancy and failover management and features like backing up or archiving data, serving graphics or videos. It is scalable to multiple petabytes and to billions of objects. Swift provides elasticity and flexibility of cloud-based storage for your web applications.

## Functions and Features

- Storing the machine images

- Working as an independent data container
- Redundancy and failover
- Backup and archival
- Extremely scalable

## OpenStack Imaging Service ( Glance )

OpenStack Imaging Service is a lookup and retrieval system for virtual machine images. It can be configured to use any one of the following 3 storage backends:

- OpenStack Object Store to store images
- S3 storage directly
- S3 storage with Object Store as the intermediate for S3 access.

## Functions and Features ( Glance )

- Provides imaging service

## Components of OpenStack Imaging Service ( Glance )

- Glance-control
- Glance-registry



## 2. Installation and Configuration

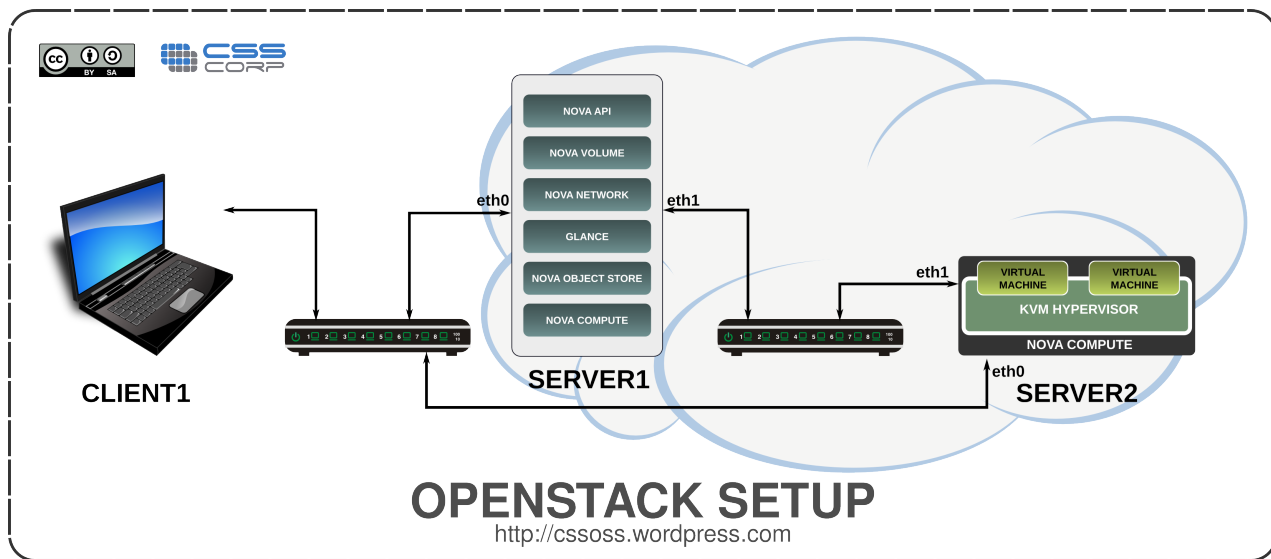
### Table of Contents

Introduction .....	6
Server1 .....	7
Base OS .....	7
Networking Configuration .....	8
NTP Server .....	9
Glance .....	9
MySQL Server .....	9
Nova Components .....	10
Nova dashboard .....	12
Server 2 .....	16
BaseOS .....	16
Networking Configuration .....	16
NTP Client .....	16
Nova Components (nova-compute alone) .....	17
Client1 .....	17
BaseOS .....	17
Networking Configuration .....	17
NTP Client .....	18
Client Tools .....	18

### Introduction

The following section describes how to set up a minimal cloud infrastructure based on OpenStack using 3 machines. These machines are referred to in this and subsequent chapters as Server1 and Server2 and Client1. Server1 runs all the components of Nova as well as Glance and OpenStack dashboard. Server2 runs only nova-compute. Since OpenStack components follow a shared-nothing policy, each component or any group of components can be installed on any server.

Client1 is not a required component. In our sample setup, it is used for bundling images, as a client to the web interface, and to run euca commands to manage the infrastructure. Having this client ensures that you do not need to meddle with the servers for tasks such as bundling. Also, bundling of Desktop Systems including Windows will require a GUI and it is better to have a dedicated machine for this purpose. We would recommend this machine to be VT-Enabled so that KVM can be run which allows for Windows VMs during image creation for bundling.



The installation steps use certain specifics such as host names/IP addresses etc. Modify them to suit your environment before using them. The following table summarizes these specifics.

**Table 2.1. Configuration**

	Server 1	Server 2	Client 1
Functionality	All components of OpenStack including nova-compute	nova-compute	Client
No of NICs	eth0 - Public N/W, eth1 - Private N/W	eth0 - Public N/W, eth1 - Private N/W	eth0 - Public N/W
IP addresses	eth0 - 10.10.10.2, eth1 - 192.168.3.1	eth0 - 10.10.10.3, eth1 - 192.168.3.2	eth0 - 10.10.10.4
Hostname	server1.example.com	server2.example.com	client.example.com
DNS servers	10.10.10.3	10.10.10.3	10.10.10.3
Gateway IP	10.10.10.1	10.10.10.1	10.10.10.1

## Server1

As shown in the figure above, Server1 contains all nova- services including nova-compute, nova-api, nova-volume, nova-network, nova-objectstore as well as the Image Service Glance and the Dashboard. It contains two Network Interface Cards (NICs).

## Base OS

Boot the server off the Ubuntu server 11.04 CD. At the graphical menu select Install Ubuntu server and proceed with basic installation steps.

We will also be running nova-volume on this server and it is ideal to have a dedicated partition for the use of nova-volume. So, ensure that you choose manual partitioning scheme while installing Ubuntu Server and create a dedicated partition with adequate amount of space for this purpose. We have referred to this partition in the rest of the

chapter as `/dev/sda6`. You can substitute the correct device name of this dedicated partition based on your local setup while following the instructions. Also ensure that the partition type is set as Linux LVM ( 8e ) using `fdisk` either during install or immediately after installation is over.

- Create the first user with the name 'localadmin' .
- Installation lets you setup the IP address for the first interface i.e. `eth0`. Set the IP address details.
- During installation select only `Openssh-server` in the packages menu.

Nova and Glance have been included in Universe repository . Enable Universe repository in your `/etc/apt/sources.list`.

Update the machine using the following commands.

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

Install `bridge-utils`:

```
sudo apt-get install bridge-utils
```

Reboot the server and login as the admin user(`localadmin`) created during the OS installation.

## Networking Configuration

Edit the `/etc/network/interfaces` file so as to looks like this:

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
address 10.10.10.2
netmask 255.255.255.0
broadcast 10.10.10.255
gateway 10.10.10.1
dns-nameservers 10.10.10.3

auto br100
iface br100 inet static
bridge_ports eth1
bridge_stp off
bridge_maxwait 0
bridge_fd 0
address 192.168.3.1
netmask 255.255.0.0
broadcast 192.168.255.255
```

Restart the network now

```
sudo /etc/init.d/networking restart
```

## NTP Server

Install NTP package. This server is going to act as an NTP server for the nodes. The time on all components of OpenStack will have to be in sync. We can run NTP server on this and have other components sync to it.

```
sudo apt-get install ntp
```

Open the file `/etc/ntp.conf` and add the following 2 lines to make sure that the server serves time even when its connectivity to the Internet is down. The following settings ensure that the NTP server uses its own clock as the clock source:

```
server 127.127.1.0
fudge 127.127.1.0 stratum 10
```

Restart NTP service to make the changes effective

```
sudo /etc/init.d/ntp restart
```

## Glance

Glance is an image Server that Nova can use to pickup images from. Glance is very modular and can use several types of storage backends such as filestore, s3 etc. We are installing Glance before installing Nova, so that when we get to configuring Nova, glance is ready to be used by Nova.

```
sudo apt-get install glance
```

The default config file at `/etc/glance/glance.conf` is good to use for a simple file store as the storage backend. Glance can be configured to use other storage backends such as Swift.

Glance uses sqlite as the default database backend. While sqlite offers a quick and easy way to get started, for production use, you may consider a database such as MySQL or PostgreSQL.

Glance has two components - `glance-api` and `glance-registry`. These can be controlled using the concerned upstart jobs.

## MySQL Server

Install `mysql-server` package

```
sudo apt-get install -y mysql-server
```

## Configuration

Set a variable called `"MYSQL_PASS"` for use in the various commands below:

```
MYSQL_PASS="mygreatsecret"
```

Change the bind address from `127.0.0.1` to `0.0.0.0` in `/etc/mysql/my.cnf` and it will look like this:

```
bind-address = 0.0.0.0
```

Restart MySQL server to ensure that it starts listening on all interfaces.

