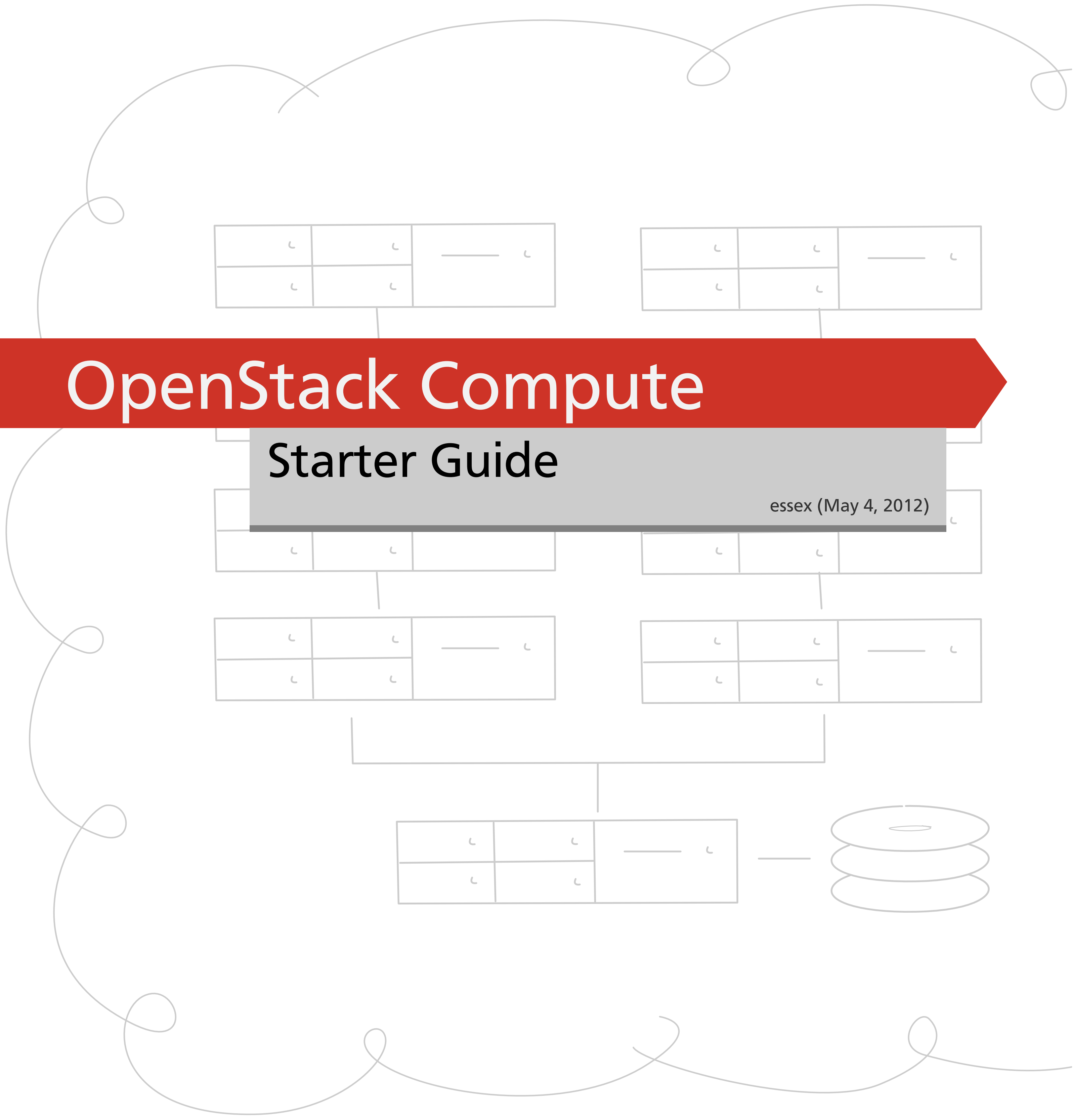


# OpenStack Compute

## Starter Guide

essex (May 4, 2012)

docs.openstack.org

# OpenStack Compute Starter Guide

essex (2012-05-04)

This is a tutorial style beginner's guide for OpenStack™ on Ubuntu 12.04, Precise Pangolin. The aim is to help the reader in setting up a minimal installation of OpenStack.

# Table of Contents

# List of Tables

# 1. Introduction to OpenStack and Its Components

## 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:

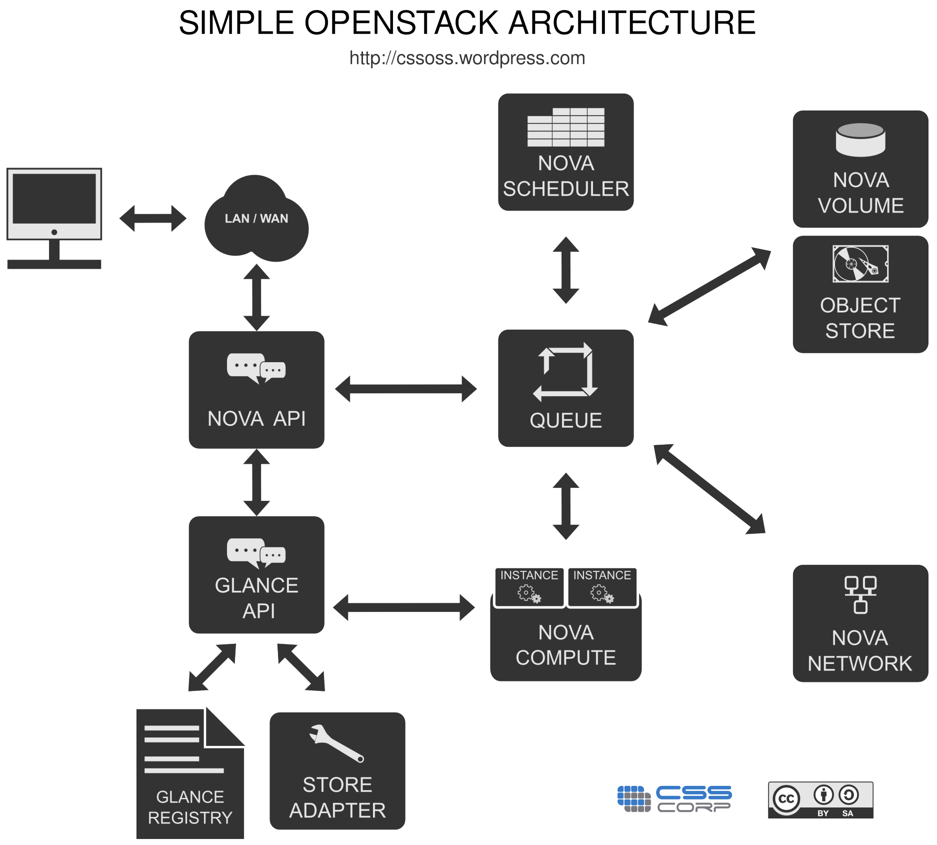
- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

## OpenStack

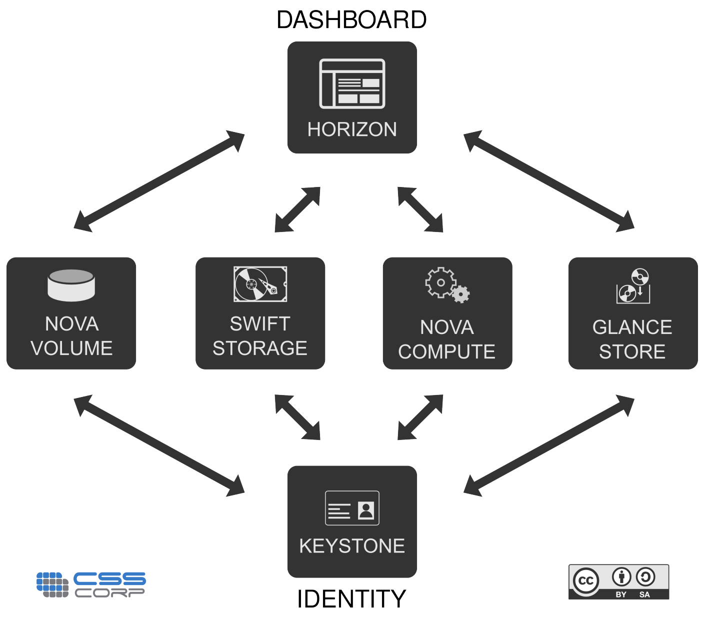
OpenStack is a collection of open source software projects that enterprises/service providers can use to setup and run their cloud compute and storage infrastructure. Rackspace and NASA are the key initial contributors to the stack. Rackspace contributed their "Cloud Files" platform (code) to power the Object Storage part of the OpenStack, while NASA contributed their "Nebula" platform (code) to power the Compute part. OpenStack consortium has managed to have more than 150 members including Canonical, Dell, Citrix etc.

There are 5 main service families under OpenStack

- Nova - Compute Service
- Swift - Storage Service
- Glance - Imaging Service
- Keystone - Identity Service
- Horizon - UI Service



The below diagram shows a simple representation of interaction between keystone and dashboard with the remaining OpenStack components.



# Open Stack Compute Infrastructure (Nova)

Nova is the 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. This makes Nova a Management Platform that manages compute resources, networking, authorization, and scalability needs of the OpenStack cloud. But, Nova does not provide any virtualization capabilities by itself; instead, it uses libvirt API to interact with supported hypervisors. Nova exposes all its capabilities through a web services API that is compatible with the EC2 API 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 for 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. As an alternative to EC2 API, OpenStack also provides a native API called "OpenStack API".

### Message Queue (Rabbit MQ Server)

OpenStack communicates among themselves using the message queue 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 user actions get stuck for long in a waiting state. This is effective 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 instance 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 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 the instances, as the root partition is non-persistent and any changes made to it are lost when an 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. This data can be accessed by re-attaching the volume to the same instance or by attaching it to other instances.

Critical data in an instance must always 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 server from a pool of available resources depending on 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 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 storage backends:

- Local filesystem (default)
- OpenStack Object Store to store images
- S3 storage directly
- S3 storage with Object Store as the intermediate for S3 access.
- HTTP (read-only)

### Functions and Features

- Provides imaging service

### Components of Glance

- Glance-control
- Glance-registry

## OpenStack Storage Infrastructure (Swift)

Swift provides a distributed, eventually consistent virtual object store for OpenStack. It is analogous to Amazon Web Services - Simple Storage Service (S3). Swift is capable of storing billions of objects distributed across nodes. Swift has built-in redundancy and failover management and is capable of archiving and media streaming. It is extremely scalable in terms of both size (several petabytes) and capacity (number of objects).

### Functions and Features

- Storage of large number of objects
- Storage of large sized objects
- Data Redundancy
- Archival capabilities - Work with large datasets
- Data container for virtual machines and cloud apps
- Media Streaming capabilities
- Secure storage of objects
- Backup and archival
- Extreme scalability

### Components of Swift

- Swift Account

- Swift Container
- Swift Object
- Swift Proxy
- The RING

### Swift Proxy Server

The consumers interact with the Swift setup through the proxy server using the Swift API. The proxy server acts as a gatekeeper and recieves requests from the world. It looks up the location of the appropriate entities and routes the requests to them.

The proxy server also handles failures of entities by rerouting requests to failover entities (handoff entities)

### Swift Object Server

The Object server is a blob store. It's responsibility is to handle storage, retrieval and deletion of objects stored in the local storage. Objects are typically binary files stored in the filesystem with metadata contained as extended file attributes (xattr).

Note: xattr is supported in several filesystems such as ext3, ext4, XFS, Btrfs, JFS and ReiserFS in Linux. But it is known to work best under XFS, JFS, ReiserFS, Reiser4, and ZFS. XFS is considered to be the best option.

### Swift Container server

The container server lists the objects in a container. The lists are stored as SQLite files. The container server also tracks the statistics like the number of objects contained and the storage size occupied by a container.

### Swift Account Server

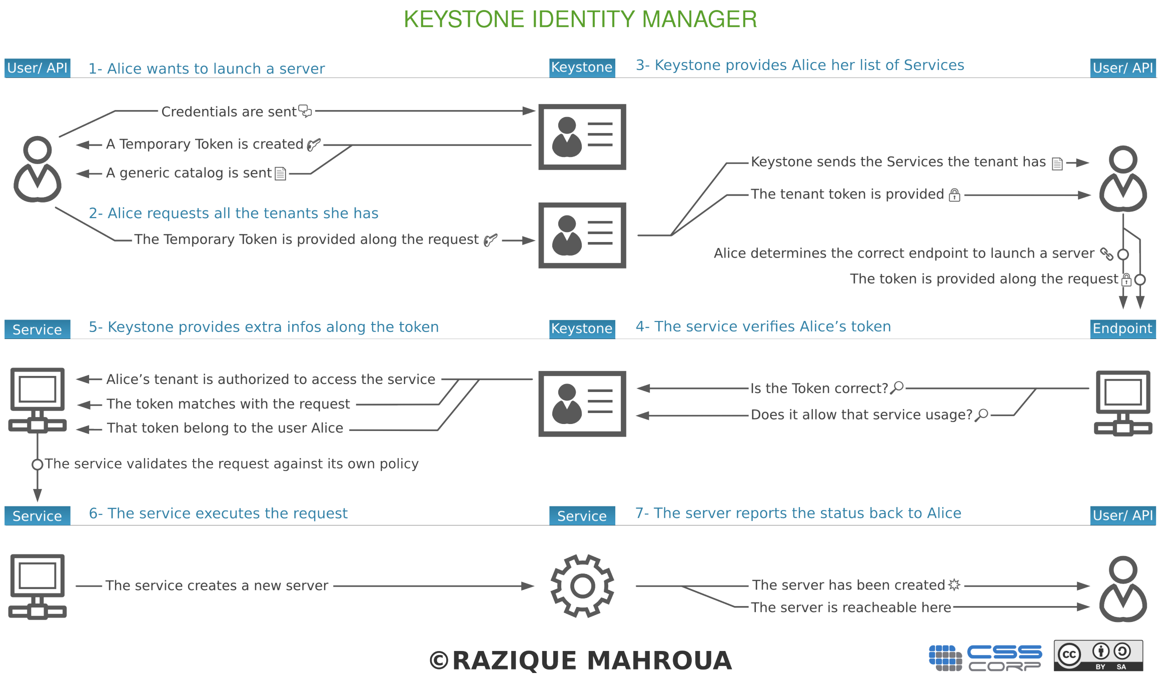
The account server lists containers the same way a container server lists objects.

### The Ring

The ring contains information about the physical location of the objects stored inside Swift. It is a virtual representation of mapping of names of entities to their real physical location. It is analogous to an indexing service that various processes use to lookup and locate the real physical location of entities within the cluster. Entities like Accounts, Containers, Objects have their own seperate rings.

## OpenStack Identity Service (Keystone)

Keystone provides identity and access policy services for all components in the OpenStack family. It implements it's own REST based API (Identity API). It provides authentication and authorization for all components of OpenStack including (but not limited to) Swift, Glance, Nova. Authentication verifies that a request actually comes from who it says it does. Authorization is verifying whether the authenticated user has access to the services he/she is requesting for.


KEYSTONE IDENTITY MANAGER
User/ API
1- Alice wants to launch a server
Credentials are sent
A Temporary Token is created
A generic catalog is sent
2- Alice requests all the tenants she has
The Temporary Token is provided along the request
Keystone
3- Keystone provides Alice her list of Services
User/ API
Keystone sends the Services the tenant has
The tenant token is provided
Alice determines the correct endpoint to launch a server
The token is provided along the request
Service
5- Keystone provides extra infos along the token
Alice's tenant is authorized to access the service
The token matches with the request
That token belong to the user Alice
The service validates the request against its own policy
Keystone
4- The service verifies Alice's token
Endpoint
Is the Token correct?
Does it allow that service usage?
Service
6- The service executes the request
The service creates a new server
Service
7- The server reports the status back to Alice
User/ API
The server has been created
The server is reacheable here
©RAZIQUE MAHROUA
CSS CORP

Keystone provides two ways of authentication. One is username/password based and the other is token based. Apart from that, keystone provides the following services:

- Token Service (that carries authorization information about an authenticated user)
- Catalog Service (that contains a list of available services at the users' disposal)
- Policy Service (that let's keystone manage access to specific services by specific users or groups).

## Components of Identity Service

- Endpoints - Every OpenStack service (Nova, Swift, Glance) runs on a dedicated port and on a dedicated URL(host), we call them endpoints.
- Regions - A region defines a dedicated physical location inside a data centre. In a typical cloud setup, most if not all services are distributed across data centers/servers which are also called regions
- User - A keystone authenticated user.
- Services - Each component that is being connected to or being administered via keystone can be called a service. For example, we can call Glance a keystone service.
- Role - In order to maintain restrictions as to what a particular user can do inside cloud infrastructure it is important to have a role associated.
- Tenant - A tenant is a project with all the service endpoint and a role associated to user who is member of that particular tenant.

# Openstack Administrative Web-Interface (Horizon)

Horizon the web based dashboard can be used to manage /administer OpenStack services. It can be used to manage instances and images, create keypairs, attach volumes to instances, manipulate Swift containers etc. Apart from this, dashboard even gives the user access to instance console and can connect to an instance through VNC. Overall, Horizon features the following:

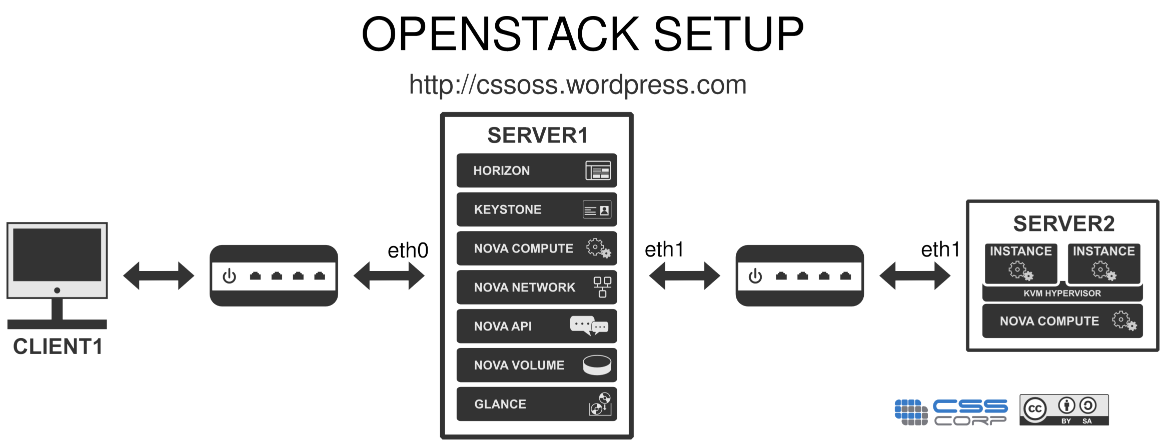
- Instance Management - Create or terminate instance, view console logs and connect through VNC, Attaching volumes, etc.
- Access and Security Management - Create security groups, manage keypairs, assign floating IPs, etc.
- Flavor Management - Manage different flavors or instance virtual hardware templates.
- Image Management - Edit or delete images.
- View service catalog.
- Manage users, quotas and usage for projects.
- User Management - Create user, etc.
- Volume Management - Creating Volumes and snapshots.
- Object Store Manipulation - Create, delete containers and objects.
- Downloading environment variables for a project.

# 2. Installation and Configuration

## 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, Server2 and Client1. Server1 runs all the components of Nova, Glance, Swift, Keystone and Horizon (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 OpenStack 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 launching of 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

| Server1 | Server2 | Client1 | |
|---|---|---|---|
| Functionality | All components of OpenStack including nova-compute | nova-compute | Client |
| Network Interfaces | 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 | 10.10.8.3 | 10.10.8.3 | 10.10.8.3 |
| 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, Glance, Swift, Keystone and Horizon. It contains two network interface cards (NICs).

## Base OS

Install 64 bit version of Ubuntu server 12.04 keeping the following configurations in mind.

- 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.

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. If you also plan to use a dedicated partition as Swift backend, create another partition for this purpose and follow the instructions in "Swift Installation" section below.

Update the machine using the following commands.

```
sudo apt-get update
sudo apt-get upgrade
```

Install bridge-utils:

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

## Network 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 10.10.8.3

```

```
auto eth1
iface eth1 inet static
  address 192.168.3.1
  netmask 255.255.255.0
  network 192.168.3.0
  broadcast 192.168.3.255
```

Restart the network now

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

## NTP Server

Install NTP package. This server shall act as the NTP server for the nodes. The time on all components of OpenStack will have to be in sync. We can run NTP server on server1 and have other servers/nodes sync to it.

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

Open the file /etc/ntp.conf and add the following lines to make sure that the time on the server stays in sync with an external server. If the Internet connectivity is down, the NTP server uses its own hardware clock as the fallback.

