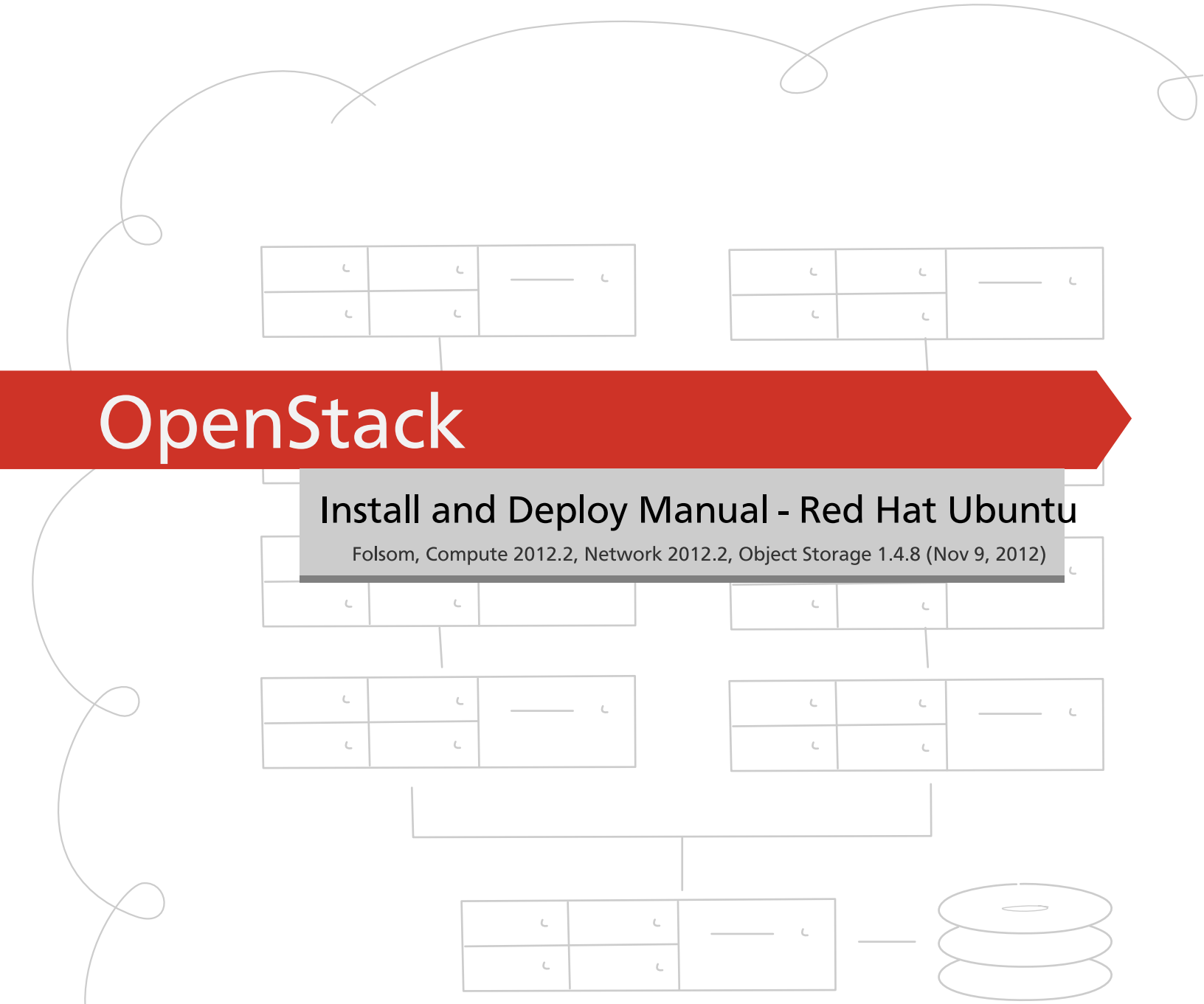


OpenStack

Install and Deploy Manual - Red Hat Ubuntu

Folsom, Compute 2012.2, Network 2012.2, Object Storage 1.4.8 (Nov 9, 2012)

docs.openstack.org

OpenStack Install and Deploy Manual - Ubuntu

Folsom, Compute 2012.2, Network 2012.2, Object Storage 1.4.8 (2012-11-09)

The OpenStack™ system has several key projects that are separate installations but can work together depending on your cloud needs: OpenStack Compute, OpenStack Object Storage, OpenStack Identity Service, and OpenStack Image Service. You can install any of these projects separately and then configure them either as standalone or connected entities. This guide walks through an installation using packages available through Ubuntu 12.04. It offers explanations for the configuration choices as well as sample configuration files.