```
sudo restart mysql
```

If you did not set the MySQL root password during installation, set it now.

```
mysqladmin -u root password $MYSQL_PASS
```

Create a database named nova.

```
sudo mysql -uroot -p$MYSQL_PASS -e 'CREATE DATABASE nova;'
```

Update the database to grant super user privileges for root user to login from any IP.

```
sudo mysql -uroot -p$MYSQL_PASS -e "GRANT ALL PRIVILEGES ON *.* TO 'root'@'%'  
WITH GRANT OPTION;"
```

Set MySQL root password for login from any IP.

```
sudo mysql -uroot -p$MYSQL_PASS -e "SET PASSWORD FOR 'root'@'%'  
= PASSWORD('$MYSQL_PASS');"
```

## Nova Components

Install the messaging queue server, RabbitMQ and various nova components.

```
sudo apt-get install -y rabbitmq-server nova-common nova-doc python-nova  
python-psycopq2 nova-api nova-network nova-volume nova-objectstore nova-  
scheduler nova-compute
```

Install euca2ools package for command line tools to interact with nova.

```
sudo apt-get install -y euca2ools
```

Install unzip for extracting archives.

```
sudo apt-get install -y unzip
```

Edit the `/etc/nova/nova.conf` file to look like this.

```
--dhcpbridge_flagfile=/etc/nova/nova.conf  
--dhcpbridge=/usr/bin/nova-dhcpbridge  
--logdir=/var/log/nova  
--lock_path=/var/lock/nova  
--state_path=/var/lib/nova  
--verbose  
--s3_host=10.10.10.2  
--rabbit_host=192.168.3.1  
--cc_host=192.168.3.1  
--ec2_url=http://10.10.10.2:8773/services/Cloud  
--fixed_range=192.168.0.0/16  
--network_size=8  
--FAKE_subdomain=ec2  
--routing_source_ip=192.168.3.1  
--sql_connection=mysql://root:mygreatsecret@10.10.10.2/nova  
--glance_host=192.168.3.1  
--image_service=nova.image.glance.GlanceImageService
```

```
--iscsi_ip_prefix=192.168.
```

Enable iscsitarget.

```
sudo sed -i 's/false/true/g' /etc/default/iscsitarget
```

Restart the iscsitarget service

```
sudo service iscsitarget restart
```

Create a Physical Volume.

```
sudo pvcreate /dev/sda6
```

Create a Volume Group named nova-volumes.

```
sudo vgcreate nova-volumes /dev/sda6
```

Create a group called "nova".

```
sudo groupadd nova
```

Change the ownership of the /etc/nova folder and permissions for /etc/nova/nova.conf:

```
sudo chown -R root:nova /etc/nova
sudo chmod 644 /etc/nova/nova.conf
```

Restart all the nova related services.

```
sudo restart libvirt-bin; sudo restart nova-network; sudo restart nova-
compute; sudo restart nova-api; sudo restart nova-objectstore; sudo restart
nova-scheduler; sudo restart nova-volume; sudo restart glance-api; sudo
restart glance-registry
```

Create nova schema in the MySQL Database.

```
sudo nova-manage db sync
```

Create a list of IPs to be used from the network of fixed ips set inside nova.conf.

```
sudo nova-manage network create 192.168.3.0/24 1 255
```

Allocate 32 public IP addresses for use with the instances starting from 10.10.10.225.

```
sudo nova-manage floating create 10.10.10.2 10.10.10.224/27
```

Create a user with admin rights on nova.

```
sudo nova-manage user admin novaadmin
```

Create a project named proj.

```
sudo nova-manage project create proj novaadmin
```

Create a directory to download nova credentials and download the zip file.

```
mkdir /home/localadmin/creds
```

Generate and save credentials for accessing/managing the nova cloud.

```
sudo nova-manage project zipfile proj novaadmin /home/localadmin/creds/  
novacreds.zip
```

Contents of novacreds.zip are required to use euca2ools to manage the cloud infrastructure and you will need to transfer this zip file to any machine from where you want to run the commands from euca2ools. We will be using these credentials from client1 as well.

Navigate in to the folder created and extract the files and change their ownership.

```
cd /home/localadmin/creds  
unzip novacreds.zip  
sudo chown localadmin:localadmin /home/localadmin/creds/ -R
```

Here are the files extracted:

ca.cert.pem, cert.pem, novarc, pk.pem

novarc contains several environmental variables including your nova credentials to be set before you can use the commands from euca2ools such euca-describe-images, euca-describe-instances etc. these variables can be set by sourcing novarc file.

```
source /home/localadmin/creds/novarc
```

Restart all the nova related services.

```
sudo restart libvirt-bin; sudo restart nova-network; sudo restart nova-  
compute; sudo restart nova-api; sudo restart nova-objectstore; sudo restart  
nova-scheduler; sudo restart nova-volume; sudo restart glance-api; sudo  
restart glance-registry
```

Check if the credentials are working and if nova has been setup properly by running:

```
euca-describe-availability-zones verbose
```

If you see something like the following with all components happy, it means that the set up is ready to be used.

```
AVAILABILITYZONE nova available  
AVAILABILITYZONE |- server1  
AVAILABILITYZONE | |- nova-compute enabled : -) 2011-04-03 07:48:50  
AVAILABILITYZONE | |- nova-scheduler enabled : -) 2011-04-03 07:48:48  
AVAILABILITYZONE | |- nova-network enabled : -) 2011-04-03 07:48:49  
AVAILABILITYZONE | |- nova-volume enabled : -) 2011-04-03 07:48:49
```

## Nova dashboard

OpenStack-dashboard is a web interface for managing users, user credentials, key pairs, images, instances etc.

```
sudo easy_install virtualenv
```

You have already finished setting up credentials for a user called localadmin in the Nova configuration section above. The credentials of this user will need to be embedded into the dashboard's configuration file.

Checkout the source of OpenStack-dashboard from github and execute `run_tests.sh`, which does not only test the installation, but also installs several dependencies of the dashboard.

```
cd /opt
sudo wget https://github.com/4P/horizon/zipball/2011.2
sudo unzip 2011.2
sudo mv 4P-horizon-03dce19 osdb
cd osdb
sudo sh run_tests.sh
cd openstack-dashboard
```

Edit `/opt/osdb/openstack-dashboard/local/local_settings.py` to include certain details required for connecting to nova-api.

```
NOVA_DEFAULT_ENDPOINT = 'http://localhost:8773/services/Cloud'
NOVA_DEFAULT_REGION = 'nova'
NOVA_ACCESS_KEY = 'b6a7e3ca-f894-473b-abca-84329d9829fa:proj'
NOVA_SECRET_KEY = '2d61a361-965a-4ed6-966a-d9f543b42531'
NOVA_ADMIN_USER = 'novaadmin'
NOVA_PROJECT = 'proj'
```

A simple way of doing this will be to copy the relevant lines from `novarc` file that we discussed above.

## Setting Up E-mail service for the web interface

In order to have mails generated by OpenStack dashboard delivered, we need to configure dashboard with the details of an smtp server by editing `local_settings.py` file.

```
EMAIL_HOST = 'server1.example.com'
EMAIL_PORT = 25
```

If the mail server provides only authenticated SMTP, add the following lines:

```
EMAIL_USER =
EMAIL_PASSWORD =
```

If the mail server requires a TLS connection, add the following lines:

```
EMAIL_USE_TLS = 'True'
```

Create a `openstack-dashboard` database and its schema with the `syncdb` command. Provide the name/email address/desired password of the administrative user when prompted.

```
sudo tools/with_venv.sh dashboard/manage.py syncdb
```

While creating the schema, the above command asks you to create an admin account for the dashboard. Choose the user name as the project admin's user name you chose above while creating the project ( `novadmin` in our case). You can choose any password you like.

Launch the default python-django server. If you want the dashboard application to be available on port 8000:

```
sudo tools/with_venv.sh dashboard/manage.py runserver 10.10.10.2:8000
```



To check the installation open a browser and enter the following URL

```
http://10.10.10.2:8000
```

You should be able to login as "novaadmin" using the password chosen above. Any other user trying to access the interface for the first time, will need to sign up and will be able to use the interface after the account is approved by the administrator.

A successful login and display of the project named "proj" on the dashboard will indicate that the dashboard has been setup successfully

## OpenStack Dashboard with Mysql Database

Dashboard uses SQLite database by default. For a production use, MySQL or PostgreSQL may be more preferable. The procedure for MySQL is given below. Procedure for PostgreSQL will be very similar.

Install python-dev and libmysqlclient-dev

```
sudo apt-get install libmysqlclient-dev
sudo apt-get install python-dev
```

Activate virtualenv and install mysql-python package inside the virtual environment of Dashboard.