```

server ntp.ubuntu.com
server 127.127.1.0
fudge 127.127.1.0 stratum 10
```

Restart the NTP server

```
sudo service ntp restart
```

Ensure that, IP addresses of the servers are resolvable by the DNS. If not, include the hostnames in /etc/hosts file.

## Databases

You can use MySQL, PostgreSQL or SQLite for Nova and Glance. Depending upon your choice of database, you will need to install the necessary packages and configure the database server.

### MySQL

Install mysql-server and python-mysqldb package

```
sudo apt-get install mysql-server python-mysqldb
```

Create the root password for mysql. The password used in this guide is "mygreatsecret"

Change the bind address from 127.0.0.1 to 0.0.0.0 in /etc/mysql/my.cnf. It should be identical to this:

```
bind-address = 0.0.0.0
```

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

```
sudo restart mysql
```

## Creating Databases

Create MySQL databases to be used with nova, glance and keystone.

Create a database named nova.

```
sudo mysql -uroot -pmygreatsecret -e 'CREATE DATABASE nova;'
```

Create a user named novadbadmin.

```
sudo mysql -uroot -pmygreatsecret -e 'CREATE USER novadbadmin;'
```

Grant all privileges for novadbadmin on the database "nova".

```
sudo mysql -uroot -pmygreatsecret -e "GRANT ALL PRIVILEGES ON nova.* TO
 'novadbadmin'@'%';"
```

Create a password for the user "novadbadmin".

```
sudo mysql -uroot -pmygreatsecret -e "SET PASSWORD FOR 'novadbadmin'@'%' =
 PASSWORD('novasecret');"
```

Create a database named glance.

```
sudo mysql -uroot -pmygreatsecret -e 'CREATE DATABASE glance;'
```

Create a user named glancedbadmin.

```
sudo mysql -uroot -pmygreatsecret -e 'CREATE USER glancedbadmin;'
```

Grant all privileges for glancedbadmin on the database "glance".

```
sudo mysql -uroot -pmygreatsecret -e "GRANT ALL PRIVILEGES ON glance.* TO
 'glancedbadmin'@'%';"
```

Create a password for the user "glancedbadmin".

```
sudo mysql -uroot -pmygreatsecret -e "SET PASSWORD FOR 'glancedbadmin'@'%' =
 PASSWORD('glancesecret');"
```

Create a database named keystone.

```
sudo mysql -uroot -pmygreatsecret -e 'CREATE DATABASE keystone;'
```

Create a user named keystonedbadmin.

```
sudo mysql -uroot -pmygreatsecret -e 'CREATE USER keystonedbadmin;'
```

Grant all privileges for keystonedbadmin on the database "keystone".

```
sudo mysql -uroot -pmygreatsecret -e "GRANT ALL PRIVILEGES ON keystone.* TO
 'keystonedbadmin'@'%';"
```

Create a password for the user "keystonedbadmin".

```
sudo mysql -uroot -pmygreatsecret -e "SET PASSWORD FOR 'keystonedbadmin'@'%' =
 PASSWORD('keystonesecret');"
```

## Keystone

Keystone is the identity service used by OpenStack. Install Keystone using the following command.

```
sudo apt-get install keystone python-keystone python-keystoneclient
```

Open /etc/keystone/keystone.conf and change the line

```
admin_token = ADMIN
```

so that it looks like the following:

```
admin_token = admin
```

(We have used 'admin' as the token in this book.)

Since MySQL database is used to store keystone configuration, replace the following line in /etc/keystone/keystone.conf

```
connection = sqlite:////var/lib/keystone/keystone.db
```

with

```
connection = mysql://keystonedbadmin:keystonesecret@10.10.10.2/keystone
```

Restart Keystone:

```
sudo service keystone restart
```

Run the following command to synchronise the database:

```
sudo keystone-manage db_sync
```

Export environment variables which are required while working with OpenStack.

```

export SERVICE_ENDPOINT="http://localhost:35357/v2.0"
export SERVICE_TOKEN=admin
```

You can also add these variables to ~/.bashrc, so that you need not have to export them everytime.

### Creating Tenants

Create the tenants by executing the following commands. In this case, we are creating two tenants - admin and service.

```

keystone tenant-create --name admin
keystone tenant-create --name service
```

### Creating Users

Create the users by executing the following commands. In this case, we are creating four users - admin, nova, glance and swift

```
keystone user-create --name admin --pass admin --email admin@foobar.com
keystone user-create --name nova --pass nova   --email nova@foobar.com
keystone user-create --name glance --pass glance   --email glance@foobar.com
keystone user-create --name swift --pass swift   --email swift@foobar.com
```

## Creating Roles

Create the roles by executing the following commands. In this case, we are creating two roles - admin and Member.

```
keystone role-create --name admin
keystone role-create --name Member
```

## Listing Tenants, Users and Roles

The tenants, users and roles that have been created above can be listed by following commands:

List Tenants:

```
keystone tenant-list
+----------------------------------+--------------------+---------+
|                id                |        name        | enabled |
+----------------------------------+--------------------+---------+
| 7f95ae9617cd496888bc412efdceabfd | admin              | True    |
| c7970080576646c6959ee35970cf3199 | service            | True    |
+----------------------------------+--------------------+---------+
```

List Users:

```
keystone user-list
+----------------------------------+---------+-------------------+--------+
|                id                | enabled |       email       |  name  |
+----------------------------------+---------+-------------------+--------+
| 1b986cca67e242f38cd6aa4bdec587ca | True    | swift@foobar.com  | swift  |
| 518b51ea133c4facadae42c328d6b77b | True    | glance@foobar.com | glance |
| b3de3aeec2544f0f90b9cbfe8b8b7acd | True    | admin@foobar.com  | admin  |
| ce8cd56ca8824f5d845ba6ed015e9494 | True    | nova@foobar.com   | nova   |
+----------------------------------+---------+-------------------+--------+
```

List Roles:

```
keystone role-list
+----------------------------------+----------------------+
|                id                |         name         |
+----------------------------------+----------------------+
| 2bbe305ad531434991d4281aaaebb700 | admin                |
| d983800dd6d54ee3a1b1eb9f2ae3291f | Member               |
+----------------------------------+----------------------+
```

Please note that the values of the 'id' column, would be required later when we associate a role to a user in a particular tenant.

## Adding Roles to Users in Tenants

Now we add roles to the users that have been created. A role to a specific user in a specific tenant can be assigned with the following command:

```
keystone user-role-add --user $USER_ID --role $ROLE_ID --tenant_id $TENANT_ID
```

The required 'id' can be obtained from the commands - keystone user-list, keystone tenant-list, keystone role-list.

To add a role of 'admin' to the user 'admin' of the tenant 'admin'.

```
keystone user-role-add --user b3de3aeec2544f0f90b9cbfe8b8b7acd --role
 2bbe305ad531434991d4281aaaebb700 --tenant_id 7f95ae9617cd496888bc412efdceabfd
```

The following commands will add a role of 'admin' to the users 'nova', 'glance' and 'swift' of the tenant 'service'.

```
keystone user-role-add --user ce8cd56ca8824f5d845ba6ed015e9494 --role
 2bbe305ad531434991d4281aaaebb700 --tenant_id c7970080576646c6959ee35970cf3199
keystone user-role-add --user 518b51ea133c4facadae42c328d6b77b --role
 2bbe305ad531434991d4281aaaebb700 --tenant_id c7970080576646c6959ee35970cf3199
keystone user-role-add --user 1b986cca67e242f38cd6aa4bdec587ca --role
 2bbe305ad531434991d4281aaaebb700 --tenant_id c7970080576646c6959ee35970cf3199
```

The 'Member' role is used by Horizon and Swift. So add the 'Member' role accordingly.

```
keystone user-role-add --user b3de3aeec2544f0f90b9cbfe8b8b7acd --role
 d983800dd6d54ee3a1b1eb9f2ae3291f --tenant_id 7f95ae9617cd496888bc412efdceabfd
```

Replace the id appropriately as listed by keystone user-list, keystone role-list, keystone tenant-list.

## Creating Services

Now we need to create the required services which the users can authenticate with. nova-compute, nova-volume, glance, swift, keystone and ec2 are some of the services that we create.

```
keystone service-create --name service_name --type service_type --description
 'Description of the service'
```

```
keystone service-create --name nova --type compute --description 'OpenStack
 Compute Service'
keystone service-create --name volume --type volume --description 'OpenStack
 Volume Service'
keystone service-create --name glance --type image --description 'OpenStack
 Image Service'
keystone service-create --name swift --type object-store --description
 'OpenStack Storage Service'
keystone service-create --name keystone --type identity --description
 'OpenStack Identity Service'
keystone service-create --name ec2 --type ec2 --description 'EC2 Service'
```

Each of the services that have been created above will be identified with a unique id which can be obtained from the following command:

```
keystone service-list
+----------------------------------+----------+--------------
+---------------------------+
|                id                |   name   |     type     |
 description         |
+----------------------------------+----------+--------------
+---------------------------+
| 1e93ee6c70f8468c88a5cb1b106753f3 | nova     | compute      | OpenStack
 Compute Service  |
| 28fd92ffe3824004996a3e04e059d875 | ec2      | ec2          | EC2 Service
            |
| 7d4ec192dfa1456996f0f4c47415c7a7 | keystone | identity     | OpenStack
 Identity Service |
| 96f35e1112b143e59d5cd5d0e6a8b22d | swift    | object-store | OpenStack
 Storage Service  |
| f38f4564ff7b4e43a52b2f5c1b75e5fa | volume   | volume       | OpenStack
 Volume Service   |
| fbafab6edcab467bb734380ce6be3561 | glance   | image        | OpenStack Image
 Service    |
+----------------------------------+----------+--------------
+---------------------------+
```

The 'id' will be used in defining the endpoint for that service.

## Creating Endpoints

Create endpoints for each of the services that have been created above.

```
keystone endpoint-create --region region_name --service_id service_id --
publicurl public_url --adminurl admin_url  --internalurl internal_url
```

For creating an endpoint for nova-compute, execute the following command:

```
keystone endpoint-create --region myregion --service_id
 1e93ee6c70f8468c88a5cb1b106753f3 --publicurl 'http://10.10.10.2:8774/
v2/$(tenant_id)s' --adminurl 'http://10.10.10.2:8774/v2/$(tenant_id)s' --
internalurl 'http://10.10.10.2:8774/v2/$(tenant_id)s'
```

For creating an endpoint for nova-volume, execute the following command:

```
keystone endpoint-create --region myregion --service_id
 f38f4564ff7b4e43a52b2f5c1b75e5fa --publicurl 'http://10.10.10.2:8776/
v1/$(tenant_id)s' --adminurl 'http://10.10.10.2:8776/v1/$(tenant_id)s' --
internalurl 'http://10.10.10.2:8776/v1/$(tenant_id)s'
```

For creating an endpoint for glance, execute the following command:

```
keystone endpoint-create --region myregion --service_id
 fbafab6edcab467bb734380ce6be3561 --publicurl 'http://10.10.10.2:9292/v1' --
adminurl 'http://10.10.10.2:9292/v1' --internalurl 'http://10.10.10.2:9292/v1'
```

For creating an endpoint for swift, execute the following command:

```
keystone endpoint-create --region myregion --service_id
 96f35e1112b143e59d5cd5d0e6a8b22d --publicurl 'http://10.10.10.2:8080/v1/AUTH_
$(tenant_id)s' --adminurl 'http://10.10.10.2:8080/v1' --internalurl 'http://
10.10.10.2:8080/v1/AUTH_$(tenant_id)s'
```

For creating an endpoint for keystone, execute the following command:

```
keystone endpoint-create --region myregion --service_id
 7d4ec192dfa1456996f0f4c47415c7a7 --publicurl http://10.10.10.2:5000/v2.0 --
adminurl http://10.10.10.2:35357/v2.0 --internalurl http://10.10.10.2:5000/v2.
0
```

For creating an endpoint for ec2, execute the following command:

```
keystone endpoint-create --region myregion --service_id
 28fd92ffe3824004996a3e04e059d875 --publicurl http://10.10.10.2:8773/services/
Cloud --adminurl http://10.10.10.2:8773/services/Admin --internalurl http://
10.10.10.2:8773/services/Cloud
```

## Glance

Install glance using the following command:

```
sudo apt-get install glance glance-api glance-client glance-common glance-
registry python-glance
```

### Glance Configuration

Glance uses SQLite by default. MySQL and PostgreSQL can also be configured to work with Glance.

Open /etc/glance/glance-api-paste.ini and at the end of the file, edit the following lines:

```
admin_tenant_name = %SERVICE_TENANT_NAME%
admin_user = %SERVICE_USER%
admin_password = %SERVICE_PASSWORD%
```

These values have to be modified as per the configurations made earlier. The admin_tenant_name will be 'service', admin_user will be 'glance' and admin_password is 'glance'.

After editing, the lines should be as follows:

```
admin_tenant_name = service
admin_user = glance
admin_password = glance
```

Now open /etc/glance/glance-registry-paste.ini and make similar changes at the end of the file.

Open the file /etc/glance/glance-registry.conf and edit the line which contains the option "sql_connection =" to this:

```
sql_connection = mysql://glancedbadmin:glancesecret@10.10.10.2/glance
```

In order to tell glance to use keystone for authentication, add the following lines at the end of the file.

```
[paste_deploy]
flavor = keystone
```

Open /etc/glance/glance-api.conf and add the following lines at the end of the document.

```
[paste_deploy]
flavor = keystone
```

Create glance schema in the MySQL database.:

```
sudo glance-manage version_control 0
sudo glance-manage db_sync
```

Restart glance-api and glance-registry after making the above changes.

```
sudo restart glance-api
```

```
sudo restart glance-registry
```

Export the following environment variables.

```
export SERVICE_TOKEN=admin
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin
export OS_AUTH_URL="http://localhost:5000/v2.0/"
export SERVICE_ENDPOINT=http://localhost:35357/v2.0
```

Alternatively, you can add these variables to ~/.bashrc.

To test if glance is setup correectly execute the following command.

```
glance index
```

The above command will not return any output. The output of the last command executed can be known from its return code - echo $?. If the return code is zero, then glance is setup properly and connects with Keystone.

With glance configured properly and using keystone as the authentication mechanism, now we can upload images to glance. This has been explained in detail in "Image Management" chapter.

## Nova

Install nova using the following commands:

```
sudo apt-get install nova-api nova-cert nova-compute nova-compute-kvm nova-
doc nova-network nova-objectstore nova-scheduler nova-volume rabbitmq-server
 novnc nova-consoleauth
```

### Nova Configuration

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
--state_path=/var/lib/nova
--lock_path=/run/lock/nova
--allow_admin_api=true
--use_deprecated_auth=false
--auth_strategy=keystone
--scheduler_driver=nova.scheduler.simple.SimpleScheduler
--s3_host=10.10.10.2
--ec2_host=10.10.10.2
--rabbit_host=10.10.10.2
--cc_host=10.10.10.2
--nova_url=http://10.10.10.2:8774/v1.1/
--routing_source_ip=10.10.10.2
--glance_api_servers=10.10.10.2:9292
--image_service=nova.image.glance.GlanceImageService
--iscsi_ip_prefix=192.168.4
--sql_connection=mysql://novadbadmin:novasecret@10.10.10.2/nova
--ec2_url=http://10.10.10.2:8773/services/Cloud
--keystone_ec2_url=http://10.10.10.2:5000/v2.0/ec2tokens
--api_paste_config=/etc/nova/api-paste.ini
--libvirt_type=kvm
--libvirt_use_virtio_for_bridges=true
--start_guests_on_host_boot=true
--resume_guests_state_on_host_boot=true
# vnc specific configuration
--novnc_enabled=true
--novncproxy_base_url=http://10.10.10.2:6080/vnc_auto.html
--vncserver_proxyclient_address=10.10.10.2
--vncserver_listen=10.10.10.2
# network specific settings
--network_manager=nova.network.manager.FlatDHCPManager
--public_interface=eth0
--flat_interface=eth1
--flat_network_bridge=br100
--fixed_range=192.168.4.1/27
--floating_range=10.10.10.2/27
--network_size=32
--flat_network_dhcp_start=192.168.4.33
--flat_injected=False
--force_dhcp_release
--iscsi_helper=tgtadm
--connection_type=libvirt
--root_helper=sudo nova-rootwrap
--verbose
```

Create a Physical Volume.

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

Create a Volume Group named nova-volumes.

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

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

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

Open /etc/nova/api-paste.ini and at the end of the file, edit the following lines:

```
admin_tenant_name = %SERVICE_TENANT_NAME%
admin_user = %SERVICE_USER%
admin_password = %SERVICE_PASSWORD%
```

These values have to be modified conforming to configurations made earlier. The admin_tenant_name will be 'service', admin_user will be 'nova' and admin_password is 'nova'.

After editing, the lines should be as follows:

```
admin_tenant_name = service
admin_user = nova
admin_password = nova
```

Create nova schema in the MySQL database.

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

Provide a range of IPs to be associated to the instances.

```
sudo nova-manage network create private --fixed_range_v4=192.168.4.32/27 --
num_networks=1 --bridge=br100 --bridge_interface=eth1 --network_size=32
```

Export the following environment variables.

```
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin
export OS_AUTH_URL="http://localhost:5000/v2.0/"
```

Restart nova 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 nova-consoleauth;
```

To test if nova is setup correctly run the following command.

```
sudo nova-manage service list
Binary           Host              Zone              Status     State
 Updated_At
nova-network     server1           nova              enabled    :-)
 2012-04-20 08:58:43
nova-scheduler   server1           nova              enabled    :-)
 2012-04-20 08:58:44
nova-volume      server1           nova              enabled    :-)
 2012-04-20 08:58:44
nova-compute     server1           nova              enabled    :-)
 2012-04-20 08:58:45
nova-cert        server1           nova              enabled    :-)
 2012-04-20 08:58:43
```

If the output is similar to the above with all components happy, your setup is ready to be used.

## OpenStack Dashboard

Install OpenStack Dashboard by executing the following command:

```
sudo apt-get install openstack-dashboard
```

Restart apache with the following command:

```
sudo service apache2 restart
```

Open a browser and enter IP address of the server1. You should see the OpenStack Dashboard login prompt. Login with username 'admin' and password 'admin'. From the dashboard, you can create keypairs, create/edit security groups, raise new instances, attach volumes etc. which are explained in "OpenStack Dashboard" chapter.

## Swift

### Swift Installation

The primary components are the proxy, account, container and object servers.

```
sudo apt-get install swift swift-proxy swift-account swift-container swift-
object
```

Other components that might be xfsprogs (for dealing with XFS filesystem), python.pastedeploy (for keystone access), curl (to test swift).

```
sudo apt-get install xfsprogs curl python-pastedeploy
```

### Swift Storage Backends

There are two methods one can try to create/prepare the storage backend. One is to use an existing partition/volume as the storage device. The other is to create a loopback file and use it as the storage device. Use the appropriate method as per your setup.

### Partition as a storage device

If you had set aside a partition for Swift during the installation of the OS, you can use it directly. If you have an unused/unpartitioned physical partition (e.g. /dev/sdb3), you have to format it to xfs filesystem using parted or fdisk and use it as the backend. You need to specify the mount point in /etc/fstab.

```
CAUTION: Replace /dev/sdb to your appropriate device. I'm assuming that there
 is an unused/un-formatted partition section in /dev/sdb
```

```
sudo fdisk /dev/sdb

    Type n for new partition
    Type e for extended partion
    Choose appropriate partition number ( or go with the default )
    Choose first and last sectors to set the hard disk size (or go with
 defaults)
    Note that 83 is the partition type number for Linux
    Type w to write changes to the disk
```

This would have created a partition (something like /dev/sdb3) that we can now format to XFS filesystem. Do 'sudo fdisk -l' in the terminal to view and verify the partion table. Find the partition Make sure that the one that you want to use is listed there. This would work only if you have xfsprogs installed.

```
sudo mkfs.xfs -i size=1024 /dev/sdb3
sudo tune2fs -l /dev/sdb3 |grep -i inode
```

Create a directory /mnt/swift_backend that can be used as a mount point to the partion tha we created.

```
sudo mkdir /mnt/swift_backend
```

Now edit /etc/fstab and append the following line to mount the partition automatically everytime the system restarts.

```
/dev/sdb3 /mnt/swift_backend xfs noatime,nodiratime,nobarrier,logbufs=8 0 0
```

### Loopback File as a storage device

We create a zero filled file for use as a loopback device for the Swift storage backend. Here we use the disk copy command to create a file named swift-disk and allocate a million 1KiB blocks (976.56 MiB) to it. So we have a loopback disk of approximately 1GiB. We can increase this size by modifying the seek value. The disk is then formated to XFS filesystem. The file command can be used to verify if it worked.