Table of Contents

List of Figures

List of Tables

1. Installing OpenStack Walk-through

The OpenStack Compute and Image services work together to provide access to virtual servers and images through REST APIs. The Identity Service provides a common authorization layer for all OpenStack services. You must use the Identity Service to install the OpenStack Dashboard, which offers a web-based user interface for OpenStack components. The OpenStack Object Storage service provides not only a storage method for virtual images but also a cloud-based object storage system with a REST API to store and retrieve objects such as images or videos. This walk-through starts with Identity, then goes through Image and Compute and also provides deployment information about an Object Storage installation.

Here are the overall steps:

1. Review the most supported platforms. Red Hat Enterprise Linux, Scientific Linux, CentOS, Fedora, Debian, and Ubuntu are the most tested platforms currently.
2. Install the Identity Service (Keystone).
3. Configure the Identity Service.
4. Install the Image Service (Glance).
5. Configure the Image Service.
6. Install Compute (Nova).
7. Review the assumptions made in this installation for Compute.
8. Configure Compute with FlatDHCP networking using `192.168.100.0/24` as the fixed range for our guest VMs on a bridge named `br100`.
9. Create and initialize the Compute database with MySQL. PostgreSQL is also documented but all examples follow MySQL as an assumed default.
10. Add images.
11. (optional) Install OpenStack Object Storage (Swift).
12. Install the OpenStack Dashboard.
13. Launch the Dashboard.
14. Add a keypair through the Dashboard.
15. Launch an image through the Dashboard to verify the entire installation.

Compute and Image System Requirements

Hardware: OpenStack components are intended to run on standard hardware. Recommended hardware configurations for a minimum production deployment are as follows for the cloud controller nodes and compute nodes for Compute and the Image Service, and object, account, container, and proxy servers for Object Storage.

Table 1.1. Hardware Recommendations

Server	Recommended Hardware	Notes
Cloud Controller node (runs network, volume, API, scheduler and image services)	Processor: 64-bit x86 Memory: 12 GB RAM Disk space: 30 GB (SATA or SAS or SSD) Volume storage: two disks with 2 TB (SATA) for volumes attached to the compute nodes Network: one 1 GB Network Interface Card (NIC)	Two NICS are recommended but not required. A quad core server with 12 GB RAM would be more than sufficient for a cloud controller node. 32-bit processors will work for the cloud controller node. The package repositories referred to in this guide do not contain i386 packages.
Compute nodes (runs virtual instances)	Processor: 64-bit x86 Memory: 32 GB RAM Disk space: 30 GB (SATA) Network: two 1 GB NICs	Note that you cannot run 64-bit VM instances on a 32-bit compute node. A 64-bit compute node can run either 32- or 64-bit VMs, however. With 2 GB RAM you can run one m1.small instance on a node or three m1.tiny instances without memory swapping, so 2 GB RAM would be a minimum for a test-environment compute node. As an example, Rackspace Cloud Builders use 96 GB RAM for compute nodes in OpenStack deployments. Specifically for virtualization on certain hypervisors on the node or nodes running nova-compute, you need a x86 machine with an AMD processor with SVM extensions (also called AMD-V) or an Intel processor with VT (virtualization technology) extensions. For XenServer and XCP refer to the XenServer installation guide and the XenServer harware compatibility list. For LXC, the VT extensions are not required. The packages referred to in this guide do not contain i386 packages.

Note

While certain parts of OpenStack are known to work on various operating systems, currently the only feature-complete, production-supported host environment is Linux.

Operating System: OpenStack currently has packages for the following distributions: CentOS, Debian, Fedora, RHEL, and Ubuntu. These packages are maintained by community members, refer to http://wiki.openstack.org/Packaging for additional links.

Note

The Folsom release of OpenStack Compute requires Ubuntu 12.04 or later, as the version of libvirt that ships with Ubuntu 11.10 does not function properly with OpenStack due to bug #1011863.

Database: For OpenStack Compute, you need access to either a PostgreSQL or MySQL database, or you can install it as part of the OpenStack Compute installation process. For Object Storage, the container and account servers use SQLite, and you can install it as part of the installation process.