```
cd /opt/osdb/openstack-dashboard
sudo bash
source .dashboard-venv/bin/activate
easy_install mysql-python
```

Create a MySQL database user with all privileges on OpenStack Dashboard database

```
mysql -uroot -pmygreatsecret
>create database dashboarddb;
>grant ALL on dashboarddb.* to nova@localhost identified by 'mygreatsecret';
```

Update the DATABASES section of the Django's local\_settings.py file (/opt/osdb/openstack-dashboard/local/local\_settings.py) with the MySQL database settings. Here is the relevant extract from the updated file:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'dashboarddb',
        'USER': 'nova',
        'PASSWORD': 'mygreatsecret',
        'HOST': 'localhost',
        'default-character-set': 'utf8',
    }
}
```

Create the schema in the database

```
sudo tools/with_venv.sh dashboard/manage.py syncdb
```

## Running Dashboard on apache2 with mod\_wsgi

While the web server that is included in Django is good for testing, for production use, it is recommended to use a web server like Apache with mod\_wsgi.

Install apache2 and wsgi module.

```
sudo apt-get install apache2 libapache2-mod-wsgi
```

Dashboard includes a file `django.wsgi(/opt/osdb/openstack-dashboard/dashboard/wsgi/django.wsgi)` to help in running dashboard under Apache with WSGI. You can replace the default file with the file below.

```
# Ref: http://jmoiron.net/blog/deploying-django-mod-wsgi-virtualenv/

import sys
import site
import os

#we are adding virtual environment path.
vepath = '/opt/osdb/openstack-dashboard/.dashboard-venv/lib/python2.7/site-packages'
os.environ['PYTHON_EGG_CACHE'] = '/opt/osdb/openstack-dashboard/.dashboard-venv/lib/python2.7/site-packages'

prev_sys_path = list(sys.path)

# add the site-packages of our virtualenv as a site dir
site.addsitedir(vepath)

# reorder sys.path so new directories from the addsitedir show up first
new_sys_path = [p for p in sys.path if p not in prev_sys_path]

for item in new_sys_path:
    sys.path.remove(item)
sys.path[:0] = new_sys_path

# import from down here to pull in possible virtualenv django install

from django.core.handlers.wsgi import WSGIHandler
os.environ['DJANGO_SETTINGS_MODULE'] = 'dashboard.settings'
application = WSGIHandler()
```

## Setting up the virtual host and WSGI alias in Apache

Create `/etc/apache2/sites-available/openstack` with the following contents:

```
Listen 8000
<VirtualHost 10.10.10.2:8000>
    ServerName 10.10.10.2
    WSGIScriptAlias / /opt/osdb/openstack-dashboard/dashboard/wsgi/django.wsgi
    Alias /media/admin/ /opt/osdb/openstack-dashboard/.dashboard-venv/lib/python2.7/site-packages/django/contrib/admin/media/
</VirtualHost>
```

Enable virtual host.

```
sudo a2ensite openstack
sudo /etc/init.d/apache2 reload
```

Dashboard should now be available at <http://10.10.10.2:8000>

## Server 2

This server runs nova-compute and all the virtual machines and hypervisor. You can also bundle images on Server 2.

## BaseOS

Install 64-bit version of Natty Server

## Networking Configuration

Install bridge-utils:

```
sudo apt-get install bridge-utils
```

Edit the `/etc/network/interfaces` file so as to look like this:

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
address 10.10.10.3
netmask 255.255.255.0
broadcast 10.10.10.255
gateway 10.10.10.1
dns-nameservers 10.10.10.3

auto br100
iface br100 inet static
bridge_ports eth1
bridge_stp off
bridge_maxwait 0
bridge_fd 0
address 192.168.3.2
netmask 255.255.0.0
broadcast 192.168.255.255
```

Restart the network now

```
sudo /etc/init.d/networking restart
```

## NTP Client

Install NTP package.

```
sudo apt-get install ntp
```

Open the file `/etc/ntp.conf` and add the following line to sync to server1.

```
server 10.10.10.2
```

Restart NTP service to make the changes effective

```
sudo /etc/init.d/ntp restart
```

## Nova Components (nova-compute alone)

Install the nova-components and dependencies.

```
sudo apt-get install -y nova-common python-nova python-psycopg2 nova-compute  
vlan
```

Install euca tools, for command line tools

```
sudo apt-get install -y euca2ools
```

Install unzip for extracting archives

```
sudo apt-get install -y unzip
```

Edit the `/etc/nova/nova.conf` file to look like this. This file is essentially similar to the configuration file (`/etc/nova/nova.conf`) of Server1

```
--dhcpbridge_flagfile=/etc/nova/nova.conf  
--dhcpbridge=/usr/bin/nova-dhcpbridge  
--logdir=/var/log/nova  
--lock_path=/var/lock/nova  
--state_path=/var/lib/nova  
--verbose  
--s3_host=10.10.10.2  
--rabbit_host=192.168.3.1  
--cc_host=192.168.3.1  
--ec2_url=http://10.10.10.2:8773/services/Cloud  
--fixed_range=192.168.0.0/16  
--network_size=8  
--FAKE_subdomain=ec2  
--routing_source_ip=192.168.3.2  
--sql_connection=mysql://root:mygreatsecret@10.10.10.2/nova  
--glance_host=192.168.3.1  
--image_service=nova.image.glance.GlanceImageService
```

## Client1

### BaseOS

Install 64-bit version of Natty Desktop

## Networking Configuration

Edit the `/etc/network/interfaces` file so as to looks like this:

```
auto lo
  iface lo inet loopback
auto eth0
  iface eth0 inet static
  address 10.10.10.4
  netmask 255.255.255.0
  broadcast 10.10.10.255
  gateway 10.10.10.1
  dns-nameservers 10.10.10.3
```

## NTP Client

Install NTP package.

```
sudo apt-get install ntp
```

Open the file `/etc/ntp.conf` and add the following line to sync to server1.

```
server 10.10.10.2
```

Restart NTP service to make the changes effective

```
sudo /etc/init.d/ntp restart
```

## Client Tools

As mentioned above, this is a desktop installation of Natty to be used for tasks such as bundling of images. It will also be used for managing the cloud infrastructure using euca2ools.

Install euca tools, for command line tools

```
sudo apt-get install -y euca2ools
```

Install qemu-kvm

```
sudo apt-get install qemu-kvm
```

Download the credentials we created for localadmin to this machine:

```
mkdir /home/localadmin/creds
cd /home/localadmin/creds
scp localadmin@10.10.10.2:/home/localadmin/creds/novacreds.zip .
unzip creds.zip
```

Source novarc file and see if connectivity to api server is working correctly:

```
source novarc
euca-describe-availability-zones verbose
```

The output should be similar to what is shown above in the configuration section for server1.

Note: If you want to avoid manually sourcing the novarc file every time, the user can add the following line to the .profile file in his home directory:

```
source /home/localadmin/creds/novarc
```

## 3. Image Management

### Table of Contents

Introduction .....	20
Creating a Linux Image - Ubuntu & Fedora .....	21
OS Installation .....	21
Extracting the EXT4 partition .....	22
Tweaking /etc/fstab .....	23
Fetching Metadata in Fedora .....	24
Kernel and Initrd for OpenStack .....	24
Uploading to OpenStack .....	24
Image Listing .....	25
Creating a Windows Image .....	25

### Introduction

There are several pre-built images for OpenStack available from various sources. You can download such images and use them to get familiar with OpenStack. You can refer to <http://docs.openstack.org/cactus/openstack-compute/admin/content/starting-images.html> for details on using such images.

For any production deployment, you may like to have the ability to bundle custom images, with a custom set of applications or configuration. This chapter will guide you through the process of creating Linux images of Debian and RedHat based distributions from scratch. We have also covered an approach to bundling Windows images.

There are some minor differences in the way you would bundle a Linux image, based on the distribution. Ubuntu makes it very easy by providing cloud-init package, which can be used to take care of the instance configuration at the time of launch. cloud-init handles importing ssh keys for password-less login, setting host name etc. The instance acquires the instance specific configuration from Nova-compute by connecting to a meta data interface running on 169.254.169.254.

While creating the image of a distro that does not have cloud-init or an equivalent package, you may need to take care of importing the keys etc. by running a set of commands at boot time from rc.local.

The process used for Ubuntu and Fedora is largely the same with a few minor differences, which are explained below.

In both cases, the documentation below assumes that you have a working KVM installation to use for creating the images. We are using the machine called 'client1' as explained in the chapter on "Installation and Configuration" for this purpose.

The approach explained below will give you disk images that represent a disk without any partitions. Nova-compute can resize such disks ( including resizing the file system) based on the instance type chosen at the time of launching the instance. These images cannot have 'bootable' flag and hence it is mandatory to have associated kernel and ramdisk

images. These kernel and ramdisk images need to be used by nova-compute at the time of launching the instance.

However, we have also added a small section towards the end of the chapter about creating bootable images with multiple partitions that can be used by nova to launch an instance without the need for kernel and ramdisk images. The caveat is that while nova-compute can re-size such disks at the time of launching the instance, the file system size is not altered and hence, for all practical purposes, such disks are not re-sizable.

## Creating a Linux Image - Ubuntu & Fedora

The first step would be to create a raw image on Client1. This will represent the main HDD of the virtual machine, so make sure to give it as much space as you will need.