```
sudo dd if=/dev/zero of=/srv/swift-disk bs=1024 count=0 seek=1000000
sudo mkfs.xfs -i size=1024 /srv/swift-disk
file /srv/swift-disk
swift-disk1: SGI XFS filesystem data (blksz 4096, inosz 1024, v2 dirs)
```

Create a directory /mnt/swift_backend that can be used as a mount point to the partion tha we created.

```
sudo mkdir /mnt/swift_backend
```

Make it mount on boot by appending this to /etc/fstab.

```
/srv/swift-disk /mnt/swift_backend xfs loop,noatime,nodiratime,nobarrier,
logbufs=8 0 0
```

### Using the backend

Now before mounting the backend that will be used, create some nodes to be used as storage devices and set ownership to 'swift' user and group.

```
sudo mount /mnt/swift_backend
pushd /mnt/swift_backend
sudo mkdir node1 node2 node3 node4
popd
```

```
sudo chown swift.swift /mnt/swift_backend/*
```

```
for i in {1..4}; do sudo ln -s /mnt/swift_backend/node$i /srv/node$i; done;
```

```
sudo mkdir -p /etc/swift/account-server /etc/swift/container-server /etc/
swift/object-server /srv/node1/device /srv/node2/device /srv/node3/device /
srv/node4/device
sudo mkdir /run/swift
sudo chown -L -R swift.swift /etc/swift /srv/node[1-4]/ /run/swift
```

Append the following lines in /etc/rc.local just before "exit 0";. This will be run everytime the system starts.

```
sudo mkdir /run/swift
sudo chown swift.swift /run/swift
```

## Configure Rsync

Rsync is responsible for maintaining object replicas. It is used by various swift services to maintain consistency of objects and perform updation operations. It is configured for all the storage nodes.

Set RSYNC_ENABLE=true in /etc/default/rsync.

Modify /etc/rsyncd.conf as follows:

```
# General stuff
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /run/rsyncd.pid
address = 127.0.0.1

# Account Server replication settings

[account6012]
max connections = 25
```

```
path = /srv/node1/
read only = false
lock file = /run/lock/account6012.lock

[account6022]
max connections = 25
path = /srv/node2/
read only = false
lock file = /run/lock/account6022.lock

[account6032]
max connections = 25
path = /srv/node3/
read only = false
lock file = /run/lock/account6032.lock

[account6042]
max connections = 25
path = /srv/node4/
read only = false
lock file = /run/lock/account6042.lock

# Container server replication settings

[container6011]
max connections = 25
path = /srv/node1/
read only = false
lock file = /run/lock/container6011.lock

[container6021]
max connections = 25
path = /srv/node2/
read only = false
lock file = /run/lock/container6021.lock

[container6031]
max connections = 25
path = /srv/node3/
read only = false
lock file = /run/lock/container6031.lock

[container6041]
max connections = 25
path = /srv/node4/
read only = false
lock file = /run/lock/container6041.lock

# Object Server replication settings

[object6010]
max connections = 25
path = /srv/node1/
read only = false
lock file = /run/lock/object6010.lock

[object6020]
max connections = 25
path = /srv/node2/
read only = false
```

```
lock file = /run/lock/object6020.lock

[object6030]
max connections = 25
path = /srv/node3/
read only = false
lock file = /run/lock/object6030.lock

[object6040]
max connections = 25
path = /srv/node4/
read only = false
lock file = /run/lock/object6040.lock
```

Restart rsync.

```
sudo service rsync restart
```

## Configure Swift Components

General server configuration options can be found in http://swift.openstack.org/deployment_guide.html. If the swift-doc package is installed it can also be viewed in the /usr/share/doc/swift-doc/html directory. Python uses paste.deploy to manage configuration. Default configuration options are set in the [DEFAULT] section, and any options specified there can be overridden in any of the other sections BUT ONLY BY USING THE SYNTAX set option_name = value.

Here is a sample paste.deploy configuration for reference:

```
[DEFAULT]
name1 = globalvalue
name2 = globalvalue
name3 = globalvalue
set name4 = globalvalue

[pipeline:main]
pipeline = myapp

[app:myapp]
use = egg:mypkg#myapp
name2 = localvalue
set name3 = localvalue
set name5 = localvalue
name6 = localvalue
```

Create and edit /etc/swift/swift.conf and add the following lines to it:

```
[swift-hash]
# random unique string that can never change (DO NOT LOSE). I'm using
 03c9f48da2229770.
# od -t x8 -N 8 -A n < /dev/random
# The above command can be used to generate random a string.
swift_hash_path_suffix = 03c9f48da2229770
```

You will need the random string when you add more nodes to the setup. So never lose the string.

You can generate a random string by running the following command:

```
od -t x8 -N 8 -A n < /dev/random
```

### Configure Swift Proxy Server

Proxy server acts as the gatekeeper to swift. It takes the responsibility of authenticating the user. Authentication verifies that a request actually comes from who it says it does. Authorization verifies the 'who' has access to the resource(s) the request wants. Authorization is done by identity services like keystone. Create and edit /etc/swift/proxy-server.conf and add the following lines.

```
[DEFAULT]
bind_port = 8080
user = swift
swift_dir = /etc/swift

[pipeline:main]
# Order of execution of modules defined below
pipeline = catch_errors healthcheck cache authtoken keystone proxy-server

[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true
set log_name = swift-proxy
set log_facility = LOG_LOCAL0
set log_level = INFO
set access_log_name = swift-proxy
set access_log_facility = SYSLOG
set access_log_level = INFO
set log_headers = True
account_autocreate = True

[filter:healthcheck]
use = egg:swift#healthcheck

[filter:catch_errors]
use = egg:swift#catch_errors

[filter:cache]
use = egg:swift#memcache
set log_name = cache

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_protocol = http
auth_host = 127.0.0.1
auth_port = 35357
auth_token = admin
service_protocol = http
service_host = 127.0.0.1
service_port = 5000
admin_token = admin
admin_tenant_name = service
admin_user = swift
admin_password = swift
delay_auth_decision = 0
```

```
[filter:keystone]
paste.filter_factory = keystone.middleware.swift_auth:filter_factory
operator_roles = admin, swiftoperator
is_admin = true
```

Note: You can find sample configuration files at the "etc" directory in the source. Some documentation can be found under "/usr/share/doc/swift-doc/html" if you had installed the swift-doc package using apt-get.

### Configure Swift Account Server

The default swift account server configuration is /etc/swift/account-server.conf.

```
[DEFAULT]
bind_ip = 0.0.0.0
workers = 2

[pipeline:main]
pipeline = account-server

[app:account-server]
use = egg:swift#account

[account-replicator]

[account-auditor]

[account-reaper]
```

Account server configuration files are also looked up under /etc/swift/account-server.conf. Here we can create several account server configuration files each of which would correspond to a device under /srv. The files can be named 1.conf, 2.conf and so on. Here are the contents of /etc/swift/account-server/1.conf:

```
[DEFAULT]
devices = /srv/node1
mount_check = false
bind_port = 6012
user = swift
log_facility = LOG_LOCAL2

[pipeline:main]
pipeline = account-server

[app:account-server]
use = egg:swift#account

[account-replicator]
vm_test_mode = no

[account-auditor]

[account-reaper]
```

For the other devices, (/srv/node2, /srv/node3, /srv/node4), we create 2.conf, 3.conf and 4.conf. So we make three more copies of 1.conf and set unique bind ports for the rest of the nodes (6022, 6032 and 6042) and different local log values (LOG_LOCAL3, LOG_LOCAL4, LOG_LOCAL5).

```
sudo cp /etc/swift/account-server/1.conf /etc/swift/account-server/2.conf
sudo cp /etc/swift/account-server/1.conf /etc/swift/account-server/3.conf
sudo cp /etc/swift/account-server/1.conf /etc/swift/account-server/4.conf
sudo sed -i 's/6012/6022/g;s/LOCAL2/LOCAL3/g;s/node1/node2/g' /etc/swift/
account-server/2.conf
sudo sed -i 's/6012/6032/g;s/LOCAL2/LOCAL4/g;s/node1/node3/g' /etc/swift/
account-server/3.conf
sudo sed -i 's/6012/6042/g;s/LOCAL2/LOCAL5/g;s/node1/node4/g' /etc/swift/
account-server/4.conf
```

### Configure Swift Container Server

The default swift container server configuration is /etc/swift/container-server.conf.

```
[DEFAULT]
bind_ip = 0.0.0.0
workers = 2

[pipeline:main]
pipeline = container-server

[app:container-server]
use = egg:swift#container

[container-replicator]

[container-updater]

[container-auditor]

[container-sync]
```

Container server configuration files are also looked up under /etc/swift/container-server.conf. Here we can create several container server configuration files each of which would correspond to a device under /srv. The files can be named 1.conf, 2.conf and so on. Here are the contents of /etc/swift/container-server/1.conf:

```
[DEFAULT]
devices = /srv/node1
mount_check = false
bind_port = 6011
user = swift
log_facility = LOG_LOCAL2

[pipeline:main]
pipeline = container-server

[app:container-server]
use = egg:swift#container

[container-replicator]
vm_test_mode = no

[container-updater]
```

```
[container-auditor]

[container-sync]
```

For the other devices, (/srv/node2, /srv/node3, /srv/node4), we create 2.conf, 3.conf and 4.conf. So we make three more copies of 1.conf and set unique bind ports for the rest of the nodes (6021, 6031 and 6041) and different local log values (LOG_LOCAL3, LOG_LOCAL4, LOG_LOCAL5).

### Configure Swift Object Server

The default swift object server configuration is /etc/swift/object-server.conf.

```
[DEFAULT]
bind_ip = 0.0.0.0
workers = 2

[pipeline:main]
pipeline = object-server

[app:object-server]
use = egg:swift#object

[object-replicator]

[object-updater]

[object-auditor]
```

Object server configuration files are also looked up under /etc/swift/object-server.conf. Here we can create several object server configuration files each of which would correspond to a device under /srv. The files can be named 1.conf, 2.conf and so on. Here are the contents of /etc/swift/object-server/1.conf:

```
[DEFAULT]
devices = /srv/node1
mount_check = false
bind_port = 6010
user = swift
log_facility = LOG_LOCAL2

[pipeline:main]
pipeline = object-server

[app:object-server]
use = egg:swift#object

[object-replicator]
vm_test_mode = no

[object-updater]

[object-auditor]
```

For the other devices, (/srv/node2, /srv/node3, /srv/node4), we create 2.conf, 3.conf and 4.conf. So we make three more copies of 1.conf and set unique bind ports for the rest of the nodes (6020, 6030 and 6040) and different local log values (LOG_LOCAL3, LOG_LOCAL4, LOG_LOCAL5).

### Configure Swift Rings

Ring is an important component of swift. It maintains the information about the physical location of objects, their replicas and devices. We now create the ring builder files corresponding to object service, container service and account service.

NOTE: We need to be in the /etc/swift directory when executing the following commands.

```
pushd /etc/swift
sudo swift-ring-builder object.builder create 18 3 1
sudo swift-ring-builder container.builder create 18 3 1
sudo swift-ring-builder account.builder create 18 3 1
```

The numbers indicate the desired number of partitions, replicas and the time in hours to restrict moving a partition more than once. See the man page for swift-ring-builder for more information.

Now we add zones and balance the rings. The syntax is as follows:

```
swift-ring-builder <builder_file> add <zone>-<ip_address>:<port>/<device>
 <weight>
```

Execute the following commands to add the zones and rebalance the ring.

```
sudo swift-ring-builder object.builder add z1-127.0.0.1:6010/device 1
sudo swift-ring-builder object.builder add z2-127.0.0.1:6020/device 1
sudo swift-ring-builder object.builder add z3-127.0.0.1:6030/device 1
sudo swift-ring-builder object.builder add z4-127.0.0.1:6040/device 1
sudo swift-ring-builder object.builder rebalance
sudo swift-ring-builder container.builder add z1-127.0.0.1:6011/device 1
sudo swift-ring-builder container.builder add z2-127.0.0.1:6021/device 1
sudo swift-ring-builder container.builder add z3-127.0.0.1:6031/device 1
sudo swift-ring-builder container.builder add z4-127.0.0.1:6041/device 1
sudo swift-ring-builder container.builder rebalance
sudo swift-ring-builder account.builder add z1-127.0.0.1:6012/device 1
sudo swift-ring-builder account.builder add z2-127.0.0.1:6022/device 1
sudo swift-ring-builder account.builder add z3-127.0.0.1:6032/device 1
sudo swift-ring-builder account.builder add z4-127.0.0.1:6042/device 1
sudo swift-ring-builder account.builder rebalance
```

## Starting Swift services

To start swift and the REST API, run the following commands.

```
sudo swift-init main start
sudo swift-init rest start
```

## Testing Swift

Swift can be tested using the swift command or the dashboard web interface (Horizon). Firstly, make sure that the ownership for /etc/swift directory is set to swift.swift.

```
sudo chown -R swift.swift /etc/swift
```

Then run the following command and verify if you get the appropriate account information. The number of containers and objects stored within are displayed as well.

```
swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0/ -U service:swift -K swift stat
StorageURL: http://127.0.0.1:8080/v1/AUTH_c7970080576646c6959ee35970cf3199
Auth Token: ba9df200a92d4a5088dcd6b7dcc19c0d
   Account: AUTH_c7970080576646c6959ee35970cf3199
Containers: 1
   Objects: 1
     Bytes: 77
Accept-Ranges: bytes
X-Trans-Id: tx11c64e218f984749bc3ec37ea46280ee
```

## Server2

This server runs only nova-compute service.

## BaseOS

Install 64 bit version of Ubuntu server 12.04

## Network Configuration

Install bridge-utils:

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

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 10.10.10.3
  netmask 255.255.255.0
  broadcast 10.10.10.255
  gateway 10.10.10.1
  dns-nameservers 10.10.8.3

auto eth1
iface eth1 inet static
  address 192.168.3.2
  netmask 255.255.255.0
  network 192.168.3.0
  broadcast 192.168.3.255
```

Restart the network.

```
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 service ntp restart
```

## Nova Components (nova-compute alone)

Install the nova-components and dependencies.

```

sudo apt-get install nova-compute
```

Edit the /etc/nova/nova.conf file to look like this. This file is identical 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
--state_path=/var/lib/nova
--lock_path=/run/lock/nova
--allow_admin_api=true
--use_deprecated_auth=false
--auth_strategy=keystone
--scheduler_driver=nova.scheduler.simple.SimpleScheduler
--s3_host=10.10.10.2
--ec2_host=10.10.10.2
--rabbit_host=10.10.10.2
--cc_host=10.10.10.2
--nova_url=http://10.10.10.2:8774/v1.1/
--routing_source_ip=10.10.10.2
--glance_api_servers=10.10.10.2:9292
--image_service=nova.image.glance.GlanceImageService
--iscsi_ip_prefix=192.168.4
--sql_connection=mysql://novadbadmin:novasecret@10.10.10.2/nova
--ec2_url=http://10.10.10.2:8773/services/Cloud
--keystone_ec2_url=http://10.10.10.2:5000/v2.0/ec2tokens
--api_paste_config=/etc/nova/api-paste.ini
--libvirt_type=kvm
--libvirt_use_virtio_for_bridges=true
--start_guests_on_host_boot=true
--resume_guests_state_on_host_boot=true
# vnc specific configuration
--novnc_enabled=true
--novncproxy_base_url=http://10.10.10.2:6080/vnc_auto.html
--vncserver_proxyclient_address=10.10.10.2
--vncserver_listen=10.10.10.2
# network specific settings
--network_manager=nova.network.manager.FlatDHCPManager
--public_interface=eth0
--flat_interface=eth1
--flat_network_bridge=br100
--fixed_range=192.168.4.1/27
--floating_range=10.10.10.2/27
--network_size=32
```

```
--flat_network_dhcp_start=192.168.4.33
--flat_injected=False
--force_dhcp_release
--iscsi_helper=tgtadm
--connection_type=libvirt
--root_helper=sudo nova-rootwrap
--verbose
```

Restart nova-compute on Server2.

```
sudo service restart nova-compute
```

Check if the second compute node (Server2) is detected by running:

```
sudo nova-manage service list
```

If you see an output similar to the following, it means that the set up is ready to be used.

```

sudo nova-manage service list
Binary           Host                 Zone              Status     State
 Updated_At
nova-network     server1              nova              enabled    :-)
 2012-04-20 08:58:43
nova-scheduler   server1              nova              enabled    :-)
 2012-04-20 08:58:44
nova-volume      server1              nova              enabled    :-)
 2012-04-20 08:58:44
nova-compute     server1              nova              enabled    :-)
 2012-04-20 08:58:45
nova-cert        server1              nova              enabled    :-)
 2012-04-20 08:58:43
nova-compute     server2              nova              enabled    :-)
 2012-04-21 10:22:27
```

# Client1

## BaseOS

Install 64-bit version of Ubuntu 12.04 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 10.10.8.3
```

## NTP Client

Install NTP package.

```
sudo apt-get install -y 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 service ntp restart
```

## Client Tools

As mentioned above, this is a desktop installation of Ubuntu 12.04 to be used for tasks such as bundling of images. It will also be used for managing the cloud infrastructure using nova, glance and swift commandline tools.

Install the required command line tools with the following command:

```
sudo apt-get install python-novaclient glance-client swift
```

Install qemu-kvm

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

Export the following environment variables or add them to your ~/.bashrc.

```

export SERVICE_TOKEN=admin
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin
export OS_AUTH_URL="http://10.10.10.2:5000/v2.0/"
export SERVICE_ENDPOINT=http://10.10.10.2:35357/v2.0
```

Execute nova and glance commands to check the connectivity to OpenStack setup.

```

nova list
+--------------------------------------+------------+--------
+----------------------+
|                  ID                  |    Name    | Status |      Networks
      |
+--------------------------------------+------------+--------
+----------------------+
| 25ee9230-6bb5-4eca-8808-e6b4e0348362 | myinstance | ACTIVE | private=192.
168.4.35 |
| c939cb2c-e662-46e5-bc31-453007442cf9 | myinstance1| ACTIVE | private=192.
168.4.36 |
+--------------------------------------+------------+--------
+----------------------+
glance index
ID                                   Name          Disk     Container Size
                                                   Format   Format
------------------------------------ ------------------------------
 ----------------
65b9f8e1-cde8-40e7-93e3-0866becfb9d4 windows       qcow2    ovf
 7580745728
f147e666-990c-47e2-9caa-a5a21470cc4e debian        qcow2    ovf
 932904960
f3a8e689-02ed-460f-a587-dc868576228f opensuse      qcow2    ovf
 1072300032
aa362fd9-7c28-480b-845c-85a5c38ccd86 centoscli     qcow2    ovf
 1611530240
49f0ec2b-26dd-4644-adcc-2ce047e281c5 ubuntuimage   qcow2    ovf
 1471807488
```

## OpenStack Dashboard

Start a browser and type the ip address of Server1 i.e, http://10.10.10.2. You should see the dashboard login screen. Login with the credentials username - admin and password - admin to manage the OpenStack setup.

# 3. Image Management

## 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.

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 popular 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 creating the Linux images of different distributions is largely the same with a few minor differences, which is explained below.

In all the 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 generate disk images that represent a disk without any partitions.

## Creating a Linux Image

The first step would be to create an 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 qcow2 server.img 5G
```

### OS Installation

Download the iso file of the Linux distribution you want to install in the image. For Ubuntu, you can download the iso from http://releases.ubuntu.com using 'wget' or with the help of a browser

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-12.04-server-amd64.iso -drive file=server.img,
if=virtio,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 creation of Linux images , create a single ext4 partition mounted on a swap partition.

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

```
sudo kvm -m 256 -drive file=server.img,if=virtio,index=0 -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.

## Ubuntu

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

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

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
```

## Fedora

```
yum update
```

```
yum install openssh-server
```

```
chkconfig sshd on
```

Edit the file /etc/sysconfig/network-scripts/ifcfg-eth0 to look like this

```
DEVICE="eth0"
BOOTPROTO=dhcp
NM_CONTROLLED="yes"
ONBOOT="yes"
```

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.

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 acpiphp
# 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 "************************"
```

## OpenSUSE

Select ssh server, curl and other packages needed.

Install ssh server.

```
zypper install openssh
```

Install curl.

```
zypper install curl
```

For ssh key injection into the instance use the following steps:

Create a file /etc/init.d/sshkey and add the following lines

```
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 "************************"
```

Change the permissions for the file.

```
chmod 755 /etc/init.d/sshkey
```

Configure the service to start automatically while booting.

```
chkconfig sshkey on
```

Configure the firewall (not iptables) using the following command and allow ssh service

```
yast2
```

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.