Permissions: You can install OpenStack Compute, the Image Service, or Object Storage either as root or as a user with sudo permissions if you configure the sudoers file to enable all the permissions.

Network Time Protocol: You must install a time synchronization program such as NTP. For Compute, time synchronization keeps your cloud controller and compute nodes talking to the same time server to avoid problems scheduling VM launches on compute nodes. For Object Storage, time synchronization ensure the object replications are accurately updating objects when needed so that the freshest content is served.

Compute Network Planning

For both conserving network resources and ensuring that network administrators understand the needs for networks and public IP addresses for accessing the APIs and VMs as necessary, this section offers recommendations and required minimum sizes. Throughput of at least 1000 Mbps is suggested. This walkthrough shows network configurations for a single server.

For OpenStack Compute, networking is configured on multi-node installations between the physical machines on a single subnet. For networking between virtual machine instances, three network options are available: flat, DHCP, and VLAN. Two NICs (Network Interface Cards) are recommended on the server running nova-network.

Management Network (RFC1918 IP Range, not publicly routable): This network is utilized for all inter-server communications within the cloud infrastructure. Recommended size: 255 IPs (CIDR /24)

Public Network (Publicly routable IP range): This network is utilized for providing Public IP accessibility to the API endpoints within the cloud infrastructure. Minimum size: 8 IPs (CIDR /29)

VM Network (RFC1918 IP Range, not publicly routable): This network is utilized for providing primary IP addresses to the cloud instances. Recommended size: 1024 IPs (CIDR /22)

Floating IP network (Publicly routable IP Range): This network is utilized for providing Public IP accessibility to selected cloud instances. Minimum size: 16 IPs (CIDR /28)

Installing Network Time Protocol (NTP)

To keep all the services in sync across multiple machines, you need to install NTP, and if you do a multi-node configuration you will configure one server to be the reference server.

```
$ sudo apt-get install -y ntp
```

Set up the NTP server on your controller node so that it receives data by modifying the `ntp.conf` file and restarting the service. As root:

```
# sed -i 's/server ntp.ubuntu.com/server ntp.ubuntu.com\nserver 127.127.1.0\
nfudge 127.127.1.0 stratum 10/g' /etc/ntp.conf
# service ntp restart
```

Set up the NTP client on your compute node so that the time between controller node and compute node is synchronized. Install the NTP package on the compute nodes in addition to the API nodes, to allow more gradual time updates.

2. OpenStack Terminology

Version Names

Each OpenStack release has a name, in increasing alphabetical order (e.g., Essex follows Diablo). There are also version numbers corresponding to these releases, but the numbering schemes are different for OpenStack Compute and OpenStack Object Storage, as shown in the table below:

Table 2.1. OpenStack version names

Release name	Release date	OpenStack Compute version number	OpenStack Object Storage version number
Folsom	October 2012	2012.2	1.7.2
Essex	April 2012	2012.1	1.4.8
Diablo	October 2011	2011.3	1.4.3
Cactus	April 2011	2011.2	1.3.0
Bexar	March 2011	2011.1	1.2.0
Austin	October 2010	0.9.0	1.0.0

Beginning with the Cactus release, OpenStack adopted a six month release schedule. The Folsom release is scheduled for October 2012.

Code Names

Each OpenStack service has a code name. For example, the Image Service is code-named Glance. The full list is shown in the table below:

Table 2.2. Code names

Service name	Code name
Identity	Keystone
Compute	Nova
Image	Glance
Dashboard	Horizon
Object Storage	Swift
Volumes	Cinder
Networking	Quantum

These code names are reflected in the names of configuration files and command-line utility programs. For example, the Identity service has a configuration file called `keystone.conf`.

OpenStack Services and Linux Services

In the Linux world, a service (also known as a daemon) refers to a single program that runs in the background and typically listens on a port to respond to service requests. An

OpenStack service, on the other hand, refers to a collection of Linux services working in concert.

OpenStack services are implemented by multiple Linux services. For example, **nova-compute** and **nova-scheduler** are two of the Linux services that implement the Compute service. OpenStack also depends on several third-party services, such as a database (typically MySQL) and a message broker (typically RabbitMQ or Qpid).

In this document, we generally use the term "service" to refer both to lower-level Linux services and higher-level OpenStack services. It should be clear from the context whether we are referring to a high-level OpenStack service (e.g., Image), or a low-level Linux service (e.g., **glance-api**).