```
kvm-img create -f raw server.img 5G
```

## OS Installation

Download the iso file of the Linux distribution you want installed in the image. The instructions below are tested on Ubuntu 11.04 Natty Narwhal 64-bit server and Fedora 14 64-bit. Most of the instructions refer to Ubuntu. The points of difference between Ubuntu and Fedora are mentioned wherever required.

```
wget http://releases.ubuntu.com/natty/ubuntu-11.04-server-amd64.iso
```

Boot a KVM instance with the OS installer ISO in the virtual CD-ROM. This will start the installation process. The command below also sets up a VNC display at port 0

```
sudo kvm -m 256 -cdrom ubuntu-11.04-server-amd64.iso -drive file=server.img,  
if=scsi,index=0 -boot d -net nic -net user -nographic -vnc :0
```

Connect to the VM through VNC (use display number :0) and finish the installation.

For Example, where 10.10.10.4 is the IP address of client1:

```
vncviewer 10.10.10.4 :0
```

During the installation of Ubuntu, create a single ext4 partition mounted on '/'. Do not create a swap partition.

In the case of Fedora 14, the installation will not progress unless you create a swap partition. Please go ahead and create a swap partition.

After finishing the installation, relaunch the VM by executing the following command.

```
sudo kvm -m 256 -drive file=server.img,if=scsi,index=0,boot=on -boot c -net  
nic -net user -nographic -vnc :0
```

At this point, you can add all the packages you want to have installed, update the installation, add users and make any configuration changes you want in your image.



At the minimum, for Ubuntu you may run the following commands

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

```
sudo apt-get install openssh-server cloud-init
```

For Fedora run the following commands as root

```
yum update
```

```
yum install openssh-server
```

```
chkconfig sshd on
```

Also remove the network persistence rules from `/etc/udev/rules.d` as their presence will result in the network interface in the instance coming up as an interface other than `eth0`.

```
sudo rm -rf /etc/udev/rules.d/70-persistent-net.rules
```

Shutdown the virtual machine and proceed with the next steps.

## Extracting the EXT4 partition

The image that needs to be uploaded to OpenStack needs to be an ext4 filesystem image. Here are the steps to create a ext4 filesystem image from the raw image i.e `server.img`

```
sudo losetup -f server.img
```

```
sudo losetup -a
```

You should see an output like this:

```
/dev/loop0: [0801]:16908388 ($filepath)
```

Observe the name of the loop device (`/dev/loop0` in our setup) when `$filepath` is the path to the mounted `.raw` file.

Now we need to find out the starting sector of the partition. Run:

```
sudo fdisk -cul /dev/loop0
```

You should see an output like this:

```
Disk /dev/loop0: 5368 MB, 5368709120 bytes
```

```
149 heads, 8 sectors/track, 8796 cylinders, total 10485760 sectors
```

```
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00072bd4
Device           Boot Start      End          Blocks      Id
System
/dev/loop0p1    *          2048      10483711    5240832     83   Linux
```

Make a note of the starting sector of the `/dev/loop0p1` partition i.e the partition whose ID is 83. This number should be multiplied by 512 to obtain the correct value. In this case:  $2048 \times 512 = 1048576$

Unmount the loop0 device:

```
sudo losetup -d /dev/loop0
```

Now mount only the partition(`/dev/loop0p1`) of `server.img` which we had previously noted down, by adding the `-o` parameter with value previously calculated value

```
sudo losetup -f -o 1048576 server.img
```

```
sudo losetup -a
```

You'll see a message like this:

```
/dev/loop0: [0801]:16908388 ($filepath) offset 1048576
```

Make a note of the mount point of our device(`/dev/loop0` in our setup) when `$filepath` is the path to the mounted `.raw` file.

Copy the entire partition to a new `.raw` file

```
sudo dd if=/dev/loop0 of=serverfinal.img
```

Now we have our ext4 filesystem image i.e `serverfinal.img`

Unmount the loop0 device

```
sudo losetup -d /dev/loop0
```

## Tweaking `/etc/fstab`

You will need to tweak `/etc/fstab` to make it suitable for a cloud instance. Nova-compute may resize the disk at the time of launch of instances based on the instance type chosen. This can make the UUID of the disk invalid. Hence we have to use File system label as the identifier for the partition instead of the UUID.

Loop mount the `serverfinal.img`, by running

```
sudo mount -o loop serverfinal.img /mnt
```

```
UUID=e7f5af8d-5d96-45cc-a0fc-d0d1bde8f31c / ext4 errors=remount-ro 0 1
```

to

```
LABEL=uec-rootfs / ext4 defaults 0 0
```

## Fetching Metadata in Fedora

Since, Fedora does not ship with cloud-init or an equivalent, you will need to take a few steps to have the instance fetch the meta data like ssh keys etc.

Edit the `/etc/rc.local` file and add the following lines before the line `"touch /var/lock/subsys/local"`

```
depmod -a
modprobe acpihp
# simple attempt to get the user ssh key using the meta-data service
mkdir -p /root/.ssh
echo >> /root/.ssh/authorized_keys
curl -m 10 -s http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key| grep 'ssh-rsa' >> /root/.ssh/authorized_keys
echo "AUTHORIZED_KEYS:"
echo "*****"
cat /root/.ssh/authorized_keys
echo "*****"
```

## Kernel and Initrd for OpenStack

Copy the kernel and the initrd image from `/mnt/boot` to user home directory. These will be used later for creating and uploading a complete virtual image to OpenStack.

```
sudo cp /mnt/boot/vmlinuz-2.6.38-7-server /home/localadmin
```

```
sudo cp /mnt/boot/initrd.img-2.6.38-7-server /home/localadmin
```

Unmount the Loop partition

```
sudo umount /mnt
```

Change the filesystem label of `serverfinal.img` to 'uec-rootfs'

```
sudo tune2fs -L uec-rootfs serverfinal.img
```

Now, we have all the components of the image ready to be uploaded to OpenStack imaging server.

## Uploading to OpenStack

The last step would be to upload the images to OpenStack Imaging Server glance. The files that need to be uploaded for the above sample setup of Ubuntu are: `vmlinuz-2.6.38-7-server`, `initrd.img-2.6.38-7-server`, `serverfinal.img`

Run the following command

```
uec-publish-image -t image --kernel-file vmlinuz-2.6.38-7-server --ramdisk-  
file initrd.img-2.6.38-7-server amd64 serverfinal.img bucket1
```

For Fedora, the process will be similar. Make sure that you use the right kernel and initrd files extracted above.

uec-publish-image, like several other commands from euca2ools, returns the prompt back immediately. However, the upload process takes some time and the images will be usable only after the process is complete. You can keep checking the status using the command 'euca-describe-images' as mentioned below.

You can upload bootable disk images without associating kernel and ramdisk images. When you do not want the flexibility of using the same disk image with different kernel/ramdisk images, you can go for bootable disk images. This greatly simplifies the process of bundling and uploading the images. However, the caveats mentioned in the introduction to this chapter apply. Please note that the instructions below use server.img and you can skip all the cumbersome steps related to extracting the single ext4 partition.

```
euca-bundle-image -i server.img
```

```
euca-upload-bundle -b mybucket -m /tmp/server.img.manifest.xml
```

```
euca-register mybucket/server.img.manifest.xml
```

## Image Listing

The status of the images that have been uploaded can be viewed by using euca-describe-images command. The output should like this:

```
localadmin@client1:~$ euca-describe-images
```

```
IMAGE   ari-7bfac859   bucket1/initrd.img-2.6.38-7-server.manifest.xml   css  
available private x86_64   ramdisk
```

```
IMAGE   ami-5e17eb9d   bucket1/serverfinal.img.manifest.xml   css   available  
private   x86_64   machine   aki-3d0aeb08   ari-7bfac859
```

```
IMAGE   aki-3d0aeb08   bucket1/vmlinuz-2.6.38-7-server.manifest.xml   css  
available private x86_64   kernel
```

```
localadmin@client1:~$
```

## Creating a Windows Image

The first step would be to create a raw image on Client1, this will represent the main HDD of the virtual machine, so make sure to give it as much space as you will need.