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

## Debian

Select SSH server, Curl and other packages needed.

Do the necessary changes needed for the image. For key injection add the following lines in the file /etc/rc.local.

```
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 "************************"
```

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.

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

## CentOS 6 and RHEL 6

Select SSH server, Curl and other packages needed.

Do the necessary changes needed for the image. For key injection add the following lines in the file /etc/rc.local.

```
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 "************************"
```

Edit the file /etc/sysconfig/network-scripts/ifcfg-eth0 to look like this

```
DEVICE="eth0"
BOOTPROTO=dhcp
NM_CONTROLLED="yes"
ONBOOT="yes"
```

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.

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

## Uploading the Linux image

Upload the image

```
glance add name="<Image name>" is_public=true container_format=ovf
 disk_format=qcow2 < <filename>.img
```

# Creating a Windows Image

The first step would be to create an 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 qcow2 windowsserver.img 20G
```

## OS Installation

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 iso image 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 executing:

```
sudo kvm -m 1024 -cdrom windows2008.iso -drive file=windowsserver1.img,if=
virtio -boot d -drive file=virtio-win-0.1-22.iso,index=3,media=cdrom  -device
 virtio-net-pci -net nic -net user -nographic -vnc :5
```

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 by browsing the secondary CDROM in which the virtio driver disk is loaded

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.

## Uploading the Windows image

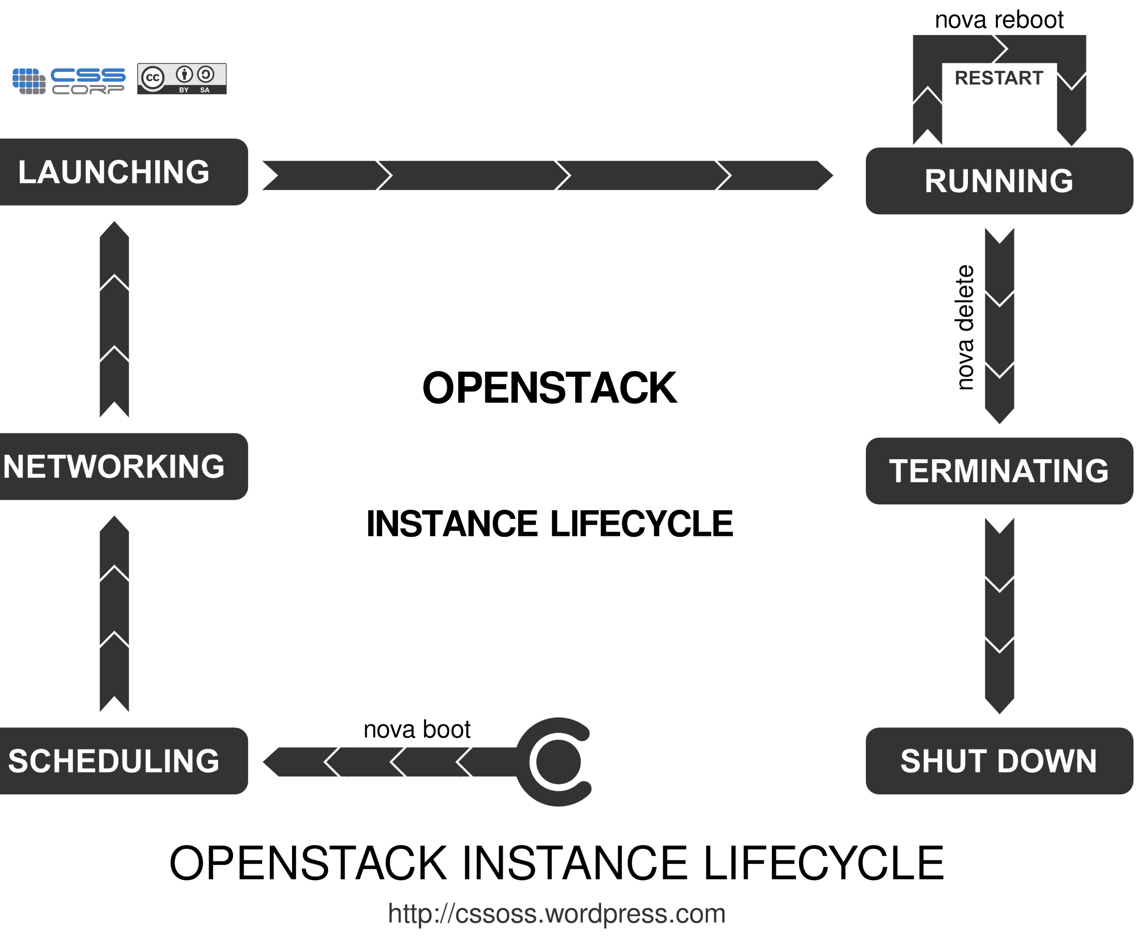
Shut-down the VM and upload the image to OpenStack

```
glance add name="windows" is_public=true container_format=ovf disk_format=
qcow2 < windowsserver.img
```

# 4. Instance Management

## 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 actions is 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:



The following interfaces can be used for managing instances in nova.

- Nova commands
- Custom applications developed using Nova APIs
- Custom applications developed using EC2 APIs

## Openstack Command Line Tools

Nova has a bunch of command line tools to manage the OpenStack setup. These commands help you manage images, instances, storage, networking etc. A few commands related to managing the instances are given below.

## Creation of Key Pairs

OpenStack services are authenticated and authorized against keystone identity server. Keystone provides a token and a service catolog containing information about the endpoints of services to which a user is authorized. Each user has a token and service catalog created for them. This can be downloaded from the OpenStack Dashboard.

You will also need to generate a keypair consisting of private key/public key to be able to launch instances on OpenStack. These keys are injected into the instances to make password-less SSH access to the instance. 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.

```
ssh-keygen
cd .ssh
nova keypair-add --pub_key id_rsa.pub mykey
```

This creates a new keypair called mykey. The private key id_rsa is saved locally in ~/.ssh which can be used to connect to an instance launched using mykey as the keypair. You can see the available keypairs with nova keypair-list command.

```
nova keypair-list
+-------+-------------------------------------------------+
|  Name |                   Fingerprint                   |
+-------+-------------------------------------------------+
| mykey  | b0:18:32:fa:4e:d4:3c:1b:c4:6c:dd:cb:53:29:13:82 |
| mykey2 | b0:18:32:fa:4e:d4:3c:1b:c4:6c:dd:cb:53:29:13:82 |
+-------+-------------------------------------------------+
```

Also while executing 'ssh-keygen' you can specify a custom location and custom file names for the keypairs that you want to create.

To delete an existing keypair:

```
nova keypair-delete mykey2
```

## Launch and manage instances

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

```
$ nova boot --flavor 1 --image 9bab7ce7-7523-4d37-831f-c18fbc5cb543 --key_name
 mykey myinstance
+-------------------------------------+--------------------------------------+
|               Property              |                 Value                |
+-------------------------------------+--------------------------------------+
| OS-DCF:diskConfig                   | MANUAL                               |
| OS-EXT-SRV-ATTR:host                | None                                 |
| OS-EXT-SRV-ATTR:hypervisor_hostname | None                                 |
| OS-EXT-SRV-ATTR:instance_name       | instance-00000002                    |
| OS-EXT-STS:power_state              | 0                                    |
| OS-EXT-STS:task_state               | scheduling                           |
| OS-EXT-STS:vm_state                 | building                             |
| accessIPv4                          |                                      |
| accessIPv6                          |                                      |
| adminPass                           | FaUPM6EEBT8F                         |
| config_drive                        |                                      |
| created                             | 2012-05-02T19:29:59Z                 |
| flavor                              | m1.tiny                              |
| hostId                              |                                      |
| id                                  | 25ee9230-6bb5-4eca-8808-e6b4e0348362 |
| image                               | ubuntu                             |
| key_name                            | mykey                                |
| metadata                            | {}                                   |
| name                                | myinstance                           |
| progress                            | 0                                    |
| status                              | BUILD                                |
| tenant_id                           | 24da687e5d844657996bd5e93d06cb89     |
| updated                             | 2012-05-02T19:29:59Z                 |
| user_id                             | 2b64e5ed949145ce92e1fb47224740fe     |
+-------------------------------------+--------------------------------------+
```

```
$ nova list
+--------------------------------------+------------+--------
+----------------------+
|                  ID                  |    Name    | Status |      Networks
      |
+--------------------------------------+------------+--------
+----------------------+
| 25ee9230-6bb5-4eca-8808-e6b4e0348362 | myinstance | ACTIVE | private=192.
168.4.35 |
| c939cb2c-e662-46e5-bc31-453007442cf9 | myinstance1| ACTIVE | private=192.
168.4.36 |
+--------------------------------------+------------+--------
+----------------------+
```

```
$ nova reboot 25ee9230-6bb5-4eca-8808-e6b4e0348362
$ nova list
+--------------------------------------+------------+--------
+----------------------+
|                  ID                  |    Name    | Status |      Networks
      |
+--------------------------------------+------------+--------
+----------------------+
| 25ee9230-6bb5-4eca-8808-e6b4e0348362 | myinstance | REBOOT | private=192.
168.4.35 |
| c939cb2c-e662-46e5-bc31-453007442cf9 | myinstance1| ACTIVE | private=192.
168.4.34 |
+--------------------------------------+------------+--------
+----------------------+
$ nova delete 25ee9230-6bb5-4eca-8808-e6b4e0348362
$ nova list
+--------------------------------------+------------+--------
+----------------------+
|                  ID                  |   Name     | Status |     Networks
      |
+--------------------------------------+------------+--------
+----------------------+
| c939cb2c-e662-46e5-bc31-453007442cf9 | myinstance1| ACTIVE | private=192.
168.4.34 |
+--------------------------------------+------------+--------
+----------------------+
```

```
$ nova console-log myinstance
```

For passwordless ssh access to the instance:

```
ssh -i <private_key> username@<ip_address>
```

VM type has implications for harddisk size, amount of RAM and number of CPUs allocated to the instance. Check the VM types available.

```
nova flavor-list
```

New flavours can be created with the command.