Storage: objects, blocks, and files

Many cloud computing use cases require persistent remote storage. Storage solutions are often divided into three categories: object storage, block storage, and file storage.

Note that some storage solutions support multiple categories. For example, NexentaStor supports both block storage and file storage (with announcements for future support for object storage), GlusterFS supports file storage and object storage, and Ceph Storage supports object storage, block storage, and file storage.

Object storage

In OpenStack: Object Storage service (Swift)

Related concepts: Amazon S3, Rackspace Cloud Files, Ceph Storage

With *object storage*, files are exposed through an HTTP interface, typically with a REST API. All client data access is done at the user level: the operating system is unaware of the presence of the remote storage system. In OpenStack, the Object Storage service provides this type of functionality. Users access and modify files by making HTTP requests. Because the data access interface provided by an object storage system is at a low level of abstraction, people often build on top of object storage to build file-based applications that provide a higher level of abstraction. For example, the OpenStack Image service can be configured to use the Object Storage service as a backend. Another use for object storage solutions is as a content delivery network (CDN) for hosting static web content (e.g., images, and media files), since object storage already provides an HTTP interface.

Block storage (SAN)

In OpenStack: Volumes (`nova-volume` service) in OpenStack Compute or cinder as a separate Volume service

Related concepts: Amazon Elastic Block Store (EBS), Ceph RADOS Block Device (RBD), iSCSI

With *block storage*, files are exposed through a low-level computer bus interface such as SCSI or ATA, that is accessible over the network. Block storage is synonymous with SAN (storage area network). Clients access data through the operating system at the device level: users access the data by mounting the remote device in a similar manner to how

they would mount a local, physical disk (e.g., using the "mount" command in Linux). In OpenStack, the `nova-volume` service that forms part of the Compute service provides this type of functionality, and uses iSCSI to expose remote data as a SCSI disk that is attached to the network.

Because the data is exposed as a physical device, the end-user is responsible for creating partitions and formatting the exposed disk device. In addition, in OpenStack Compute a device can only be attached to one server at a time, so block storage cannot be used to share data across virtual machine instances concurrently.

File storage (NAS)

In OpenStack: none

Related concepts: NFS, Samba/CIFS, GlusterFS, Dropbox, Google Drive

With *file storage*, files are exposed through a distributed file system protocol. Filesystem storage is synonymous with NAS (network attached storage). Clients access data through the operating system at the file system level: users access the data by mounting a remote file system. Examples of file storage include NFS and GlusterFS. The operating system needs to have the appropriate client software installed to be able to access the remote file system.

Currently, OpenStack Compute does not have any native support for this type of file storage inside of an instance. However, there is a Gluster storage connector for OpenStack that enables the use of the GlusterFS file system as a back-end for the Image service.

3. Underlying Technologies

You can think of OpenStack Compute as a toolkit for building a cloud computing environment by stitching together existing Linux technologies.

The figures below shows two examples of how these underlying technologies can be assembled to construct an OpenStack Compute cloud. The circles are Linux services that are part of OpenStack Compute, and the rectangles are external (not maintained by the OpenStack project) components. Solid lines show interactions between OpenStack components and external components, and dashed lines show interactions between external components. All services that are part of OpenStack Compute interact with a queueing backend (e.g., RabbitMQ, Qpid) and a database backend (e.g., MySQL, PostgreSQL); these connections are not shown. Also not shown are services that do not explicitly rely on external technologies. For example, the `nova-api` service, the Identity service, and the Image service are not shown.

Figure 3.1. Underlying technologies (Scenario 1)