```
kvm-img create -f raw windowsserver.img 20G
```

OpenStack presents the disk using a VIRTIO interface while launching the instance. Hence the OS needs to have drivers for VIRTIO. By default, the Windows Server 2008 ISO does not have the drivers for VIRTIO. Download the virtual floppy drive containing VIRTIO drivers from the following location

<http://alt.fedoraproject.org/pub/alt/virtio-win/latest/images/bin/>

and attach it during the installation

Start the installation by running

```
sudo kvm -m 1024 -cdrom win2k8_dvd.iso -drive file=windowsserver.img,if=
virtio,boot=on -fda virtio-win-1.1.16.vfd -boot d -nographic -vnc :0
```

When the installation prompts you to choose a hard disk device you won't see any devices available. Click on "Load drivers" at the bottom left and load the drivers from A:\i386\Win2008

After the Installation is over, boot into it once and install any additional applications you need to install and make any configuration changes you need to make. Also ensure that RDP is enabled as that would be the only way you can connect to a running instance of Windows. Windows firewall needs to be configured to allow incoming ICMP and RDP connections.

For OpenStack to allow incoming RDP Connections, use euca-authorize command to open up port 3389 as described in the chapter on "Security".

Shut-down the VM and upload the image to OpenStack

```
euca-bundle-image -i windowsserver.img
```

```
euca-upload-bundle -b mybucket -m /tmp/windowsserver.img.manifest.xml
```

```
euca-register mybucket/windowsserver.img.manifest.xml
```

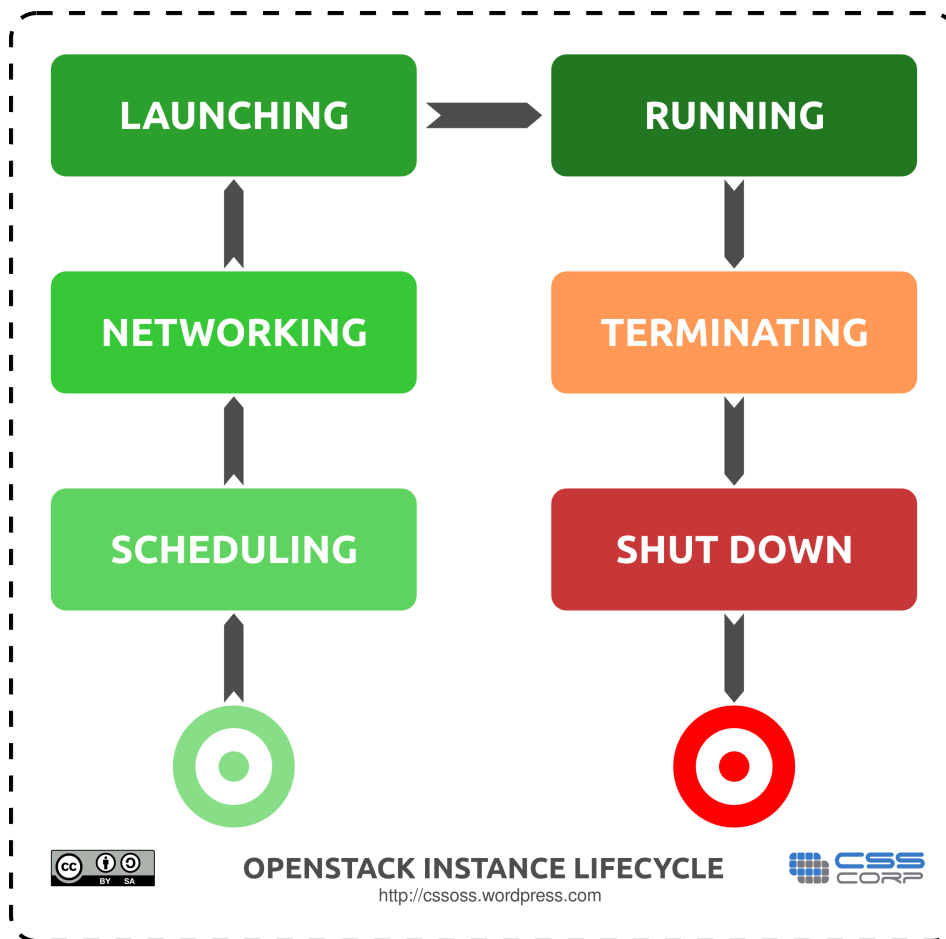
# 4. Instance Management

## Table of Contents

Introduction .....	27
Hybridfox .....	28
Features .....	29
Installation .....	29
Configuration .....	29
Euca2ools-Command Line Tools .....	30
Installation .....	30
Creation of Key Pairs .....	30
Launch and manage instances .....	31
Using the OpenStack Dashboard .....	32
Creating Keypairs .....	32
Launching an instance .....	33
Terminating an instance .....	33
Displaying the Console Output .....	34

## Introduction

An instance is a virtual machine provisioned by OpenStack on one of the nova-compute servers. When you launch an instance, a series of steps are triggered on various components of the OpenStack. During the life cycles of an instance, it moves through various stages as shown in the diagram below:



There are four interfaces that can be used for managing instances in nova.

- Firefox with Hybridfox Plugin
- Command line tools like euca2ools
- OpenStack Dashboard
- Custom applications developed using EC2 APIs

## Hybridfox

ElasticFox is an open source Mozilla Firefox extension that works on Firefox Version 2.0 or later to help you with managing AWS compute and associated resources - Launch new instances, mount Elastic Block Storage volumes, map Elastic IP addresses, and more. This was originally written for EC2, but, it has become possible to use to manage resources from other cloud platforms that are API compatible with EC2 as well, particularly Eucalyptus.

Hybridfox is a fork from ElasticFox to make it usable with Eucalyptus, when ElasticFox worked only with AWS. The aim of Hybridfox was to be usable as the single interface to AWS and Eucalyptus as well as other cloud platforms API compatible with AWS. Recent versions of Hybridfox have been quick to add support for newer features from AWS such as handling streaming media etc.

## Features

- List available Machine images (AMI)
- List running instances
- Launch new instances of an AMI
- Manage security groups and rules
- Manage Snapshots/EBS volumes

## Installation

You can install the extension from <http://code.google.com/p/hybridfox/downloads/list>. The latest version (at the time of writing) is hybridfox-1.6.000040.xpi.

## Configuration

Define a Region

- Click on Regions button
- Enter a logical name for the region (Example: "Eucalyptus" , "MyEucalyptus" etc.)
- Enter the value of EC2\_URL from your eucarc file as the Endpoint URL(<http://192.168.10.121:8773/services/Eucalyptus> in our sample setup)

Define Credentials

- Click on Credentials button
- Enter a logical name for the credential set (Example: "EucaAcc1" etc.)
- Enter EC2\_ACCESS\_KEY and EC2\_SECRET\_KEY from your eucarc file as the AWS Access Key and AWS Secret Access Key respectively

Define Key Pairs

- Click on KeyPairs tab
- Click on "Create a new keypair" icon
- Enter a name for the key pair (Example: "eucakey" etc.)
- Choose location on the client machine to save the id file to (You will need this to use with putty etc. to make an SSH connection)

Define Security Groups

- Click on SecurityGroups tab



- Enter a name for the group (Example: "Eucalyptus", "EucaGroup" etc.)
- Enter the description and click on create button

You will have an option to specify some basic ports like SSH/RDP to be opened up while creating the group itself. After the group is created, you can add rules any time by choosing the security group in the left pane titled "Your groups" and adding rules in the right pane titled "Group Permissions"

## Euca2ools-Command Line Tools

Euca2ools from Eucalyptus provide a bunch of command line tools to manage the eucalyptus setup. These commands help you manage images, instances, storage, networking etc. A few commands related to managing the instances are given below.

For a complete list of commands, see Appendix.

## Installation

```
sudo apt-get install euca2ools
```

## Creation of Key Pairs

OpenStack expects the client tools to use 2 kinds of credentials. One set of credentials are called Access Key and Secret Key that all clients would need to use to make any requests to the Cloud Controller. Each user registered on the web interface has this set created for him. You can download it from the web interface as mentioned in the chapter on "Web Interface".

You will also need to generate a keypair consisting of private key/public key to be able to launch instances on Eucalyptus. These keys are injected into the instances to make password-less SSH access to the instance possible. This depends on the way the necessary tools are bundled into the images. Please refer to the chapter on Image Management for more details.

Keypairs can also be generated using the following commands.