```
sudo nova-manage flavor create <args> [options]
```

A flavour can be deleted with the command.

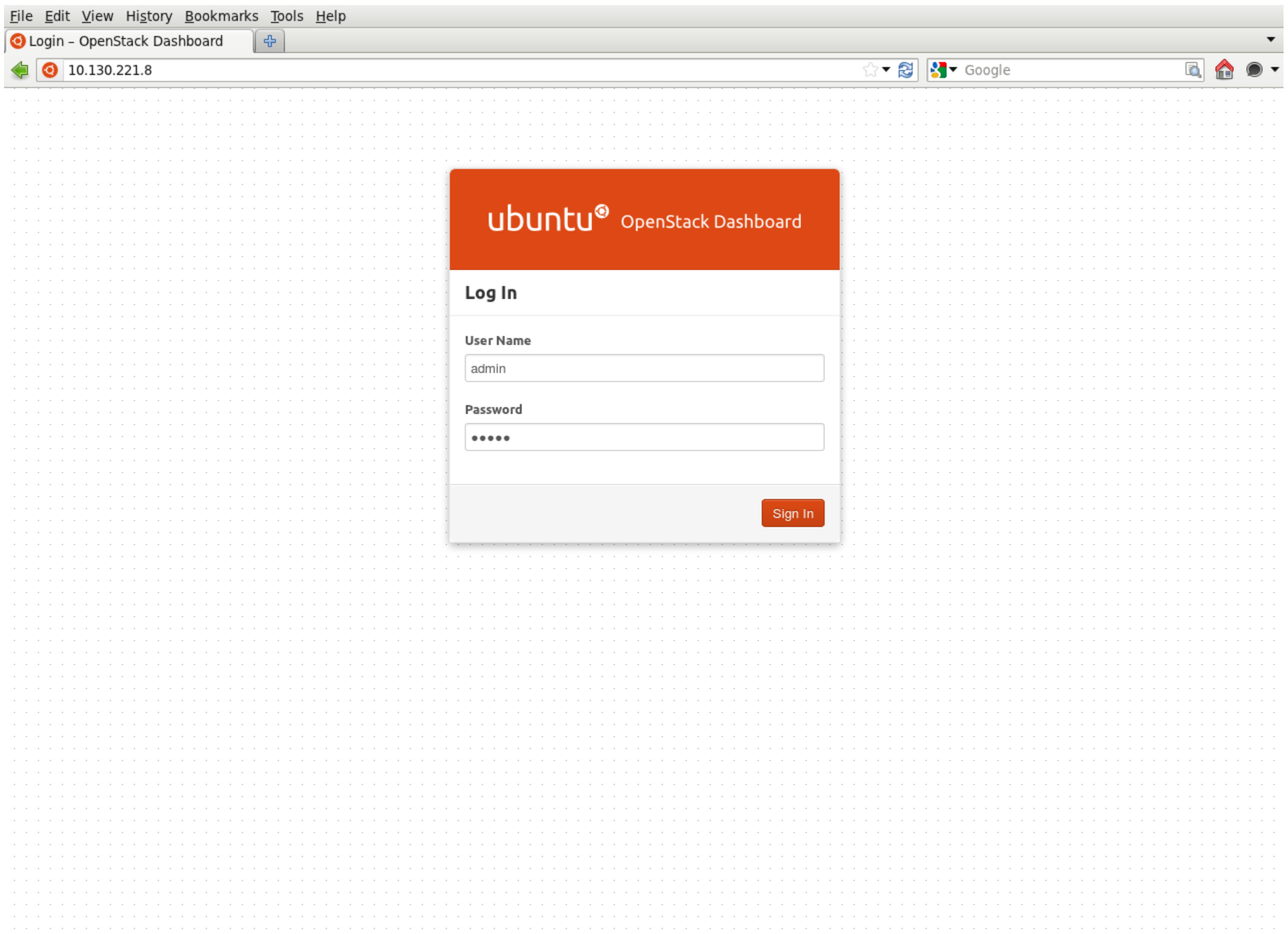
```
sudo nova-manage flavor delete <args> [options]
```

# 5. OpenStack Dashboard (Horizon)

Using the OpenStack Dashboard, one can manage various OpenStack services. It may be used to manage instances and images, create keypairs, attach volumes to instances, manipulate Swift containers etc. The OpenStack Dashboard is accessible via http://<ip_address>

## Login

Login to the dashboard with username "admin" and password "admin".

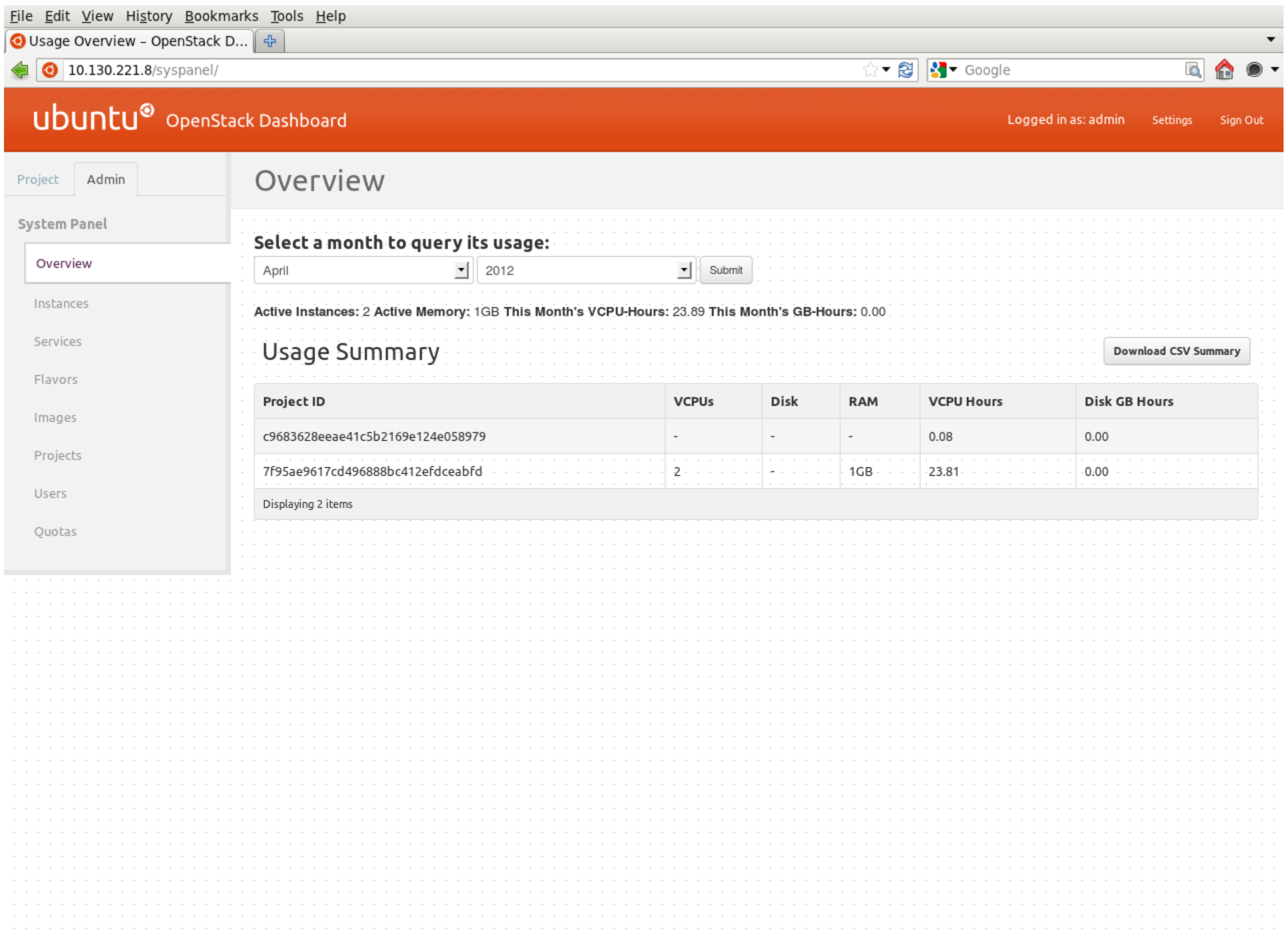

File Edit View History Bookmarks Tools Help
Login - OpenStack Dashboard
10.130.221.8
Google
ubuntu OpenStack Dashboard
Log In
Sign In

User Name: admin

Password: •••••

## User Overview

After logging, depending on the access privileges, the user is allowed access to specific projects. The below is an overview page for a project belonging to the 'admin' user. One can view and download some basic usage metric reports here.



Overview

Select a month to query its usage:

April 2012

**Active Instances:** 2 **Active Memory:** 1GB **This Month's VCPU-Hours:** 23.89 **This Month's GB-Hours:** 0.00

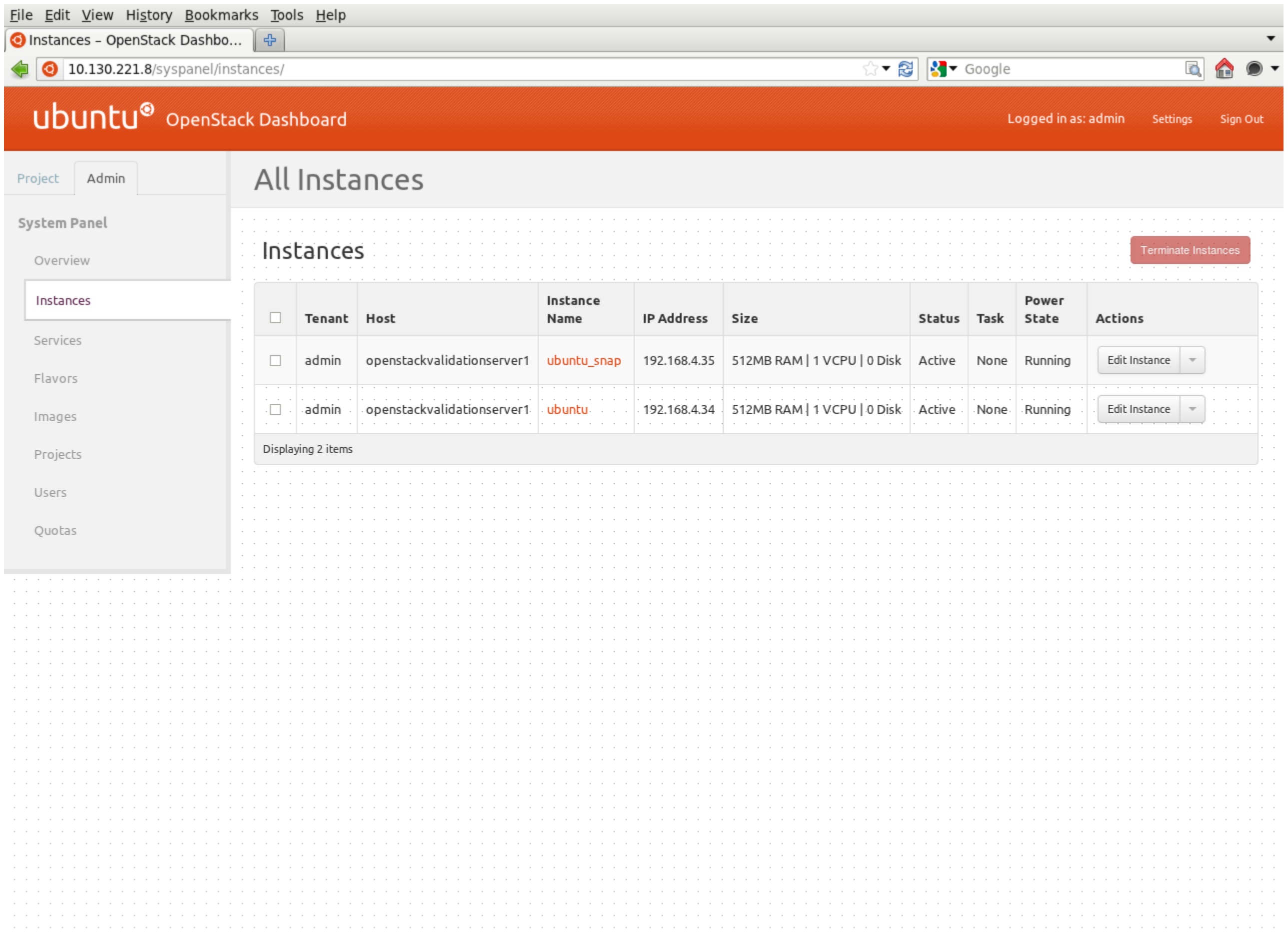
Usage Summary

| Project ID | VCPUs | Disk | RAM | VCPU Hours | Disk GB Hours |
|---|---|---|---|---|---|
| c9683628eeae41c5b2169e124e058979 | - | - | - | 0.08 | 0.00 |
| 7f95ae9617cd496888bc412efdceabfd | 2 | - | 1GB | 23.81 | 0.00 |

Displaying 2 items

## Instances

The page lists currently running instances belonging to the user 'admin'. From this page, one can terminate, pause, reboot any running instances, connect to vnc console of the instance etc.

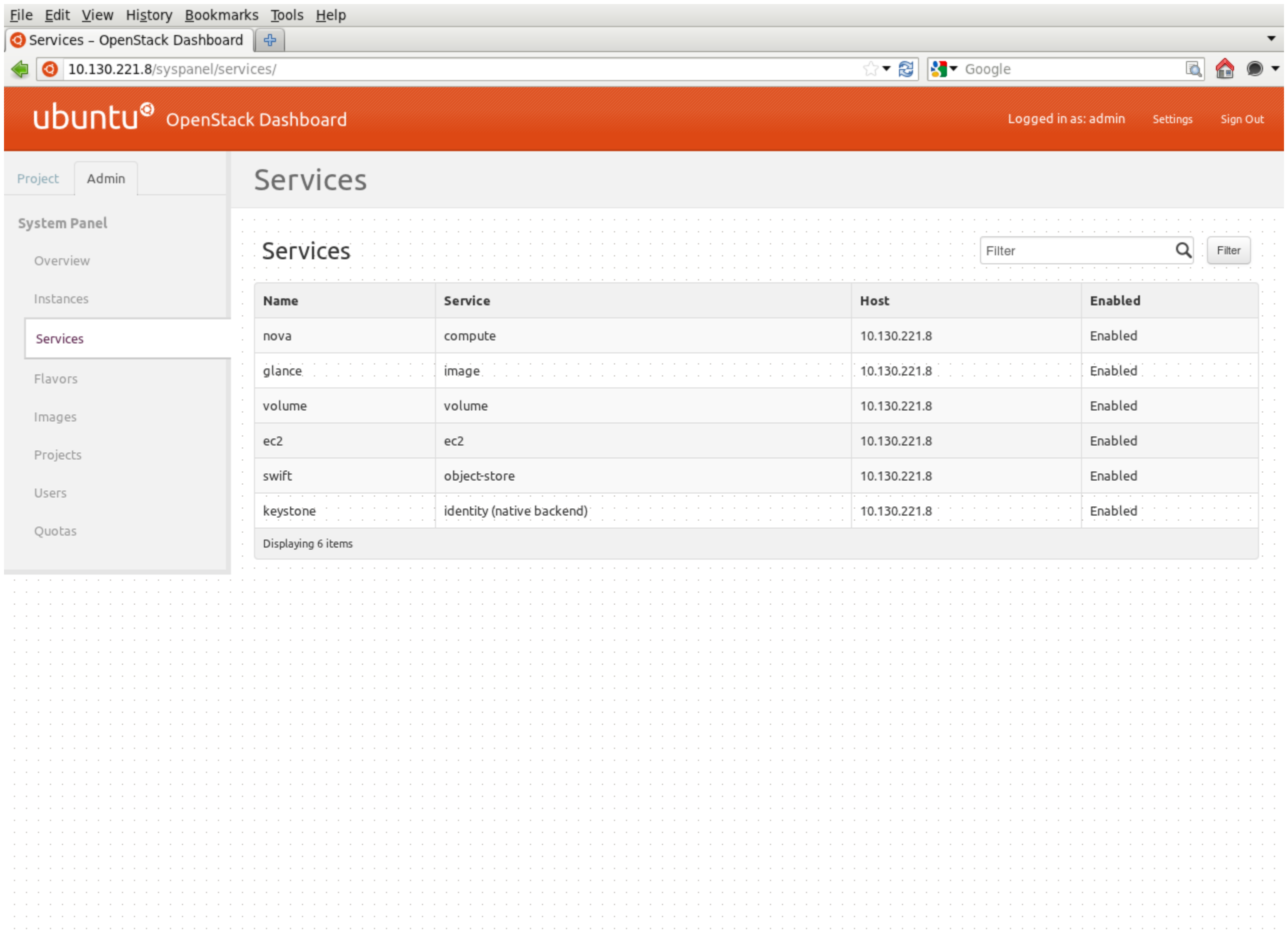


| | Tenant | Host | Instance Name | IP Address | Size | Status | Task | Power State | Actions |
|---|---|---|---|---|---|---|---|---|---|
| ☐ | admin | openstackvalidationserver1 | ubuntu_snap | 192.168.4.35 | 512MB RAM \| 1 VCPU \| 0 Disk | Active | None | Running | Edit Instance |
| ☐ | admin | openstackvalidationserver1 | ubuntu | 192.168.4.34 | 512MB RAM \| 1 VCPU \| 0 Disk | Active | None | Running | Edit Instance |

Displaying 2 items

## Services

The list of services defiend can be viewed on this page.



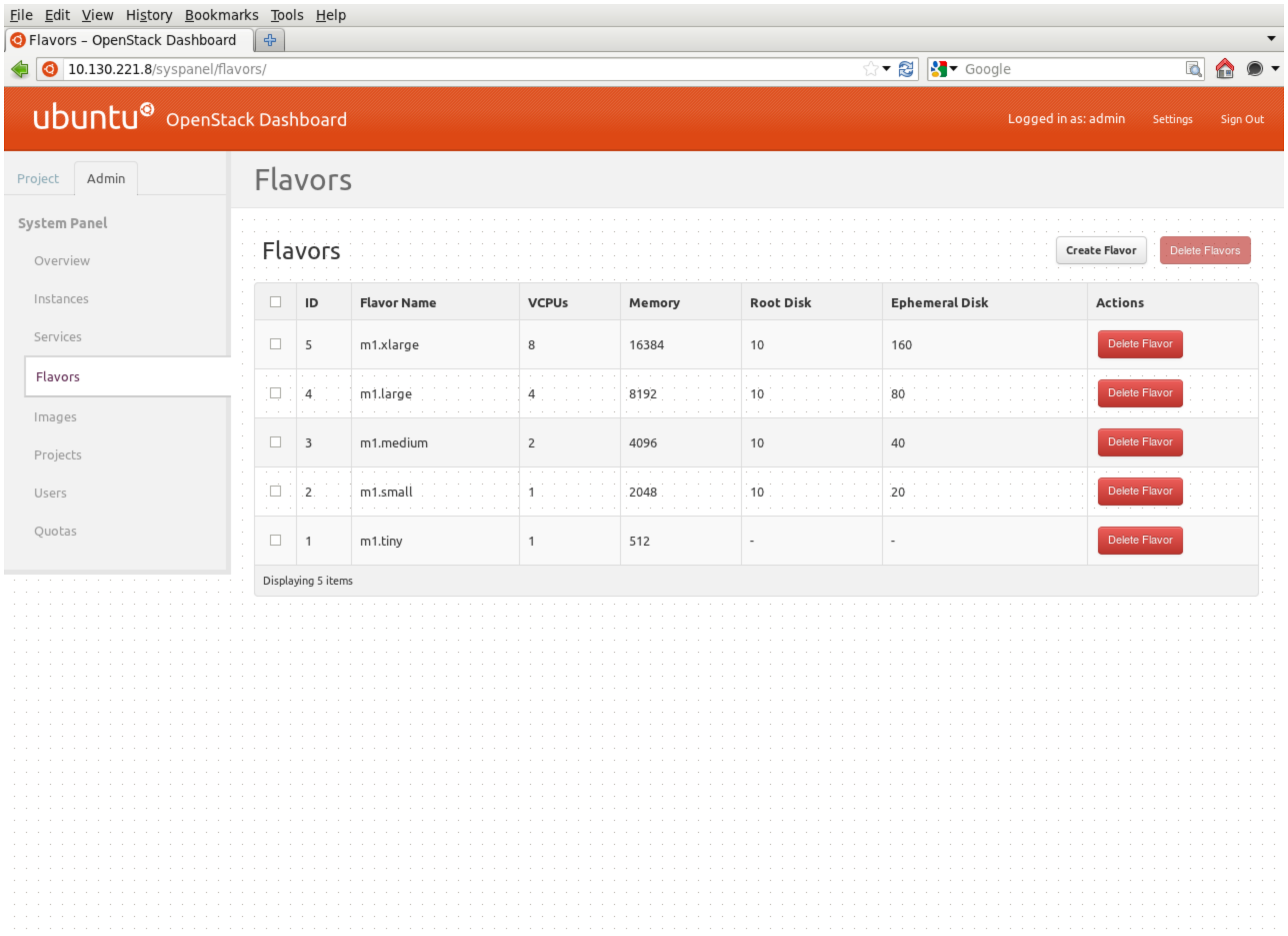
Services

| Name | Service | Host | Enabled |
|---|---|---|---|
| nova | compute | 10.130.221.8 | Enabled |
| glance | image | 10.130.221.8 | Enabled |
| volume | volume | 10.130.221.8 | Enabled |
| ec2 | ec2 | 10.130.221.8 | Enabled |
| swift | object-store | 10.130.221.8 | Enabled |
| keystone | identity (native backend) | 10.130.221.8 | Enabled |

Displaying 6 items

## Flavors

This page lists the currently available flavors that can be used to launch an instance. One can also create custom flavors on this page.



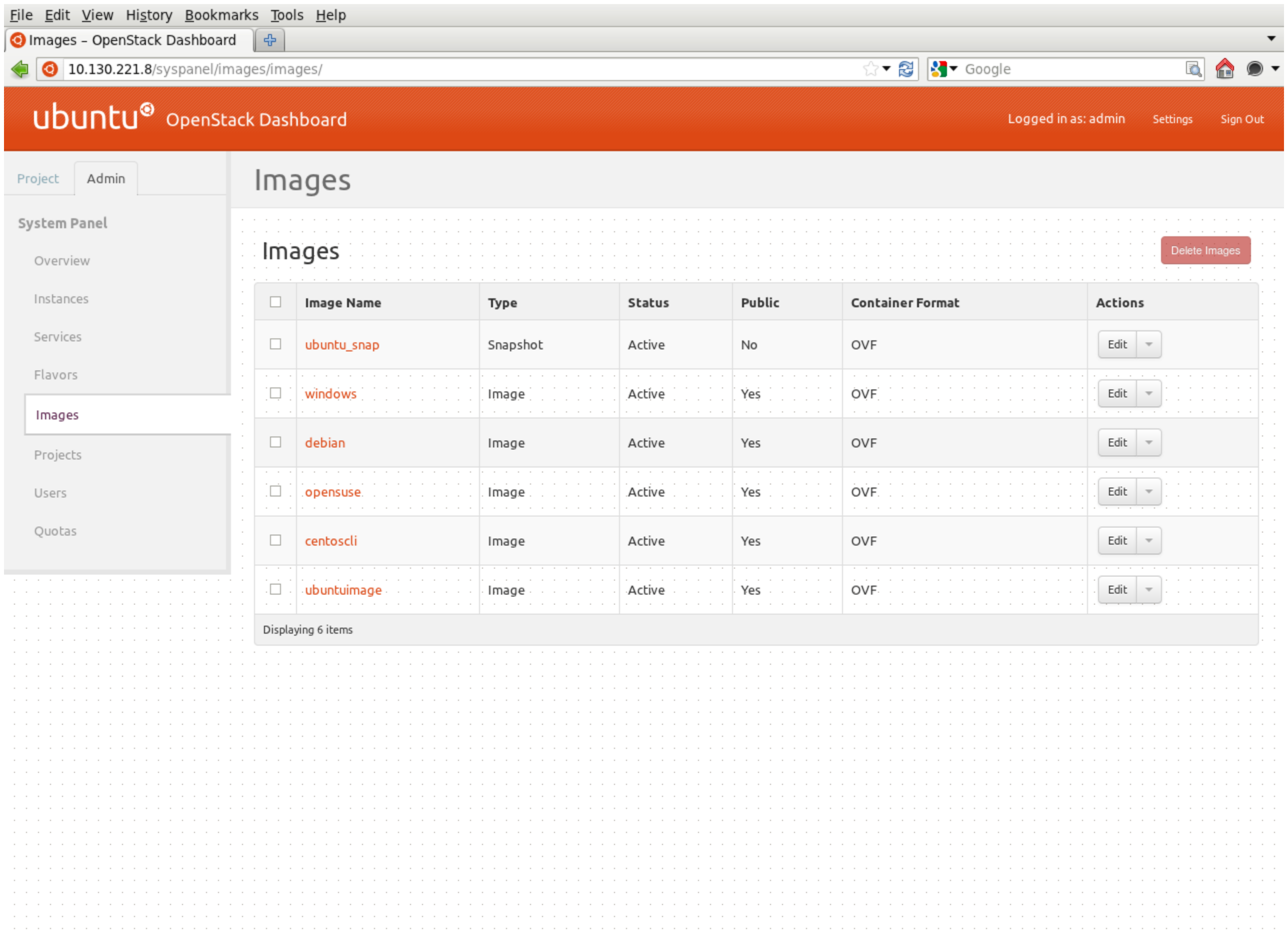
Flavors

| ID | Flavor Name | VCPUs | Memory | Root Disk | Ephemeral Disk | Actions |
|---|---|---|---|---|---|---|
| 5 | m1.xlarge | 8 | 16384 | 10 | 160 | Delete Flavor |
| 4 | m1.large | 4 | 8192 | 10 | 80 | Delete Flavor |
| 3 | m1.medium | 2 | 4096 | 10 | 40 | Delete Flavor |
| 2 | m1.small | 1 | 2048 | 10 | 20 | Delete Flavor |
| 1 | m1.tiny | 1 | 512 | - | - | Delete Flavor |

Displaying 5 items

## Images

This page lists the available images for the 'admin' user. One can also delete any images, if they are not required.



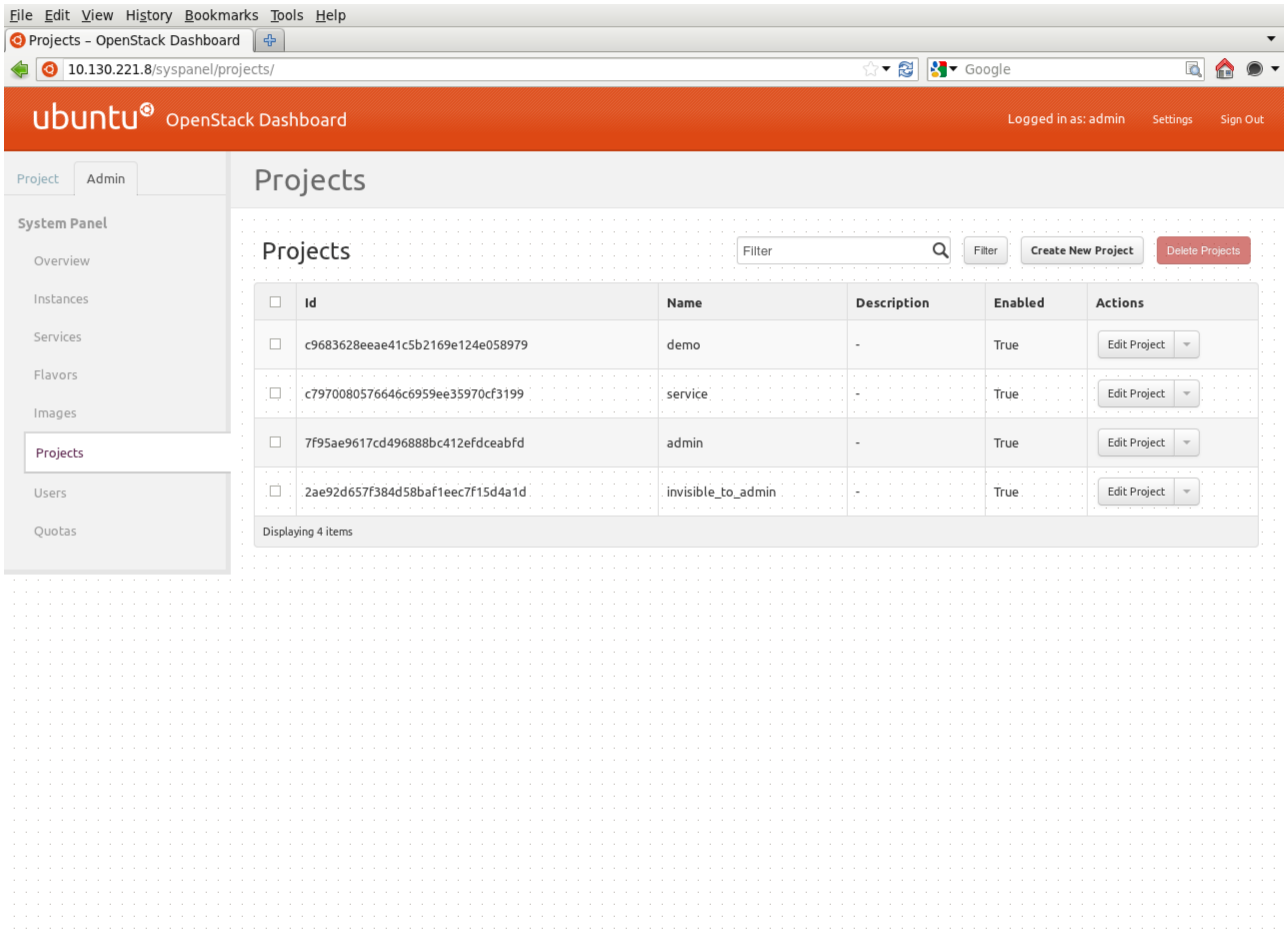
Images

| | Image Name | Type | Status | Public | Container Format | Actions |
|---|---|---|---|---|---|---|
| ☐ | ubuntu_snap | Snapshot | Active | No | OVF | Edit |
| ☐ | windows | Image | Active | Yes | OVF | Edit |
| ☐ | debian | Image | Active | Yes | OVF | Edit |
| ☐ | opensuse | Image | Active | Yes | OVF | Edit |