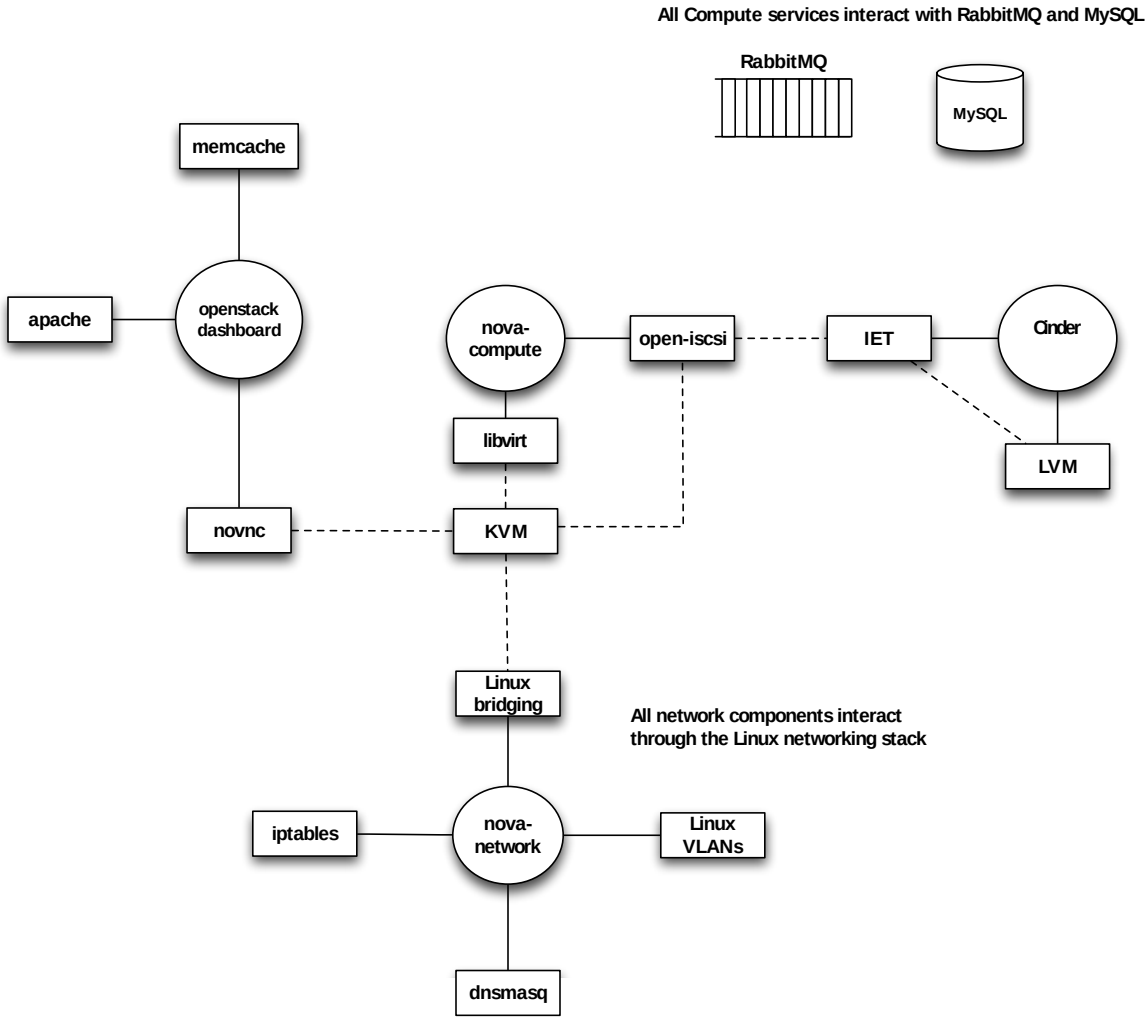
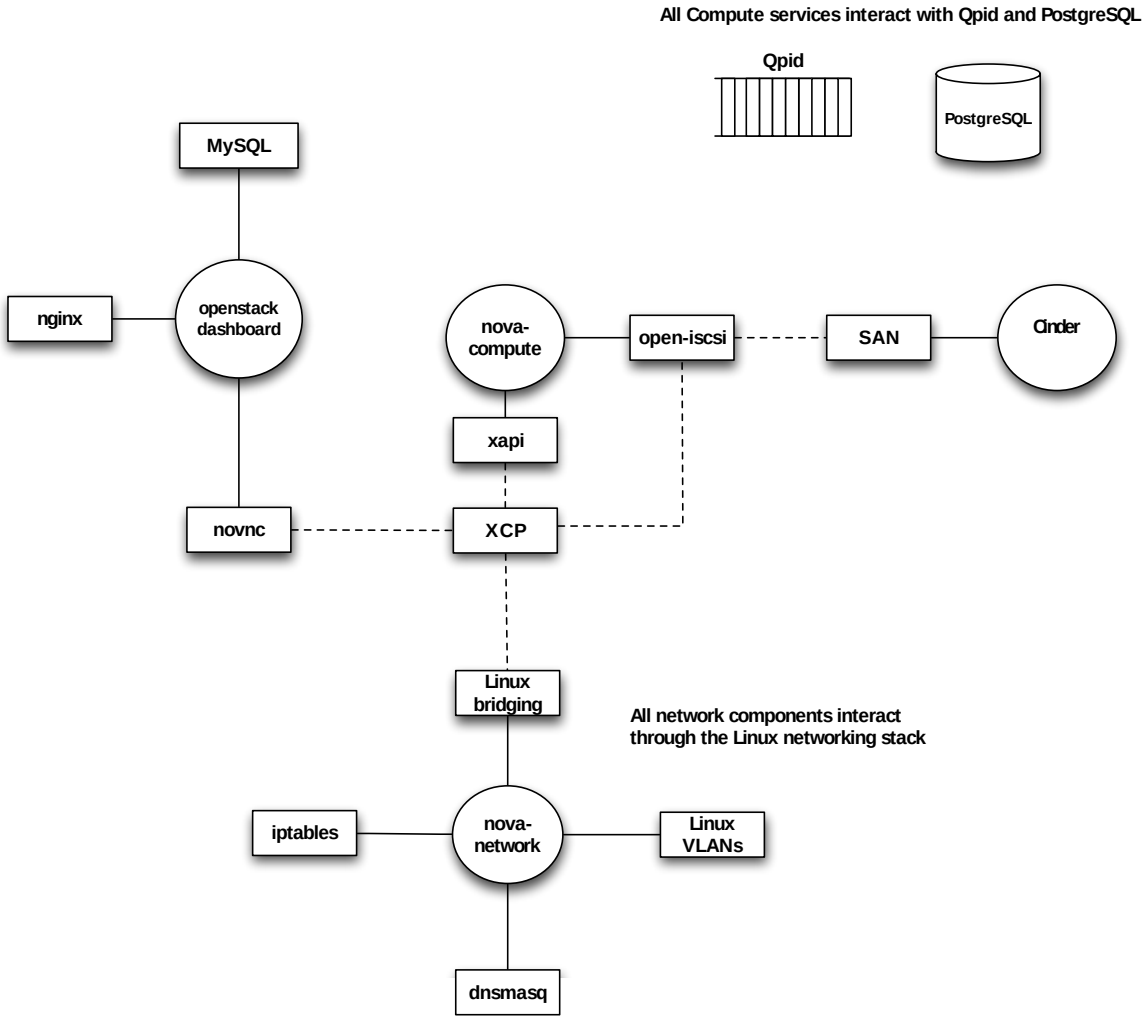