```
cd ~/creds
euca-add-keypair mykey > mykey.priv
chmod 600 mykey.priv
```

This creates a new keypair called mykey. The private key mykey.priv is saved locally which can be used to connect to an instance launched with mykey as the keypair. euca-describe-keypairs command to list the available keypairs.

The output should like this:

```
u.ecadmin@client1:~$ euca-describe-keypairs
```

```
KEYPAIR mykey f7:ac:8e:f5:05:19:2b:31:28:8c:9b:d7:b8:07:0c:3c:b6:34:8f:79
KEYPAIR helloworld
12:96:b3:21:34:8d:6a:3f:92:2e:2b:70:23:ff:7f:51:b5:b7:ad:37
KEYPAIR ubuntu f6:af:9a:59:65:35:32:c4:3a:c4:62:0e:e1:44:0f:71:29:03:2d:91
KEYPAIR lucid 74:04:70:33:ed:57:7a:30:36:1f:ca:c6:ec:d5:4f:10:34:1a:52:51
KEYPAIR karmic 01:f9:aa:5f:4d:20:e2:53:d1:29:d0:0f:e2:f3:8c:21:91:70:7e:c8
```

To delete an existing keypair:

```
euca-delete-keypair helloworld
```

The above tasks can be achieved using Hybridfox from the "Keypairs" tab.

## Launch and manage instances

There are several commands that help in managing the instances. Here are a few examples:

```
$ euca-run-instances emi-721D0EBA -k mykey -t c1.medium
RESERVATION r-55560977 admin admin-default
INSTANCE i-50630A2A emi-721D0EBA 0.0.0.0 0.0.0.0 pending mykey
2010-05-07T07:17:48.23Z eki-675412F5 eri-A1E113E0

$ euca-describe-instances
RESERVATION r-55560977 admin default
INSTANCE i-50630A2A emi-721D0EBA 192.168.3.130 192.168.3.
130 running mykey 0 c1.medium 2010-05-07T07:17:48.23Z
myuecluster eki-675412F5 eri-A1E113E0

$ euca-reboot-instances i-50630A2A

$ euca-terminate-instances i-50630A2A

$ euca-run-instances ami-XXXXXXXX -k mykey

$ euca-get-console-output i-50630A2A
i-50630A2A
2010-05-07T07:22:40.795Z
[ 0.000000] Initializing cgroup subsys cpuset
[ 0.000000] Initializing cgroup subsys cpu
[ 0.000000] Linux version 2.6.32-21-server (buildd@yellow) (gcc version 4.
4.3 (Ubuntu 4.4.3-4ubuntu5) ) #32-Ubuntu SMP Fri Apr 16 09:17:34 UTC 2010 (Ub
untu 2.6.32-21.32-server 2.6.32.11+drm33.2)
[ 0.000000] Command line: root=/dev/sdal console=ttyS0
[ 0.000000] KERNEL supported cpus:
[ 0.000000] Intel GenuineIntel
[ 0.000000] AMD AuthenticAMD
[ 0.000000] Centaur CentaurHauls
[ 0.000000] BIOS-provided physical RAM map:
[ 0.000000] BIOS-e820: 0000000000000000 - 000000000009f000 (usable)
[ 0.000000] BIOS-e820: 000000000009f000 - 00000000000a0000 (reserved)
.....
```

You can make password less ssh access to the instance as follows:

```
ssh -i mykey.priv user@192.168.3.130
```

Make sure that you launch the instance with the correct VM type. If it is launched with a smaller VM type, then the following error is encountered.

```
error: insufficient disk capacity remaining
```

By default, m1.small is the VM type that is used which comes with a 2GB Hard Disk. So if you have a disk image of size 5GB, instead of

```
euca-run-instances ami-XXXXXXXX -k mykey
```

use

```
euca-run-instances ami-XXXXXXXX -k mykey -t c1.medium
```

VM type also has implications for amount of RAM and number of CPUs allocated to the instance.

Check the VM types available.

```
sudo nova-manage instance_type list
```

## Using the OpenStack Dashboard

### Creating Keypairs

The screenshot shows the OpenStack Dashboard interface. On the left is a navigation sidebar with 'Home' at the top, followed by 'Projects' and a sub-menu for 'PROJ' containing 'Instances', 'Images', 'Keys', and 'Volumes'. The main content area is titled 'Project: proj' and 'Region: nova'. Below this is a 'Keys' section with a descriptive text: 'Key pairs are ssh credentials which are injected into images when they are launched. Creating a new key pair registers the public key and downloads the private key (a pem file). Protect and use the key as a normal private key.' Below the text is a table with two columns: 'Key Pair Name' and 'Fingerprint'. The table contains one entry: 'mykey' with fingerprint '8c:90:35:70:90:d4:25:47:18:86:66:cd:4fef:2c:ff' and a 'Delete' button. Below the table is a 'Create New Keypair' form with a 'Name' field containing 'newkeypair' and a 'Create' button.

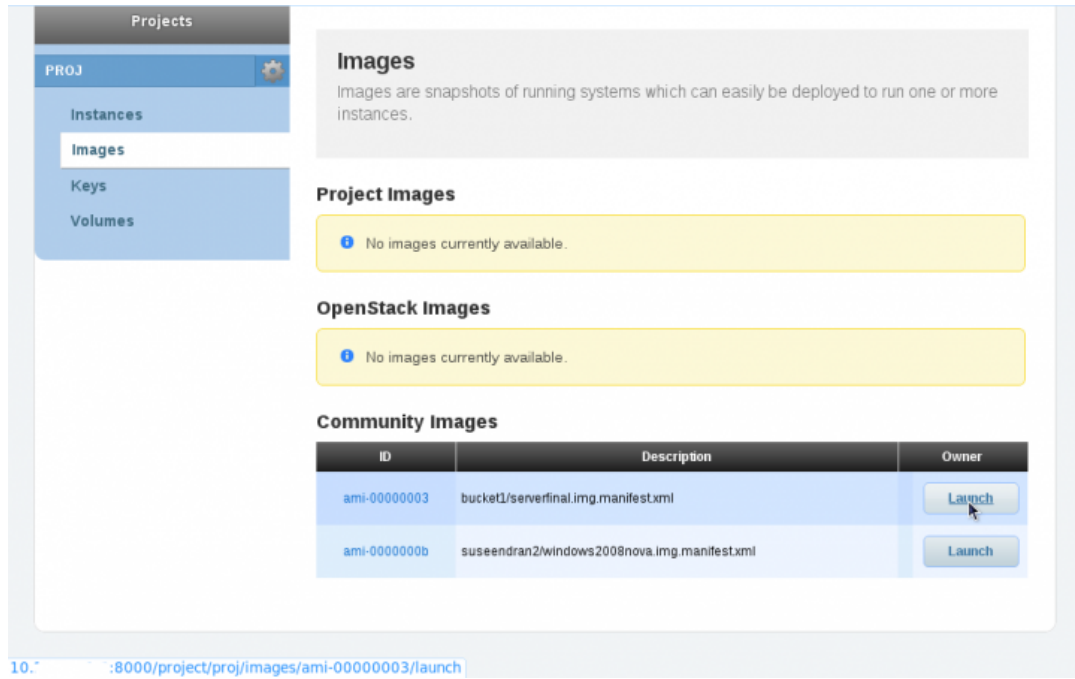
Key Pair Name	Fingerprint	
mykey	8c:90:35:70:90:d4:25:47:18:86:66:cd:4fef:2c:ff	Delete

Create New Keypair

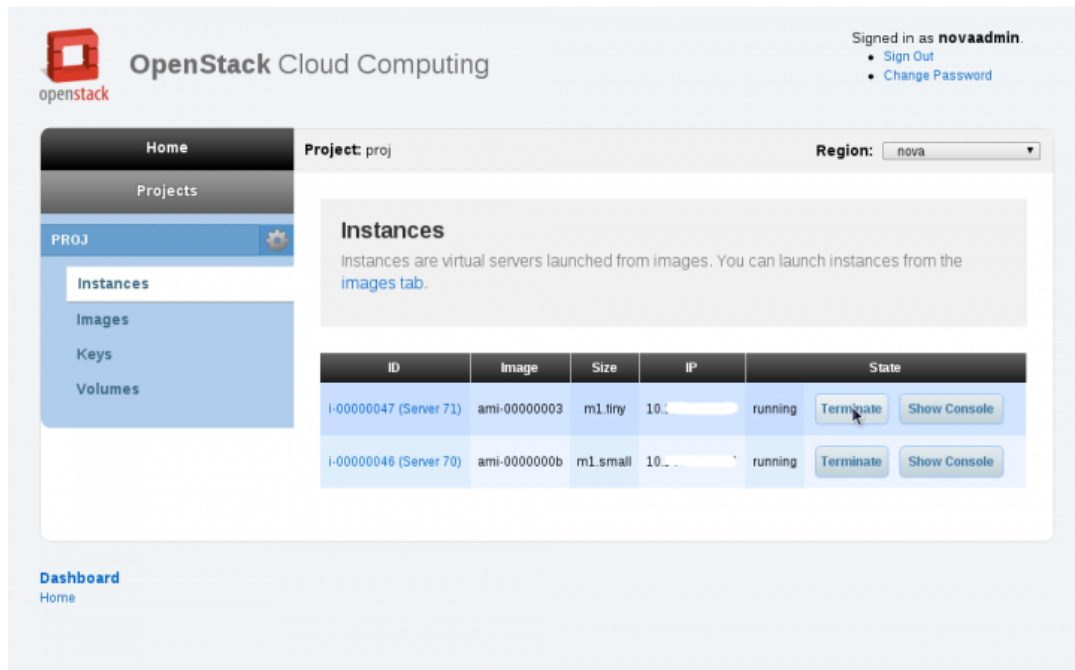
Name:

Create

## Launching an instance



## Terminating an instance



## Displaying the Console Output

The screenshot shows the OpenStack Cloud Computing dashboard. The user is signed in as 'novaadmin'. The dashboard is for project 'proj' in the 'nova' region. The 'Instances' page is active, showing a table of instances. The table has columns for ID, Image, Size, IP, and State. Two instances are listed: i-00000047 (Server 71) and i-00000046 (Server 70). Both are in a 'running' state. The 'Show Console' button for the first instance is highlighted.

ID	Image	Size	IP	State		
i-00000047 (Server 71)	ami-00000003	m1.tiny	10.0.0.1	running	Terminate	Show Console
i-00000046 (Server 70)	ami-0000000b	m1.small	10.0.0.2	running	Terminate	Show Console

## 5. Storage Management

### Table of Contents

Nova-volume .....	35
Interacting with Storage Controller .....	35

### Nova-volume

Nova-volume provides persistent block storage compatible with Amazon's Elastic Block Store. The storage on the instances is non persistent in nature and hence any data that you generate and store on the file system on the first disk of the instance gets lost when the instance is terminated. You will need to use persistent volumes provided by nova-volume if you want any data generated during the life of the instance to persist after the instance is terminated.

Commands from euca2ools package can be used to manage these volumes.

Here are a few examples:

### Interacting with Storage Controller

Make sure that you have sourced novarc before running any of the following commands. The following commands refer to a zone called 'nova', which we created in the chapter on "Installation and Configuration". The project is 'proj' as referred to in the other chapters.

Create a 10 GB volume