| ☐ | centoscli | Image | Active | Yes | OVF | Edit |
| ☐ | ubuntuimage | Image | Active | Yes | OVF | Edit |

Displaying 6 items

## Projects

This page lists the available projects (tenants) that have been created. One can also create new projects, assign users to the projects etc.



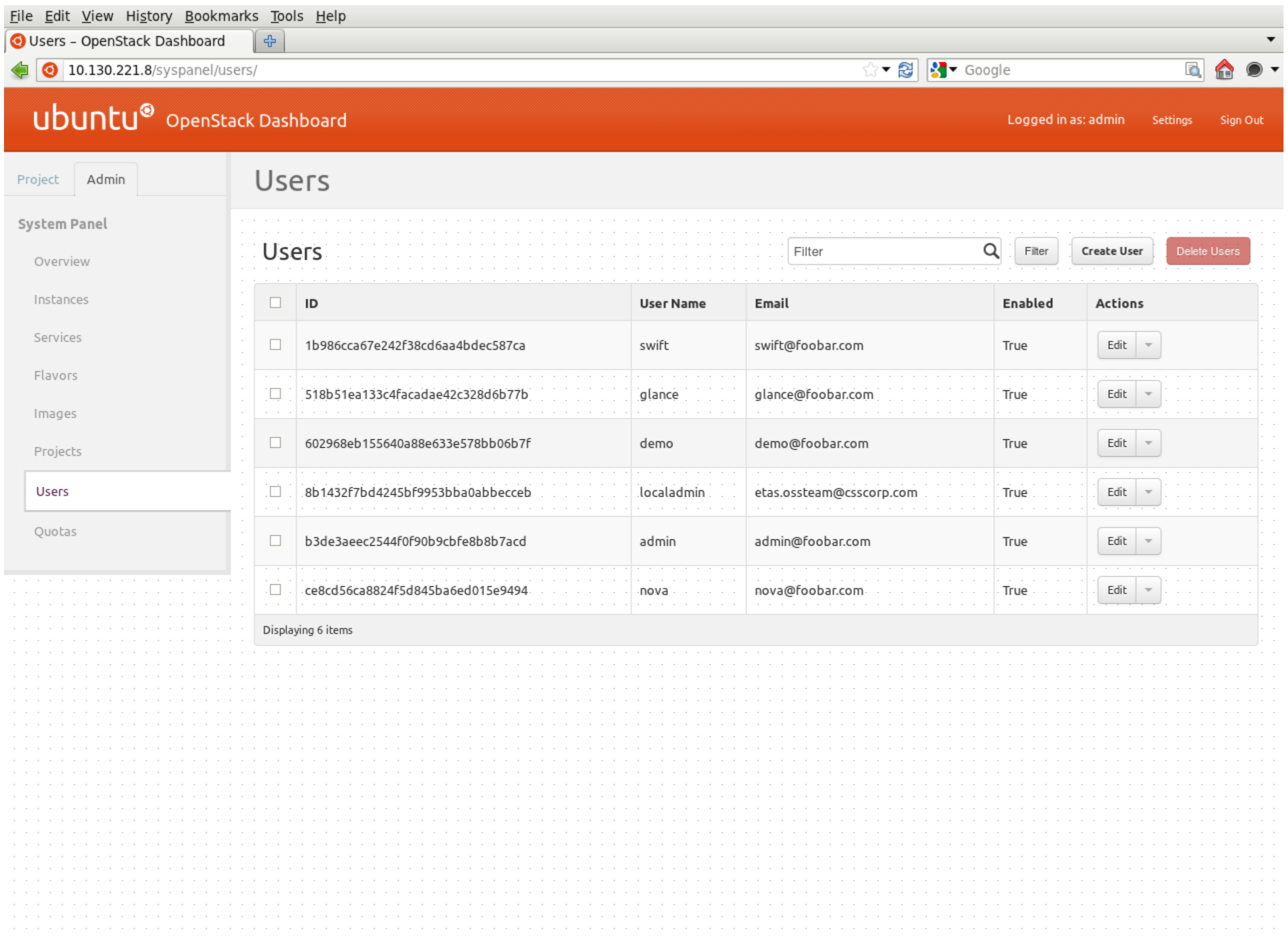
Projects

| Id | Name | Description | Enabled | Actions |
|---|---|---|---|---|
| c9683628eeae41c5b2169e124e058979 | demo | - | True | Edit Project |
| c7970080576646c6959ee35970cf3199 | service | - | True | Edit Project |
| 7f95ae9617cd496888bc412efdceabfd | admin | - | True | Edit Project |
| 2ae92d657f384d58baf1eec7f15d4a1d | invisible_to_admin | - | True | Edit Project |

Displaying 4 items

## Users

This page lists the users that have been created. One can also create new users, disable/ delete existing users.



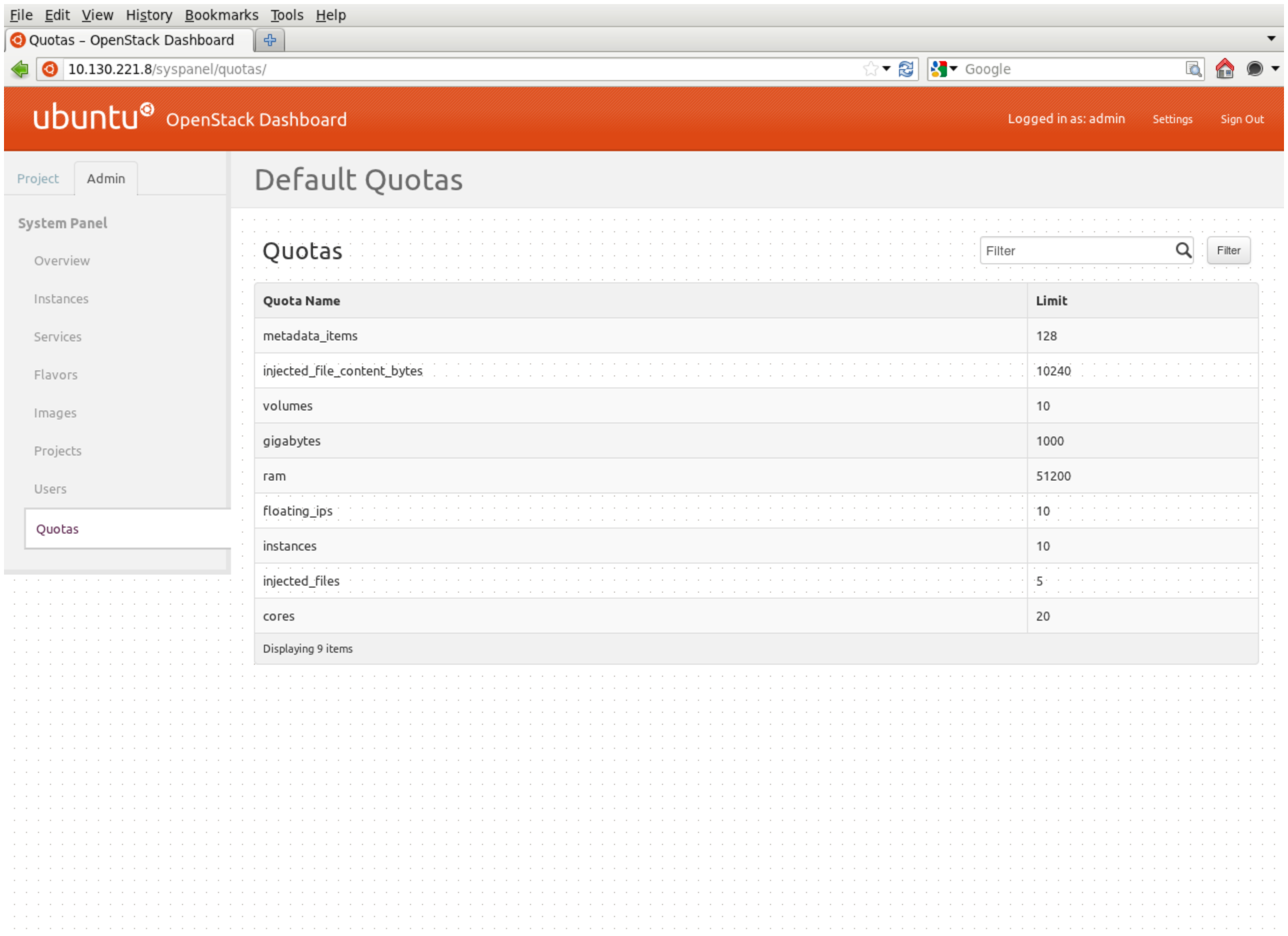
Users

| | ID | User Name | Email | Enabled | Actions |
|---|---|---|---|---|---|
| ☐ | 1b986cca67e242f38cd6aa4bdec587ca | swift | swift@foobar.com | True | Edit |
| ☐ | 518b51ea133c4facadae42c328d6b77b | glance | glance@foobar.com | True | Edit |
| ☐ | 602968eb155640a88e633e578bb06b7f | demo | demo@foobar.com | True | Edit |
| ☐ | 8b1432f7bd4245bf9953bba0abbecceb | localadmin | etas.ossteam@csscorp.com | True | Edit |
| ☐ | b3de3aeec2544f0f90b9cbfe8b8b7acd | admin | admin@foobar.com | True | Edit |
| ☐ | ce8cd56ca8824f5d845ba6ed015e9494 | nova | nova@foobar.com | True | Edit |

Displaying 6 items

## Users

This page lists the quota of resources allocated to a user; number of CPUs, amount of RAM, diskspace, max. number of instances that can be raised etc.



Default Quotas

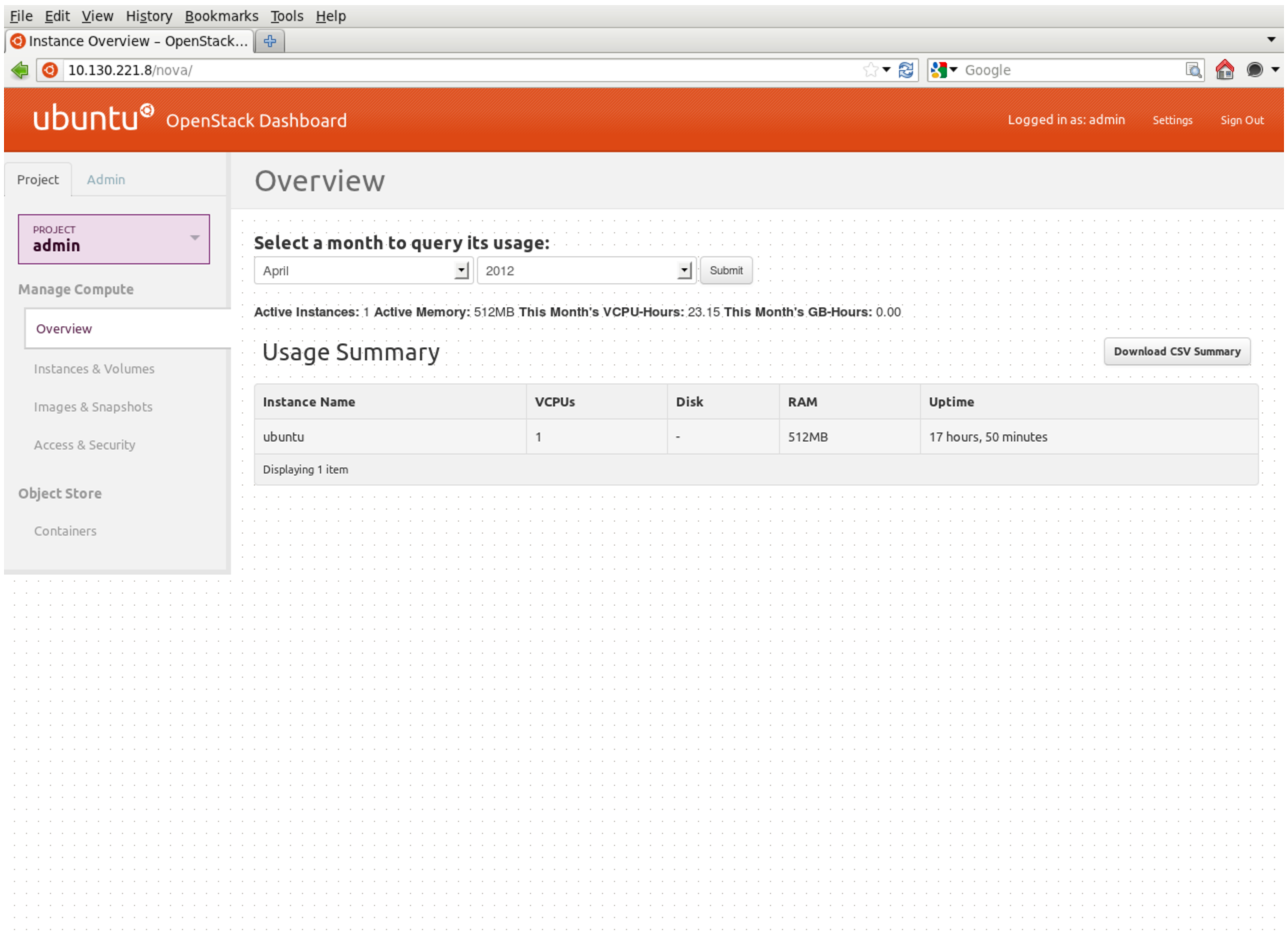
Quotas

| Quota Name | Limit |
|---|---|
| metadata_items | 128 |
| injected_file_content_bytes | 10240 |
| volumes | 10 |
| gigabytes | 1000 |
| ram | 51200 |
| floating_ips | 10 |
| instances | 10 |
| injected_files | 5 |
| cores | 20 |

Displaying 9 items

## Project Overview

This page shows an overview of the project 'admin'. One can view and download some basic usage metric reports here.



Overview

Select a month to query its usage:

April 2012 Submit

**Active Instances:** 1 **Active Memory:** 512MB **This Month's VCPU-Hours:** 23.15 **This Month's GB-Hours:** 0.00

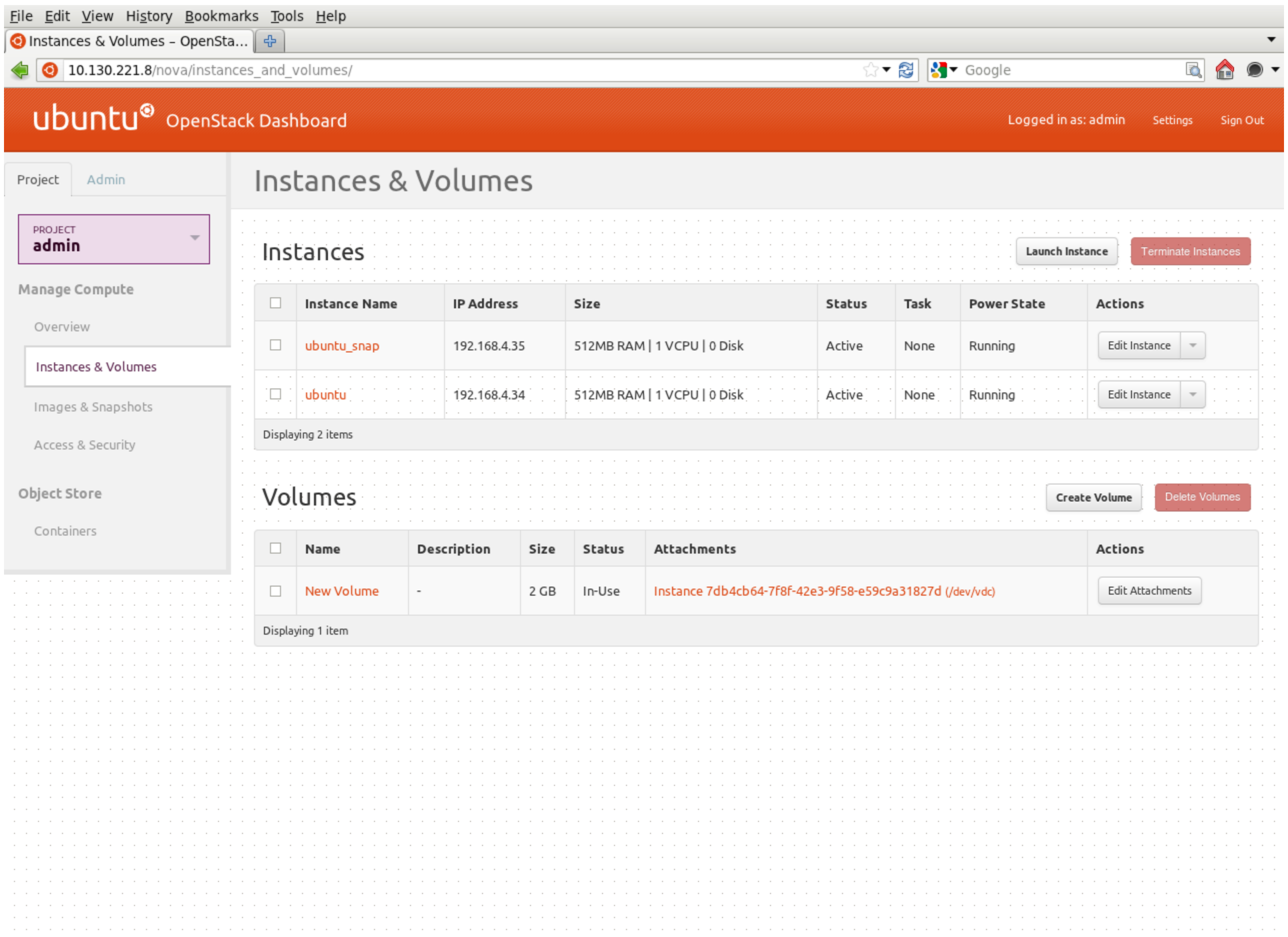
Usage Summary

| Instance Name | VCPUs | Disk | RAM | Uptime |
|---|---|---|---|---|
| ubuntu | 1 | - | 512MB | 17 hours, 50 minutes |

Displaying 1 item

## Instances & Volumes

This page lists all the instances belonging to various users of the project, instance properties etc. It also list all the volumes that have been created and their status; whether available or attached to any running instances. One can also create new volumes and attach them to the instances on this page.



Instances & Volumes

Instances

| | Instance Name | IP Address | Size | Status | Task | Power State | Actions |
|---|---|---|---|---|---|---|---|
| ☐ | ubuntu_snap | 192.168.4.35 | 512MB RAM \| 1 VCPU \| 0 Disk | Active | None | Running | Edit Instance |
| ☐ | ubuntu | 192.168.4.34 | 512MB RAM \| 1 VCPU \| 0 Disk | Active | None | Running | Edit Instance |

Displaying 2 items

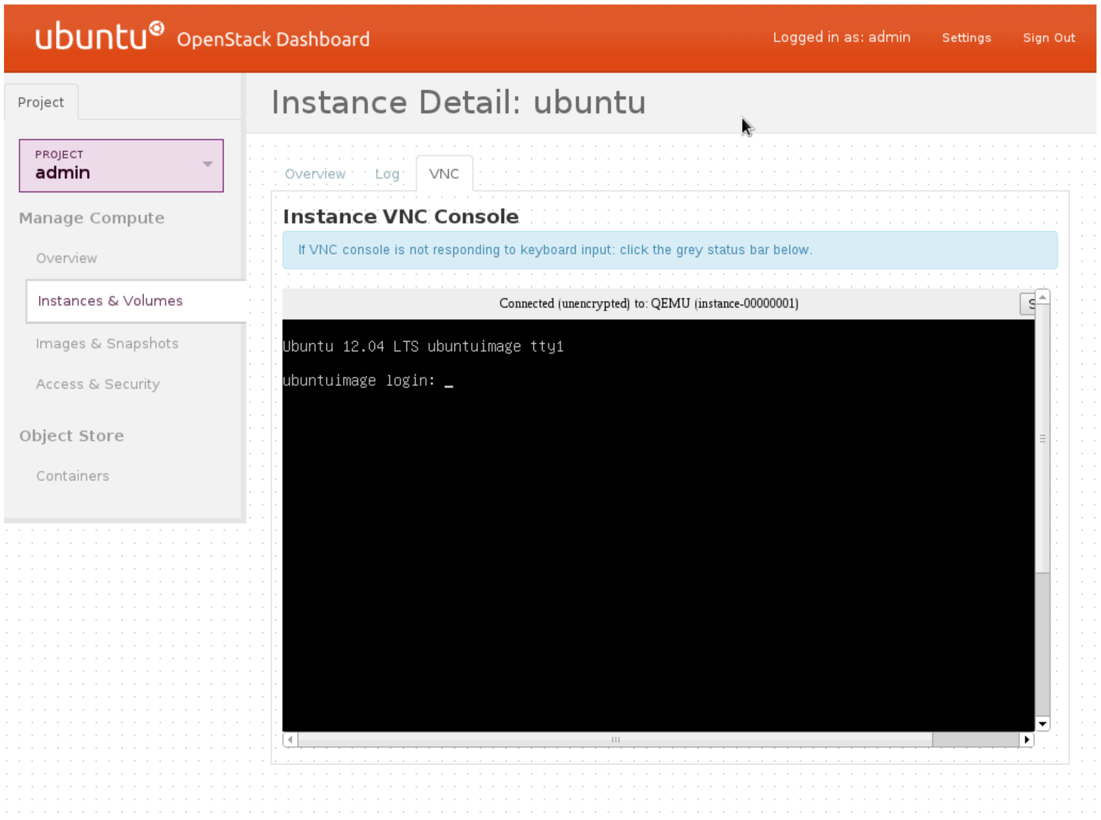
Volumes

| | Name | Description | Size | Status | Attachments | Actions |
|---|---|---|---|---|---|---|
| ☐ | New Volume | - | 2 GB | In-Use | Instance 7db4cb64-7f8f-42e3-9f58-e59c9a31827d (/dev/vdc) | Edit Attachments |

Displaying 1 item

## Instances - VNC Console

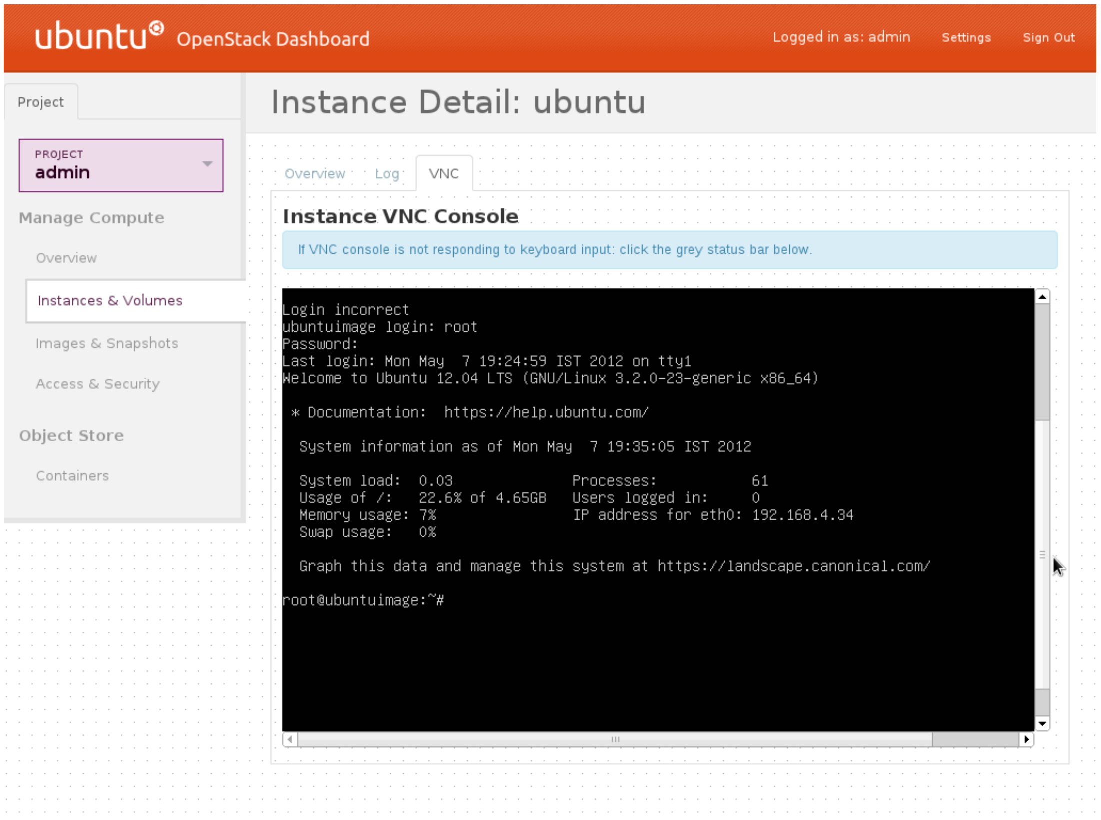
For a running instance, one can connect to the instance console via VNC.



If VNC console is not responding to keyboard input: click the grey status bar below.

Connected (unencrypted) to: QEMU (instance-00000001)