Figure 3.2. Underlying technologies (Scenario 2)



Many of the external technologies can be substituted with other components, as shown in the table below.

Table 3.1. Technologies and supported implementations

Technology	Supported implementations
Message queue	RabbitMQ, Qpid, ZeroMQ
Virtualization	xapi+XCP, xapi+XenServer, libvirt+KVM, libvirt+QEMU, libvirt+LXC, libvirt+VMWare
iSCSI back-end	LVM+IET, LVM+tgt, Xen Storage Manager, SAN (Solaris, HP, SolidFire), NexentaStor, NetApp, Ceph, Sheepdog
Database	MySQL, PostgreSQL, sqlite
Web server	Apache, Nginx
Session cache	memcache, any Django-supported database backend (e.g., MySQL, PostgreSQL, sqlite)

nova-compute

The `nova-compute` service depends on a virtualization driver to manage virtual machines. By default, this driver is *libvirt*, which is used to drive *KVM*. However, the libvirt driver can also drive other hypervisor technologies, and there is a separate Xen virtualization driver for driving Xen-based virtual machines if configured to use Xen Cloud Platform (XCP) or XenServer. *Open-iscsi* is used to mount remote block devices, also known as volumes. Open-iscsi exposes these remote devices as local device files which can be attached to instances.

nova-network

The `nova-network` service depends on a number of Linux networking technologies. It uses *Linux bridging* to create network bridges to connect virtual machines to the physical networks. These bridges may be associated with VLANs using *Linux networking VLAN support*, if running in the VLAN networking mode. *Iptables* is used to implement security rules and implement NAT functionality, which is used for providing instances with access to the metadata service and for supporting floating IP addresses. *Dnsmasq* is used as a DHCP server to hand out IP addresses to virtual machine instances, as well as a DNS server.

In a future OpenStack release, functionality that is currently implemented by `nova-network` will be available through a separate OpenStack project, codenamed Quantum.

Cinder

By default, `Cinder` service uses *LVM* to create and manage local volumes, and exports them via iSCSI using *IET* or *tgt*. It can also be configured to use other iSCSI-based storage technologies.