```
euca-create-volume -s 10 -z nova
```

You should see an output like:

```
VOLUME    vol-00000002    1    creating (proj, None, None, None)
2011-04-21T07:19:52Z
```

List the volumes

```
euca-describe-volumes
```

You should see an output like this:

```
VOLUME    vol-00000001    1    nova    available (proj, server1, None,
None)    2011-04-21T05:11:22Z
VOLUME    vol-00000002    1    nova    available (proj, server1, None,
None)    2011-04-21T07:19:52Z
```

Attach a volume to a running instance

```
euca-attach-volume -i i-00000009 -d /dev/vdb vol-00000002
```

A volume can only be attached to one instance at a time. When `euca-describe-volumes` shows the status of a volume as 'available', it means it is not attached to any instance and ready to be used. If you run `euca-describe-volumes`, you can see that the status changes from "available" to "in-use" if it is attached to an instance successfully.

When a volume is attached to an instance, it shows up as an additional SCSI disk on the instance. You can login to the instance and mount the disk, format it and use it.

Detach a volume from an instance.

```
euca-detach-volume vol-00000002
```

The data on the volume persists even after the volume is detached from an instance. You can see the data on reattaching the volume to another instance.

Even though you have indicated `/dev/vdb` as the device on the instance, the actual device name created by the OS running inside the instance may differ. You can find the name of the device by looking at the device nodes in `/dev` or by watching the syslog when the volume is being attached.

# 6. Network Management

## Table of Contents

Introduction .....	37
--------------------	----

## Introduction

In OpenStack, the networking is managed by a component called "nova-network". This interacts with nova-compute to ensure that the instances have the right kind of networking setup for them to communicate among themselves as well as with the outside world. Just as in Eucalyptus or AWS, each OpenStack instance can have 2 IP addresses attached to it. One is the private IP address and the other called Public IP address. The private IP address is typically used for communication between instances and the public IP is used for communication of instances with the outside world. The so called public IP address need not be a public IP address route-able on the Internet ; it can even be an address on the corporate LAN.

The network configuration inside the instance is done with the private IP address in view. The association between the private IP and the public IP and necessary routing is handled by nova-network and the instances need not be aware of it.

nova-network provides 3 different network management options. Currently you can only choose one of these 3 options for your network management.

- Flat Network
- Flat DHCP Network
- VLAN Network

VLAN Network is the most feature rich and is the idea choice for a production deployment, while the other modes can be used while getting familiar with OpenStack and when you do not have VLAN Enabled switches to connect different components of the OpenStack infrastructure.

The network type is chosen by using one of the following configuration options in nova.conf file. If no network manager is specified explicitly, the default network manager, VLANManager is used.

```
--network_manager = nova.network.manager.FlatManager  
--network_manager = nova.network.manager.FlatDHCPManager  
--network_manager = nova.network.manager.VlanManager
```

In each of these cases, run the following commands to set up private and public IP addresses for use by the instances:



```
sudo nova-manage network create 192.168.3.0/24 1 255  
sudo nova-manage floating create 10.10.10.2 10.10.10.224/27
```

The public IP which you are going to associate with an instance needs to be allocated first by using "euca-allocate-address" command:

```
euca-allocate-address 10.10.2.225
```

You can then associate a public IP to a running instance by using "euca-associate-address" command:

```
euca-associate-address -i i-0000008 10.10.2.225
```

Please refer to <http://docs.openstack.org/openstack-compute/admin/content/ch04.html> for more details about each of the networking types.

# 7. Role Based Access Control

## Table of Contents

Role Based Access Control Overview .....	39
Administrator(admin) .....	39
IT security(itsec) .....	39
Project Manager (projectmanager) .....	39
Network Administrator(netadmin) .....	40
Developer (developer) .....	40
Tabular representation of Roles .....	40

## Role Based Access Control Overview

Every nova user has an associated role. This role can be assigned at the time of creation of the account using "nova-manage add user (name)" or by editing the profile later using the OpenStack Dashboard by the project manager. The role can be either global or project specific in scope. All access in OpenStack is governed by roles. Each role has a predefined set of operations permitted within the relevant scope (global or local).

### Administrator(admin)

This is a project based role. Users who are created with admin roles at time of creation. They enjoy the rights as a administrator for carrying out tasks such as

- adding an instance
- removing an instance
- removing an image
- adding a key

### IT security(itsec)

This is a global role. It permits role holders to quarantine instances.

### Project Manager (projectmanager)

This is the default role for project owners. It permits the following tasks:

- adding available roles to user associated in project
- revoking provided roles to a specific user in the project.
- adding an instance
- removing an instance

- removing an image
- adding a key
- managing network related operations

## Network Administrator(**netadmin**)

A role which allows particular user to carry out network related operations such as:

- allocate publicly accessible IP addresses
- assign publicly accessible IP addresses
- create firewall rules
- modify firewall rules

## Developer (**developer**)

This is a general purpose role that is assigned to users by default. This role can create and download keys.

Summary of role and permitted tasks for each role:

Role Management is done using "nova-manage role" command. Please refer to the section on OpenStack Commands for more details.

Examples:

Add role to a user

```
nova-manage role add user1 netadmin
```

Remove a role from a particular user

```
nova-manage role remove user1 netadmin
```

## Tabular representation of Roles

**Table 7.1. Role representation**

Roles	Global	Local	Key mgmt.	Instance mgmt.	Image mgmt.	Network mgmt.	Project mgmt.	Creating / Modifying Firewall Rules
Developer	No	Yes	Yes	No	No	No	No	No
Project Manager	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
It Security	Yes	No	Yes	No	No	No	No	No
Cloud Admin	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Net Admin	No	No	No	No	No	No	No	Yes

## 8. Security

### Table of Contents

Security Overview .....	41
-------------------------	----

### Security Overview

OpenStack provides ingress filtering for the instances based on the concept of security groups. OpenStack accomplishes ingress filtering by creating suitable IP Tables rules. A Security Group is a named set of rules that get applied to the incoming packets for the instances. You can specify a security group while launching an instance. Each security group can have multiple rules associated with it. Each rule specifies the source IP/network, protocol type, destination ports etc. Any packet matching these parameters specified in a rule is allowed in. Rest of the packets are blocked.

A security group that does not have any rules associated with it causes blocking of all incoming traffic. The mechanism only provides ingress filtering and does not provide any egress filtering. As a result all outbound traffic is allowed. If you need to implement egress filtering, you will need to implement that inside the instance using a firewall.

Tools like Hybridfox let you manage security groups and also let you specify a security group while launching an instance. You can also use command line tools from euca2ools package such as euca-authorize for this purpose.

Here are a few euca commands to manage security groups. Like in our earlier chapters, the project name is "proj"

Create a security group named "myservers".

```
euca-add-group -d "My Servers" myservers
```

Add a rule to the security group "myservers" allowing icmp and tcp traffic from 192.168.1.1.

```
euca-authorize -P tcp -s 192.168.1.1 -p 22 myservers  
euca-authorize -P icmp -s 192.168.1.1 -t -1:-1 myservers
```

For a Windows instance, add a rule to accept incoming RDP connections

```
euca-authorize -P tcp -s 192.168.1.1 -p 3389 myservers
```

Rules can be viewed with euca-describe-groups command.