```
Ubuntu 12.04 LTS ubuntuimage tty1

ubuntuimage login: _
```

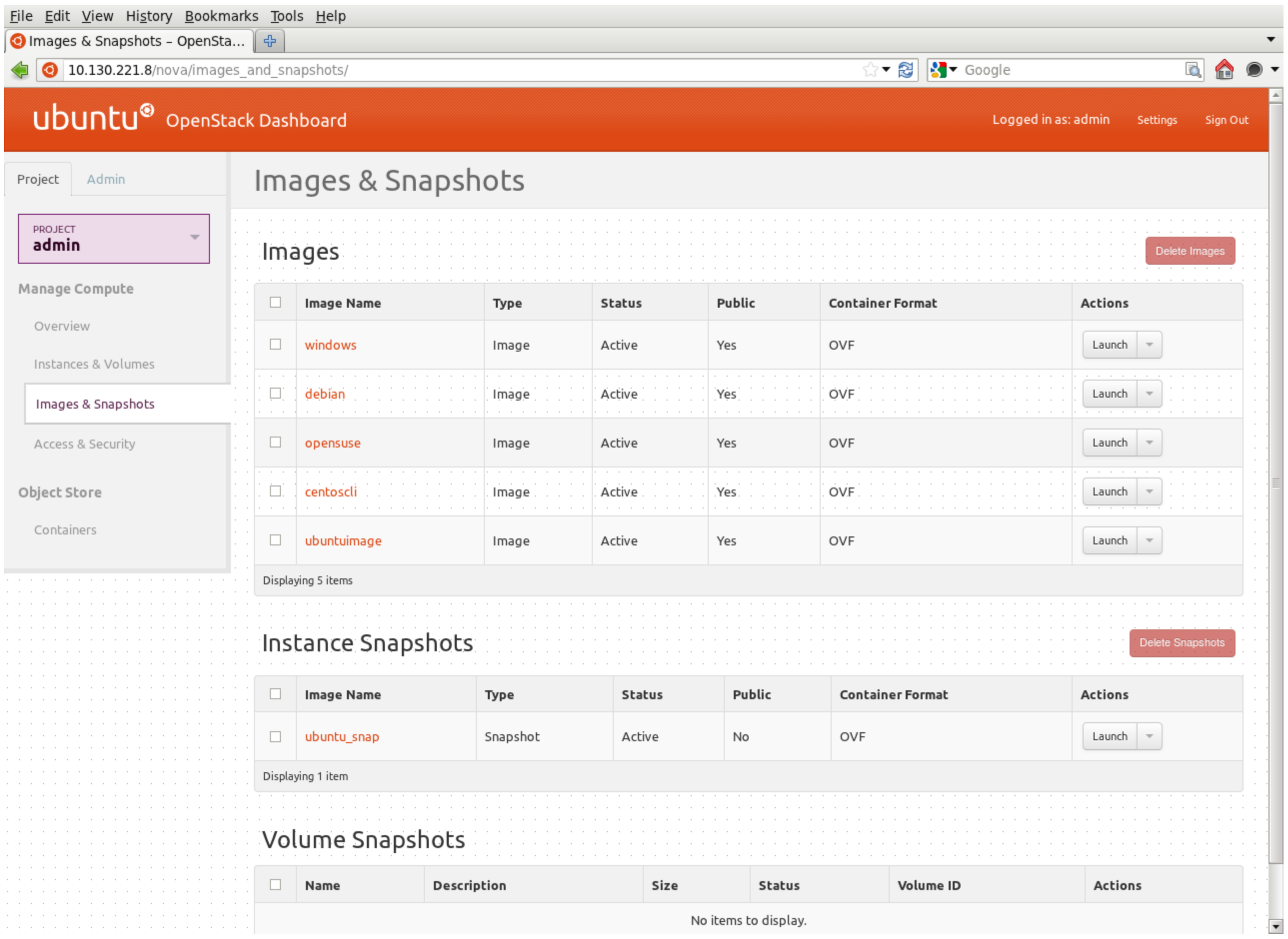


If VNC console is not responding to keyboard input: click the grey status bar below.

```
Login incorrect
ubuntuimage login: root
Password:
Last login: Mon May  7 19:24:59 IST 2012 on tty1
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Mon May  7 19:35:05 IST 2012

  System load:  0.03              Processes:           61
  Usage of /:   22.6% of 4.65GB   Users logged in:     0
  Memory usage: 7%                IP address for eth0: 192.168.4.34
  Swap usage:   0%

  Graph this data and manage this system at https://landscape.canonical.com/

root@ubuntuimage:~#
```

## Images & Snapshots

This page lists the custom images that have been uploaded. One can edit the image properties, delete and launch new instances of the images. This page also lists the snapshots taken from instances and volumes.



Images & Snapshots

Images

| | Image Name | Type | Status | Public | Container Format | Actions |
|---|---|---|---|---|---|---|
| ☐ | windows | Image | Active | Yes | OVF | Launch |
| ☐ | debian | Image | Active | Yes | OVF | Launch |
| ☐ | opensuse | Image | Active | Yes | OVF | Launch |
| ☐ | centoscli | Image | Active | Yes | OVF | Launch |
| ☐ | ubuntuimage | Image | Active | Yes | OVF | Launch |

Displaying 5 items

Instance Snapshots

| | Image Name | Type | Status | Public | Container Format | Actions |
|---|---|---|---|---|---|---|
| ☐ | ubuntu_snap | Snapshot | Active | No | OVF | Launch |

Displaying 1 item

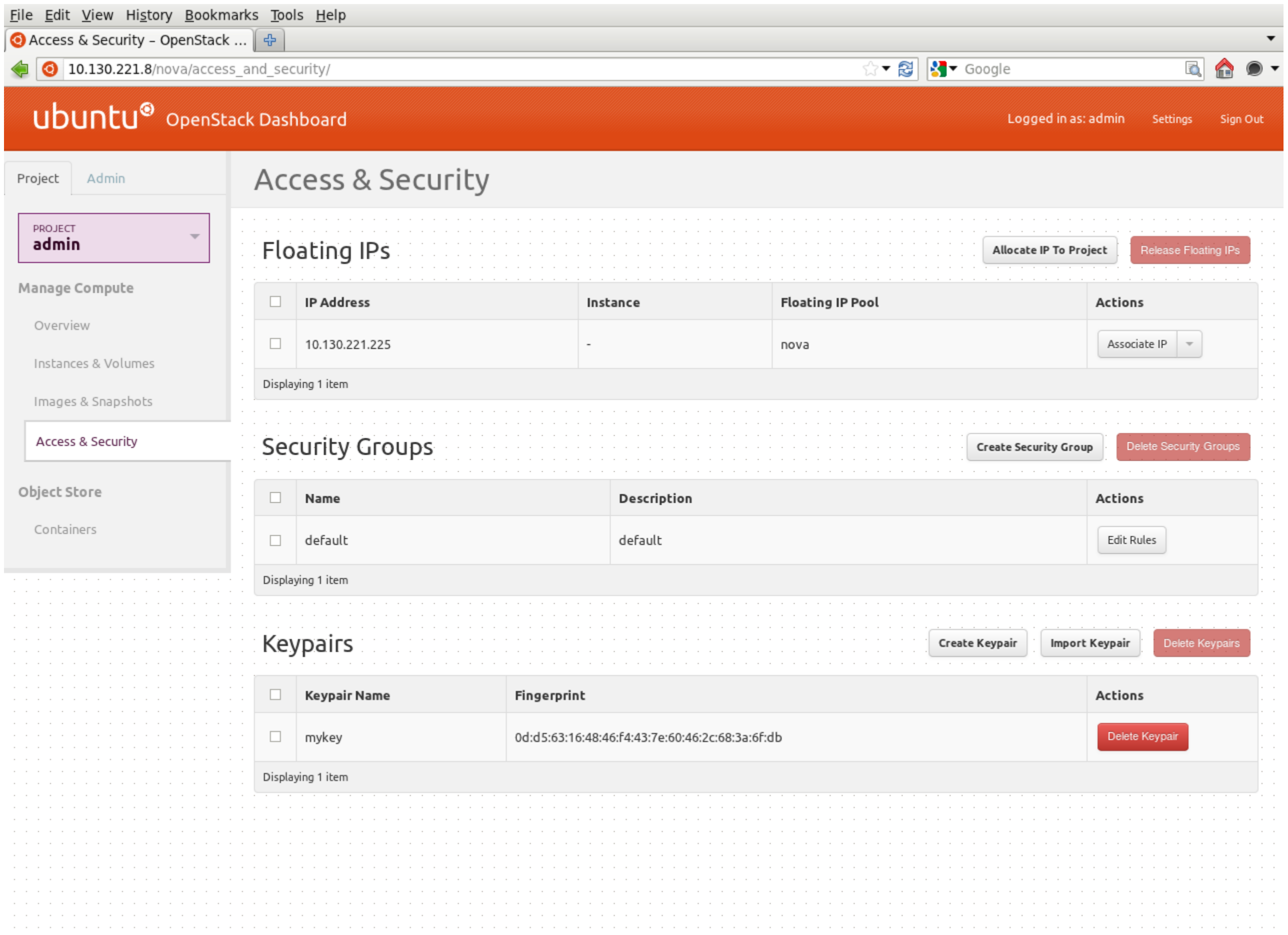
Volume Snapshots

| | Name | Description | Size | Status | Volume ID | Actions |
|---|---|---|---|---|---|---|

No items to display.

## Access & Security

On this page, one can allocate and release floating ip addresses, associate and dissociate them to instances. New security groups can be created and one can modify the rules belonging to each security group.



Access & Security

Floating IPs

| IP Address | Instance | Floating IP Pool | Actions |
|---|---|---|---|
| 10.130.221.225 | - | nova | Associate IP |

Displaying 1 item

Security Groups

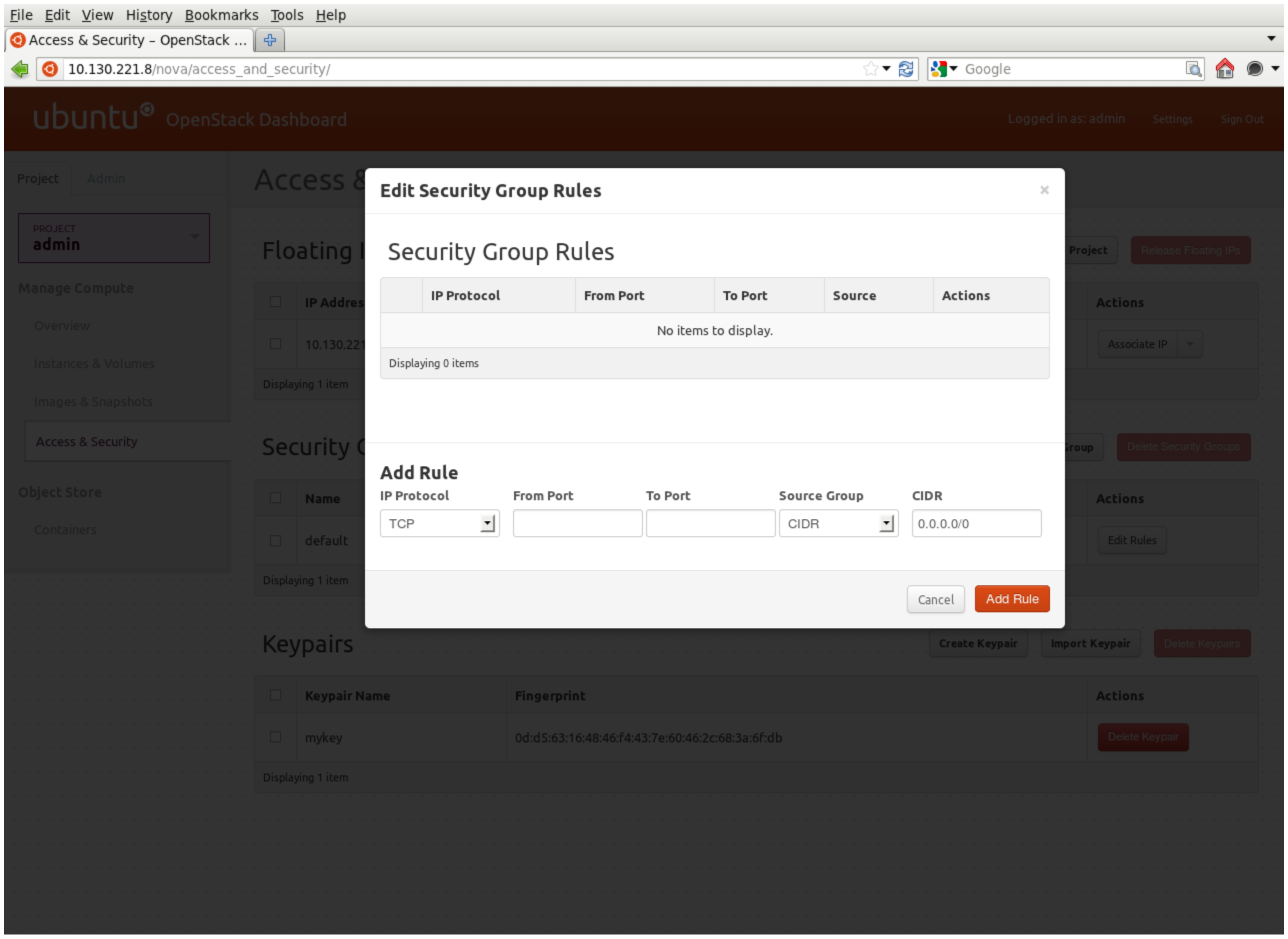
| Name | Description | Actions |
|---|---|---|
| default | default | Edit Rules |

Displaying 1 item

Keypairs

| Keypair Name | Fingerprint | Actions |
|---|---|---|
| mykey | 0d:d5:63:16:48:46:f4:43:7e:60:46:2c:68:3a:6f:db | Delete Keypair |

Displaying 1 item



**Edit Security Group Rules**

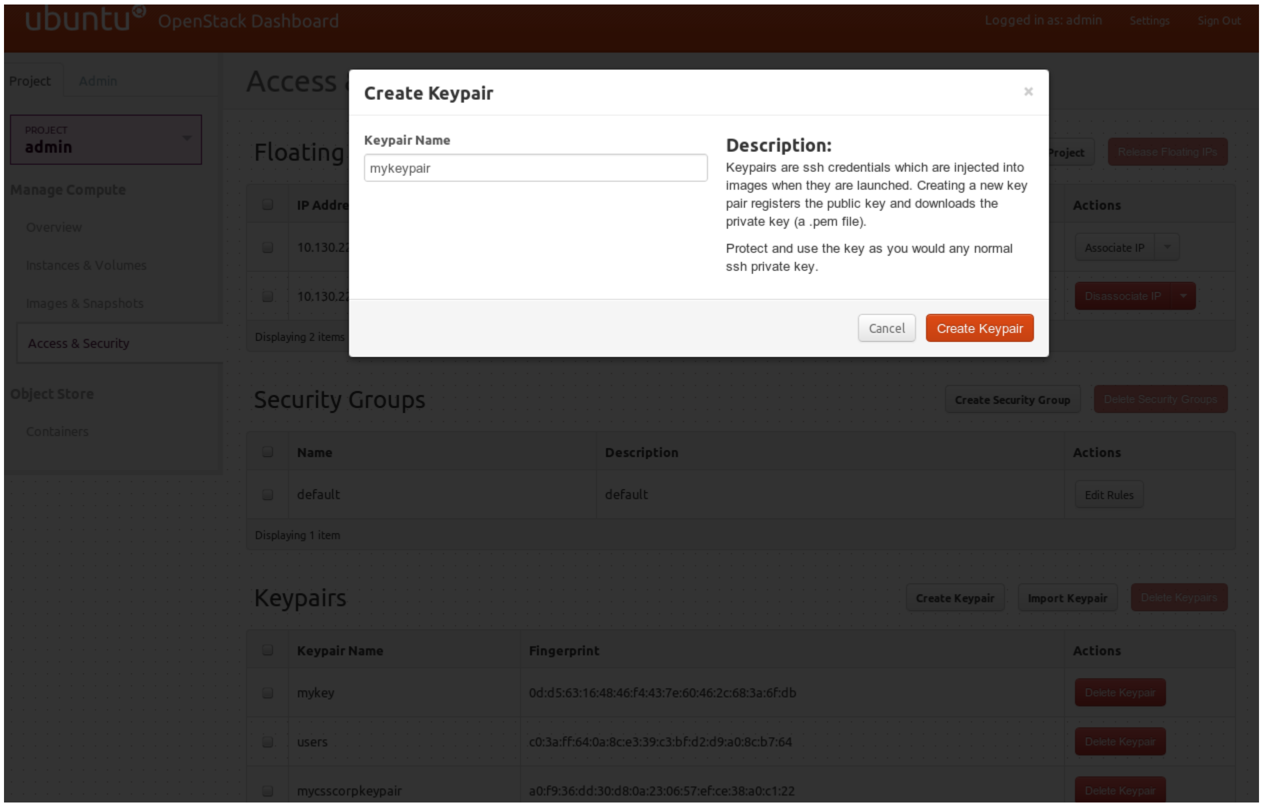
Security Group Rules

| | IP Protocol | From Port | To Port | Source | Actions |
|---|---|---|---|---|---|
| No items to display. | | | | | |

Displaying 0 items

**Add Rule**

| IP Protocol | From Port | To Port | Source Group | CIDR |
|---|---|---|---|---|
| TCP | | | CIDR | 0.0.0.0/0 |



**Create Keypair**

Keypair Name

mykeypair

**Description:**

Keypairs 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 you would any normal ssh private key.

Security Groups

| Name | Description |
|---|---|
| default | default |

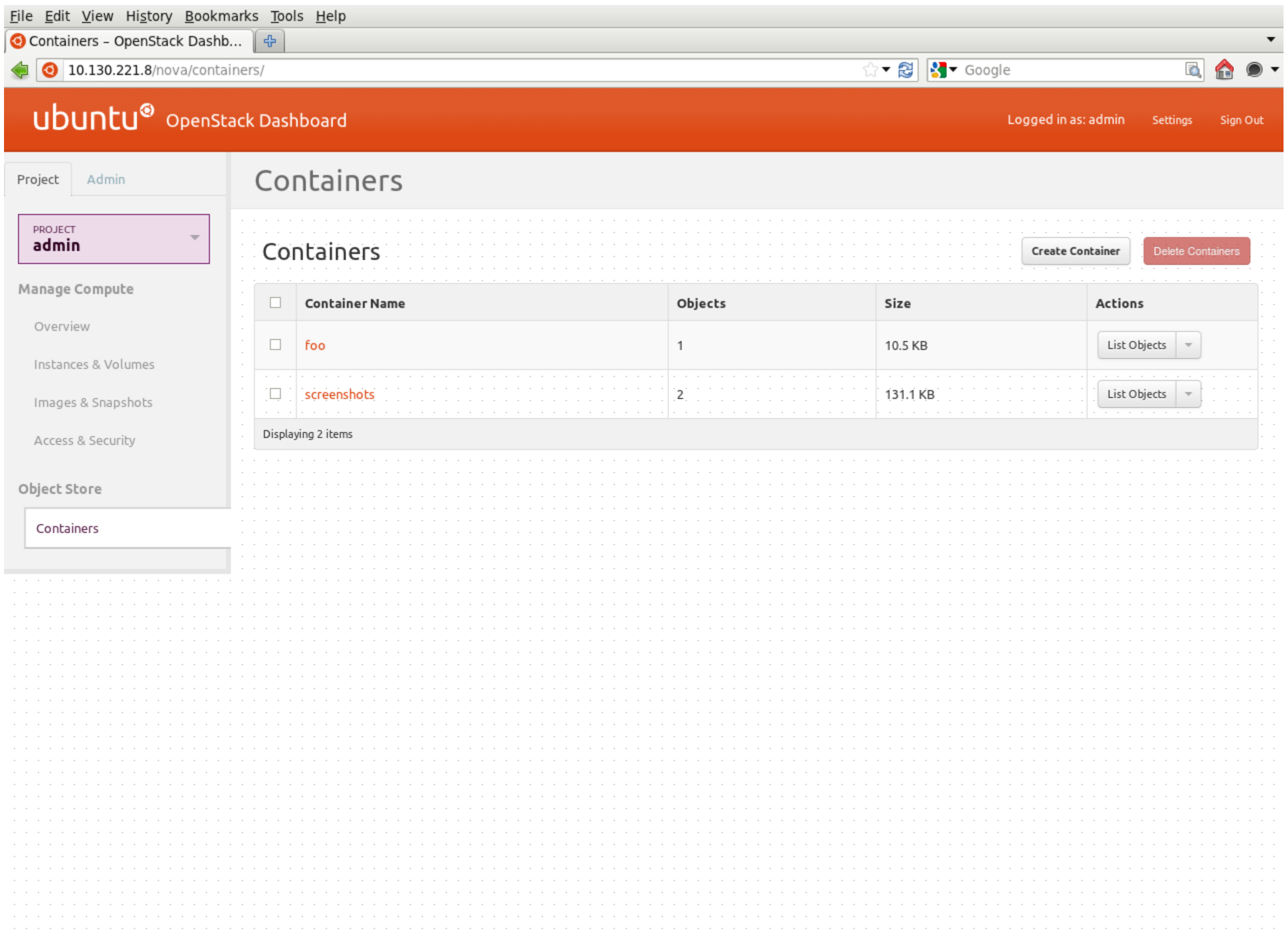
Displaying 1 item

Keypairs

| Keypair Name | Fingerprint |
|---|---|
| mykey | 0d:d5:63:16:48:46:f4:43:7e:60:46:2c:68:3a:6f:db |
| users | c0:3a:ff:64:0a:8c:e3:39:c3:bf:d2:d9:a0:8c:b7:64 |
| mycsscorpkeypair | a0:f9:36:dd:30:d8:0a:23:06:57:ef:ce:38:a0:c1:22 |

# Containers & Objects

On this page, one can create/delete containers, lists objects, upload/download objects and delete objects.

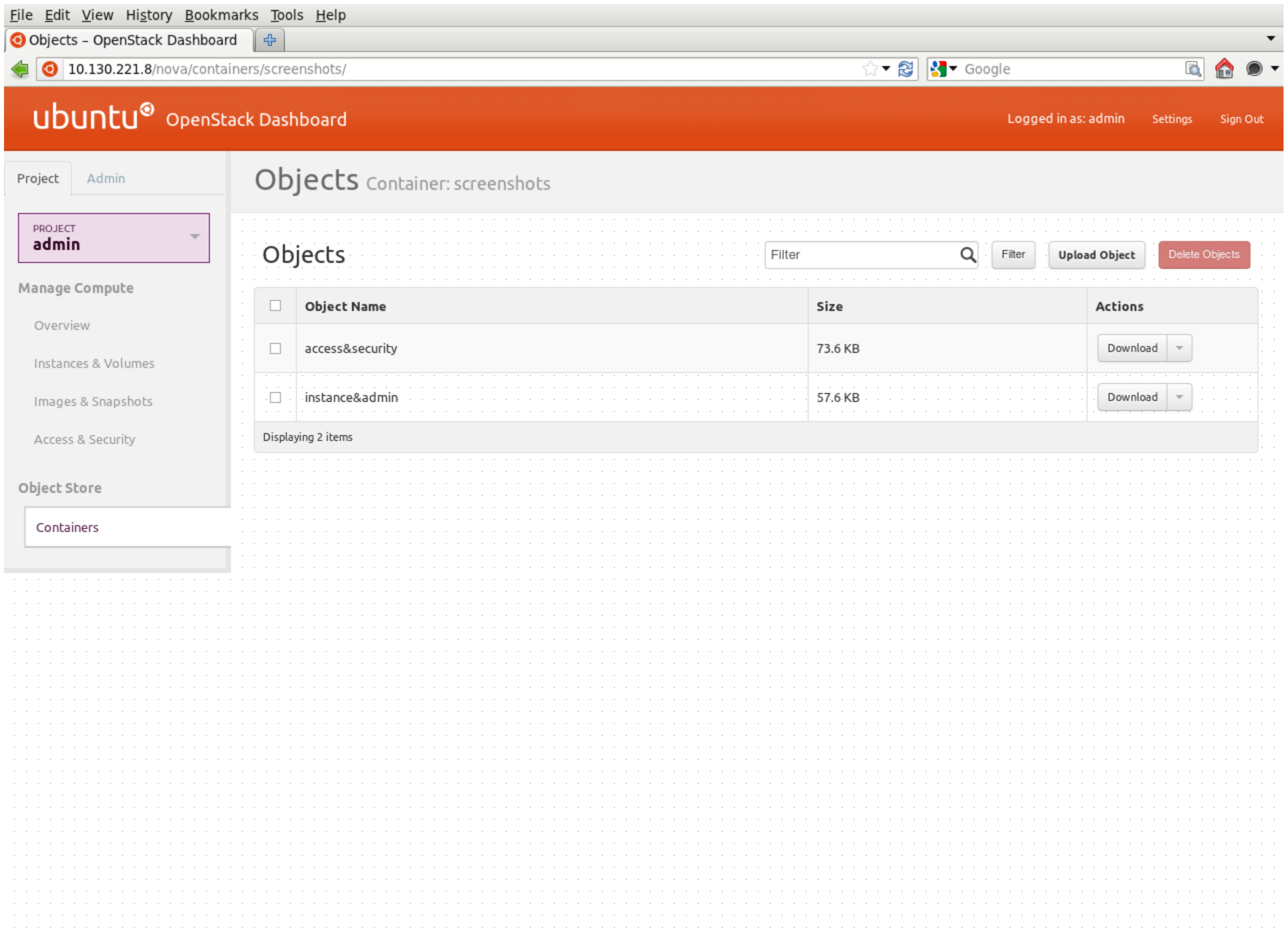


Containers

Containers

| | Container Name | Objects | Size | Actions |
|---|---|---|---|---|
| ☐ | foo | 1 | 10.5 KB | List Objects |
| ☐ | screenshots | 2 | 131.1 KB | List Objects |

Displaying 2 items



| | Object Name | Size | Actions |
|---|---|---|---|
| ☐ | access&security | 73.6 KB | Download |
| ☐ | instance&admin | 57.6 KB | Download |

Displaying 2 items

# 6. Storage Management

## Nova-volume

Nova-volume provides persistent block storage compatible with Amazon's Elastic Block Store. The storage on the instances is non-persistent by nature and hence any data that are generated and stored on the file system on the first disk of the instance are 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.

Nova commands 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