```
$ euca-describe-groups
```

GROUP	proj	myservers	my servers						
PERMISSION	proj	myservers	ALLWS	tcp	22	22	FROM	CIDR	192.168.1.1
PERMISSION	proj	myservers	ALLWS	icmp	-1	-1	FROM	CIDR	192.168.1.1
PERMISSION	proj	myservers	ALLWS	tcp	3389	3389	FROM	CIDR	192.168.1.1

Remove the rule for ssh traffic from the source ip 192.168.1.1 from the security group "myservers"

```
euca-revoke -P tcp -s 192.168.1.1 -p 22 myservers
```

Delete the security group "myservers"

```
euca-delete-group myservers
```

Launch an instance associated with the security group "myservers".

```
euca-run-instances emi-XXXXXXXX -k mykey -g myservers
```




When you do not specify a security group, the instance gets associated with an inbuilt security group called "default". The rules for this security group can also be modified using `euca-add`, `euca-revoke` commands.


Security Groups can also be viewed and manipulated using tools like HybridFox. A screenshot of listing of security groups in HybridFox is below:

Regions nova Credentials novamain



Instances Images KeyPairs Security Groups Elastic IPs Volumes and Snapshots Bundle Tasks Availability Zones Res

### Your Groups

Owner	Name	Description	
css	default	default	

### Group Permissions

Protocol	From Port/ICMP Type	
icmp	-1	-
tcp	22	2

## 9. OpenStack Commands

### Table of Contents

Nova Manage Commands .....	44
User/Role Management .....	44
Project Management .....	45
Database Management .....	46
Instance Type Management .....	46
Service Management .....	47
Euca2ools Commands .....	47

### Nova Manage Commands

OpenStack provides commands for administrative tasks such as user/role management, network management etc. In all the examples we will use username as "novadmin" and project name as "proj". All the nova-manage commands will need to be run as "root". Either run them as root or run them under sudo.

#### User/Role Management

Add a new user

```
nova-manage user create novaadmin
```

Add a user with admin privileges

```
nova-manage user admin novaadmin
```

List existing users

```
nova-manage user list
```

Delete an existing user

```
nova-manage user delete novaadmin
```

Associate a user to a specific existing project

```
nova-manage project add proj novaadmin
```

Remove a user from a specific existing project.

```
nova-manage project remove proj novaadmin
```

View access key and secret keys of particular user.

```
nova-manage user exports novaadmin
```

Add a role to a particular user. Please refer to the chapter on "Role Based Access Controls" for more details on role management.

```
nova-manage role add novaadmin netadmin
```

Remove a role from a particular user

```
nova-manage role remove novaadmin netadmin
```

With the command below, you can change any or all of access key, secret key and admin role flag for a particular user.

```
Syntax:
nova-manage user modify username new_access_key new_secret_key admin_flag
<admin flag - T or F>

nova-manage user modify novaadmin mygreatnewaccesskey "" ""
nova-manage user modify novaadmin "" mygreatsecretkey ""
nova-manage user modify novaadmin "" "" T
```

Check if a particular user has a specific role or not. The role can be either local or global. The output of the command will be True or False

```
nova-manage role has novaadmin cloudadmin
True

nova-manage role has novaadmin netadmin proj
False
```

## Project Management

The following commands help you create and manage projects. "nova-manage account" command is an alias to "nova-manage project" and you can use them interchangeably.

Create a project. It requires you to mention name of the project admin as well. css1 is the name of the project and user5 is the name of the project admin here.

```
nova-manage project create css1 user5
```

List the registered projects.

```
nova-manage project list
```

Download the credentials and associated file for a specific project. Please refer to the chapter on "Installation & Configuration" for more details.

```
nova-manage project zipfile csscorp user5 /home/user5/mysec.zip
```



Delete an existing project.

```
nova-manage project delete css1
```

Check the project wise resource allocation. The output will look like this:

```
nova-manage project quota css1
metadata_items: 128
gigabytes: 1000
floating_ips: 10
instances: 10
volumes: 10
cores: 20
```

## Database Management

Nova stores the data related to the projects, users, resources etc. in a database, by default in a MySQL database.

Print the current database version.

```
nova-manage db version
```

Sync the DB schema to be in sync with the current configuration.

```
nova-manage db sync
```

## Instance Type Management

Nova has the concept of instance types. Each instance type is defined with certain amount of RAM and certain size of the hard disk. When an instance is launched with a particular instance type, Nova resizes the disk image to suit the instance type and allocates the RAM as defined for the instance type chosen. Nova calls instance types as 'flavors' and lets you add to the list of flavors. By default Nova has 5 types - m1.tiny, m1.small, m1.medium, m1.large and m1.xlarge.

List the current instance types

```
nova-manage flavor list
m1.medium: Memory: 4096MB, VCPUS: 2, Storage: 40GB, FlavorID: 3, Swap: 0GB,
RXTX Quota: 0GB, RXTX Cap: 0MB
m1.large: Memory: 8192MB, VCPUS: 4, Storage: 80GB, FlavorID: 4, Swap: 0GB,
RXTX Quota: 0GB, RXTX Cap: 0MB
m1.tiny: Memory: 512MB, VCPUS: 1, Storage: 0GB, FlavorID: 1, Swap: 0GB,
RXTX Quota: 0GB, RXTX Cap: 0MB
m1.xlarge: Memory: 16384MB, VCPUS: 8, Storage: 160GB, FlavorID: 5, Swap: 0GB,
RXTX Quota: 0GB, RXTX Cap: 0MB
m1.small: Memory: 2048MB, VCPUS: 1, Storage: 20GB, FlavorID: 2, Swap: 0GB,
RXTX Quota: 0GB, RXTX Cap: 0MB
```

Define a new instance type

```
nova-manage flavor create m1.verytiny 256 2 20 6 0 0 0
```

Remove an existing instance type.

```
nova-manage flavor delete m1.verytiny
m1.verytiny deleted
```

## Service Management

Check state of available services.

```
nova-manage service list
server1 nova-scheduler enabled :- ) 2011-04-06 17:01:21
server1 nova-network enabled :- ) 2011-04-06 17:01:30
server1 nova-compute enabled :- ) 2011-04-06 17:01:22
server2 nova-compute enabled :- ) 2011-04-06 17:01:28
```

Disable a running service

```
nova-manage service disable <hostname> <service>
nova-manage service disable server2 nova-compute

nova-manage service list
server1 nova-network enabled :- ) 2011-04-06 17:05:11
server1 nova-compute enabled :- ) 2011-04-06 17:05:13
server1 nova-scheduler enabled :- ) 2011-04-06 17:05:17
server2 nova-compute disabled :- ) 2011-04-06 17:05:19
```

Re-enable a service that is currently disabled

```
Syntax: nova-manage service enable <hostname> <service>
nova-manage service enable server2 nova-compute

nova-manage service list
server1 nova-scheduler enabled :- ) 2011-04-06 17:08:23
server1 nova-network enabled :- ) 2011-04-06 17:08:22
server1 nova-compute enabled :- ) 2011-04-06 17:08:23
server2 nova-compute enabled :- ) 2011-04-06 17:08:19
```

Get Information about resource utilization of the OpenStack components

```
Syntax: nova-manage service describe_resource <hostname>

nova-manage service describe_resource server1
HOST PROJECT cpu mem(mb) disk(gb)
server1(total) 2 3961 224
server1(used) 1 654 30
server1 proj 2 1024 0
```

## Euca2ools Commands

euca2ools provide a set of commands to communicate with the cloud. All these commands require you to authenticate and this is done by sourcing novarc file as detailed in the chapter on "Installation & Configuration"

Most of the euca2ools command line utilities work with OpenStack, just as they work with EC2 of AWS. There may be some differences due to some of the functionality that is yet to be implemented in OpenStack. Help is available for each of these commands with the switch `-help`.

- euca-add-group
- euca-delete-bundle
- euca-describe-instances
- euca-register
- euca-add-keypair
- euca-delete-group
- euca-describe-keypairs
- euca-release-address
- euca-allocate-address
- euca-delete-keypair
- euca-describe-regions
- euca-reset-image-attribute
- euca-associate-address
- euca-delete-snapshot
- euca-describe-snapshots
- euca-revoke
- euca-attach-volume
- euca-delete-volume
- euca-describe-volumes
- euca-run-instances
- euca-authorize
- euca-deregister
- euca-detach-volume
- euca-terminate-instances
- euca-bundle-image
- euca-describe-addresses

- euca-disassociate-address
- euca-unbundle
- euca-bundle-vol
- euca-describe-availability-zones
- euca-download-bundle
- euca-upload-bundle
- euca-confirm-product-instance
- euca-describe-groups
- euca-get-console-output
- euca-version
- euca-create-snapshot
- euca-describe-image-attribute
- euca-modify-image-attribute
- euca-create-volume
- euca-describe-images
- euca-reboot-instances