```
nova volume-create --display_name myvolume 10
```

List the volumes

```
nova volume-list
```

You should see an output like this:

```
+----+-----------+-------------+------+-------------
+--------------------------------------+
| ID |   Status  | Display Name | Size | Volume Type |             Attached to
              |
+----+-----------+-------------+------+-------------
+--------------------------------------+
| 1  | in-use    | New Volume   | 20   | None        |
 7db4cb64-7f8f-42e3-9f58-e59c9a31827d |
| 4  | available | volume1      | 10   | None        |
              |
| 5  | available | myvolume     | 10   | None        |
              |
| 6  | available | myvolume1    | 10   | None        |
              |
+----+-----------+-------------+------+-------------
+--------------------------------------+
```

Attach a volume to a running instance

```
nova volume-attach 857d70e4-35d5-4bf6-97ed-bf4e9a4dcf5a <volume-id> /dev/vdb
```

A volume can only be attached to one instance at a time. When nova volume-list 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 nova volume-list, 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 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.

```
nova volume-detach 857d70e4-35d5-4bf6-97ed-bf4e9a4dcf5a <volume-id>
```

The data on the volume persists even after the volume is detached from an instance or after the instance is terminated. You can view the data after attaching 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.

## Swift

Swift is a reliable, distributed, massively scalable blob storage service that can be used for storage and archival of objects. Swift provides a REST interface. You can use Swift commandline which is an interface for the OpenStack object store service.

To get the information about swift account, container and objects.

```
$ swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0/ -U service:swiftuser -K
 swiftpasswd stat
Account: AUTH_43b42dae-dc0b-4a4b-ac55-97de614d6e6e
Containers: 1
Objects: 1
Bytes: 1124
Accept-Ranges: bytes
X-Trans-Id: txb21186a9eef64ed295a1e95896a0fc72
```

To get information about a particular container (mycontainer):

```
$ swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0/ -U service:swiftuser -K
 swiftpasswd stat mycontainer
```

To get information about an object (abc123.txt) within container (mycontainer):

```
$ swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0/ -U service:swiftuser -K
 swiftpasswd stat mycontainer abc123.txt
```

To list available containers in account:

```
$ swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0/ -U service:swiftuser -K
 swiftpasswd list
```

To list all containers whose names begin with 'my':

```
$ swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0/ -U service:swiftuser -K
 swiftpasswd --prefix=my list
```

To list all objects within container 'mycontainer':

```
$ swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0/ -U service:swiftuser -K
 swiftpasswd --prefix=my list mycontainer
```

To upload files 'abc.txt' and 'xyz.txt' to 'mycontainer':

```
$ swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0/ -U service:swiftuser -K
 swiftpasswd upload mycontainer /path/abc.txt /path/xyz.txt
```

To download all the objects from all containers:

```
$ swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0/ -U service:swiftuser -K
 swiftpasswd --all download
```

To download all objects from container 'mycontainer':

```
$ swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0/ -U service:swiftuser -K
 swiftpasswd download mycontainer
```

To download 'abc.txt' and 'xyz.txt' from container 'mycontainer':

```
$ swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0/ -U service:swiftuser -K
 swiftpasswd download mycontainer abc.txt xyz.txt
```

To delete all objects in all containers:

```
$ swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0/ -U service:swiftuser -K
 swiftpasswd --all delete
```

To delete all objects in container 'mycontainer':

```
$ swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0/ -U service:swiftuser -K
 swiftpasswd delete mycontainer
```

To delete files 'abc.txt' and 'xyz.txt' from container 'mycontainer':

```
$ swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0/ -U service:swiftuser -K
 swiftpasswd delete mycontainer abc.txt xyz.txt
```

# 7. Network Management

## 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. OpenStack instances can have 2 types of IP addresses associated to it: Private IP address (fixed) and Public IP address (floating). Private IP addresses are typically used for communication between instances (internal) and public IP addresses are used for communication of instances with the outside world (external or Internet). The so-called public IP addresses need not necessarily be IP addresses route-able on the Internet ; they can even be addresses on a corporate LAN.

The network configurations inside the instances are done with private IP addresses in view. The association between the private IP and the public IP addresses and necessary routing are handled by nova-network and the instances need not be aware of them.

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 ideal 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 private --fixed_range_v4=192.168.4.3/27 --
num_networks=1 --bridge=br100 --bridge_interface=eth1 --network_size=32
   sudo nova-manage floating create --ip_range=10.10.10.224/27
```

The public IP which you are going to associate with an instance needs to be allocated first by using the command:

```
    nova floating-ip-create
    +----------------+-------------+----------+------+
    |       Ip       | Instance Id | Fixed Ip | Pool |
    +----------------+-------------+----------+------+
    | 10.10.10.225   | None        | None     | nova |
    +----------------+-------------+----------+------+
```

You can then associate a public IP to a running instance by using the command:

```
nova add-floating-ip <instance-name> 10.10.2.225
```

# 8. Security

## Security Overview

OpenStack provides ingress filtering for the instances based on the concept of security groups. OpenStack accomplishes ingress filtering by creating suitable iptables 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 (during bundling process) using a firewall.

The OpenStack Dashboard lets you manage security groups and also let you specify a security group while launching an instance. You can also use commands like 'nova secgroup-add-rule' etc. for this purpose.

Here are a few nova commands to manage security groups.

Create a security group named "myservers".

```
nova secgroup-create <name> <description>
nova secgroup-create myservers my-default-server-group
```

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

```
nova secgroup-add-rule myservers tcp 22 22 192.168.1.1/0
nova secgroup-add-rule myservers icmp -1 -1 192.168.1.1/0
```

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

```
nova secgroup-add-rule myservers tcp 3389 3389 192.168.1.1/0
```

Rules can be viewed with the command.

```
$ nova secgroup-list-rules myservers
+-------------+-----------+---------+--------------+--------------+
| IP Protocol | From Port | To Port |  IP Range    | Source Group |
+-------------+-----------+---------+--------------+--------------+
|             |           |         |              |              |
| icmp        | -1        | -1      |192.168.1.1/0 |  myservers   |
| tcp         | 22        | 22      |192.168.1.1/0 |  myservers   |
+-------------+-----------+---------+--------------+--------------+
```

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

```
nova secgroup-delete-rule myservers ssh 22 22 192.168.1.1
```

Delete the security group "myservers"

```
nova secgroup-delete myservers
```

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

```
nova boot --flavor 1 --image 9bab7ce7-7523-4d37-831f-c18fbc5cb543 --key_name
 mykey myinstance --security_groups 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 nova secgroup-add-rule , nova secgroup-delete-rule commands.

# 9. OpenStack Commands

## Nova Commands

nova is the command line interface for OpenStack Compute API.

```
Usage: nova command [options] [args]

Commands:
    help                      Display help about this program or one of its
 subcommands.
    actions                   Retrieve server actions.
    backup-schedule           Show or edit the backup schedule for a server.
    backup-schedule-delete Delete the backup schedule for a server.
    boot                      Boot a new server.
    delete                    Immediately shut down and delete a server.
    diagnostics               Retrieve server diagnostics.
    flavor-list               Print a list of available 'flavors' (sizes of
 servers).
    image-create              Create a new image by taking a snapshot of a
 running server.
    image-delete              Delete an image.
    image-list                Print a list of available images to boot from.
    ip-share                  Share an IP address from the given IP group onto a
 server.
    ip-unshare                Stop sharing an given address with a server.
    ipgroup-create            Create a new IP group.
    ipgroup-delete            Delete an IP group.
    ipgroup-list              Show IP groups.
    ipgroup-show              Show details about a particular IP group.
    list                      List active servers.
    pause                     Pause a server.
    reboot                    Reboot a server.
    rebuild                   Shutdown, re-image, and re-boot a server.
    rename                    Rename a server.
    rescue                    Rescue a server.
    resize                    Resize a server.
    resize-confirm            Confirm a previous resize.
    resize-revert             Revert a previous resize (and return to the
 previous VM).
    resume                    Resume a server.
    root-password             Change the root password for a server.
    show                      Show details about the given server.
    suspend                   Suspend a server.
    unpause                   Unpause a server.
    unrescue                  Unrescue a server.
    zone                      Show or edit a child zone.
    zone-add                  Add a new child zone.
    zone-delete               Delete a zone.
    zone-info                 Get this zones name and capabilities.
    zone-list                 List the children of a zone.
```

## Glance Commands

Glance is the command line interface for the OpenStack Imaging service.

```
Usage: glance command [options] [args]

Commands:
    help command    Output help for one of the commands below
    add             Adds a new image to Glance
    update          Updates an image's metadata in Glance
    delete          Deletes an image from Glance
    index           Return brief information about images in Glance
    details         Return detailed information about images in
                    Glance
    show            Show detailed information about an image in
                    Glance
    clear           Removes all images and metadata from Glance

Member Commands:
    image-members     List members an image is shared with
    member-images     List images shared with a member
    member-add        Grants a member access to an image
    member-delete     Revokes a member's access to an image
    members-replace   Replaces all membership for an image
```

## Swift Commands

Swift is the command line interface for OpenStack Object Store service.

```
Usage: swift command [options] [args]

Commands:
    stat     Displays information for the account, container, or object
 depending
             on the args given (if any).
    list     Lists the containers for the account or the objects for a
 container.
    upload   Uploads to the given container the files and directories
 specified by
             the remaining args.
    post     Updates meta information for the account, container, or object
             depending on the args given.
    download Downloads everything in the account (with --all), or everything
 in
             a container, or a list of objects depending on the args given.
    delete   Deletes everything in the account (with --all), or everything in
             a container, or a list of objects depending on the args given.
```

## Keystone Commands

Keystone is the command line interface to the OpenStack Identity service.

```
Usage: keystone command [options] [args]

Commands:
    catalog                  List service catalog, possibly filtered by
 service.
    ec2-credentials-create   Create EC2-compatibile credentials for user per
 tenant
```

```
   ec2-credentials-delete  Delete EC2-compatibile credentials
   ec2-credentials-get     Display EC2-compatibile credentials
   ec2-credentials-list    List EC2-compatibile credentials for a user
   endpoint-create         Create a new endpoint associated with a service
   endpoint-delete         Delete a service endpoint
   endpoint-get            Find endpoint filtered by a specific attribute or
service type
   endpoint-list           List configured service endpoints
   role-create             Create new role
   role-delete             Delete role
   role-get                Display role details
   role-list               List all roles, or only those granted to a user.
   service-create          Add service to Service Catalog
   service-delete          Delete service from Service Catalog
   service-get             Display service from Service Catalog
   service-list            List all services in Service Catalog
   tenant-create           Create new tenant
   tenant-delete           Delete tenant
   tenant-get              Display tenant details
   tenant-list             List all tenants
   tenant-update           Update tenant name, description, enabled status
   token-get               Display the current user token
   user-create             Create new user
   user-delete             Delete user
   user-get                Display user details.
   user-list               List users
   user-password-update    Update user password
   user-role-add           Add role to user
   user-role-remove        Remove role from user
   user-update             Update user's name, email, and enabled status
   discover                Discover Keystone servers and show authentication
protocols and
   help                    Display help about this program or one of its
subcommands.
```