In the Folsom OpenStack release, functionality available through `nova-volume` is also available through a separate OpenStack Volumes project, code-named Cinder.

openstack-dashboard

The `openstack-dashboard` is a Django-based application that runs behind an *Apache* web server by default. It uses *memcache* for the session cache by default. A web-based VNC client called *novnc* is used to provide access to the VNC consoles associated with the running KVM instances.

4. Installation Assumptions

OpenStack Compute has a large number of configuration options. To simplify this installation guide, we make a number of assumptions about the target installation.

- You have a collection of compute nodes, each installed with Ubuntu Server 12.04.

Note

There is also an OpenStack Install and Deploy Manual for RHEL, CentOS and Fedora Debian, openSUSE, and SLES also have OpenStack support, but are not documented here.

- You have designated one of the nodes as the Cloud Controller, which will run all of the services (RabbitMQ or Qpid, MySQL, Identity, Image, nova-api, nova-network, nova-scheduler, nova-volume) except for nova-compute.
- The disk partitions on your cloud controller are being managed by the Logical Volume Manager (LVM).
- Your Cloud Controller has an LVM volume group named "nova-volumes" to provide persistent storage to guest VMs. Either create this during the installation or leave some free space to create it prior to installing nova services.
- Ensure that the server can resolve its own hostname, otherwise you may have problems if you are using RabbitMQ as the messaging backend. RabbitMQ is the default messaging back-end on Ubuntu
- `192.168.206.130` is the primary IP for our host on `eth0`.
- `192.168.100.0/24` as the fixed range for our guest VMs, connected to the host via `br100`.
- FlatDHCP with a single network interface.
- KVM or Xen (XenServer or XCP) as the hypervisor.
- On Ubuntu, enable the Cloud Archive repository by adding the following to /etc/apt/sources.list.d/folsom.list:

```
deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/folsom
 main
```

  Prior to running apt-get update and apt-get upgrade, install the keyring :

```
sudo apt-get install ubuntu-cloud-keyring
```

- Ensure the operating system is up-to-date by running **apt-get update** and **apt-get upgrade** prior to the installation.

This installation process walks through installing a cloud controller node and a compute node using a set of packages that are known to work with each other. The cloud controller node contains all the nova- services including the API server and the database server. The

compute node needs to run only the nova-compute service. You only need one nova-network service running in a multi-node install, though if high availability for networks is required, there are additional options.

Co-locating services

While in a best-practice deployment, each OpenStack project's services would live on a different machine, this is not always practical. For example, in small deployments there might be too few machines available, or a limited number of public IP addresses. Components from different OpenStack projects are not necessarily engineered to be able to be co-located, however many users report success with a variety of deployment scenarios.

The following is a series of pointers to be used when co-location of services from different OpenStack projects on the same machine is a must:

- Ensure dependencies aren't in conflict. The OpenStack Continuous Integration team does attempt to ensure there is no conflict - so if you see issues during package installation, consider filing a bug.
- Monitor your systems and ensure they are not overloaded. Some parts of OpenStack use a lot of CPU time (eg Swift Proxy Servers), while others are IO focused (eg Swift Object Server). Try to balance these so they complement each other.
- Beware of security. Different parts of OpenStack assume different security models. For example, Swift assumes the storage nodes will be on a private network and does not provide additonal security between nodes in the cluster.
- Ensure the ports you are running the services on don't conflict. Most ports used by OpenStack are configurable.

5. Installing OpenStack Identity Service

The OpenStack Identity service manages users, tenants (accounts or projects) and offers a common identity system for all the OpenStack components.

Basic Concepts

The Identity service has two primary functions:

1. User management: keep track of users and what they are permitted to do
2. Service catalog: Provide a catalog of what services are available and where their API endpoints are located

The Identity Service has several definitions which are important to understand.

User

A digital representation of a person, system, or service who uses OpenStack cloud services. Identity authentication services will validate that incoming request are being made by the user who claims to be making the call. Users have a login and may be assigned tokens to access resources. Users may be directly assigned to a particular tenant and behave as if they are contained in that tenant.

Credentials

Data that belongs to, is owned by, and generally only known by a user that the user can present to prove they are who they are (since nobody else should know that data).

Examples are:

- a matching username and password
- a matching username and API key
- yourself and a driver's license with a picture of you
- a token that was issued to you that nobody else knows of

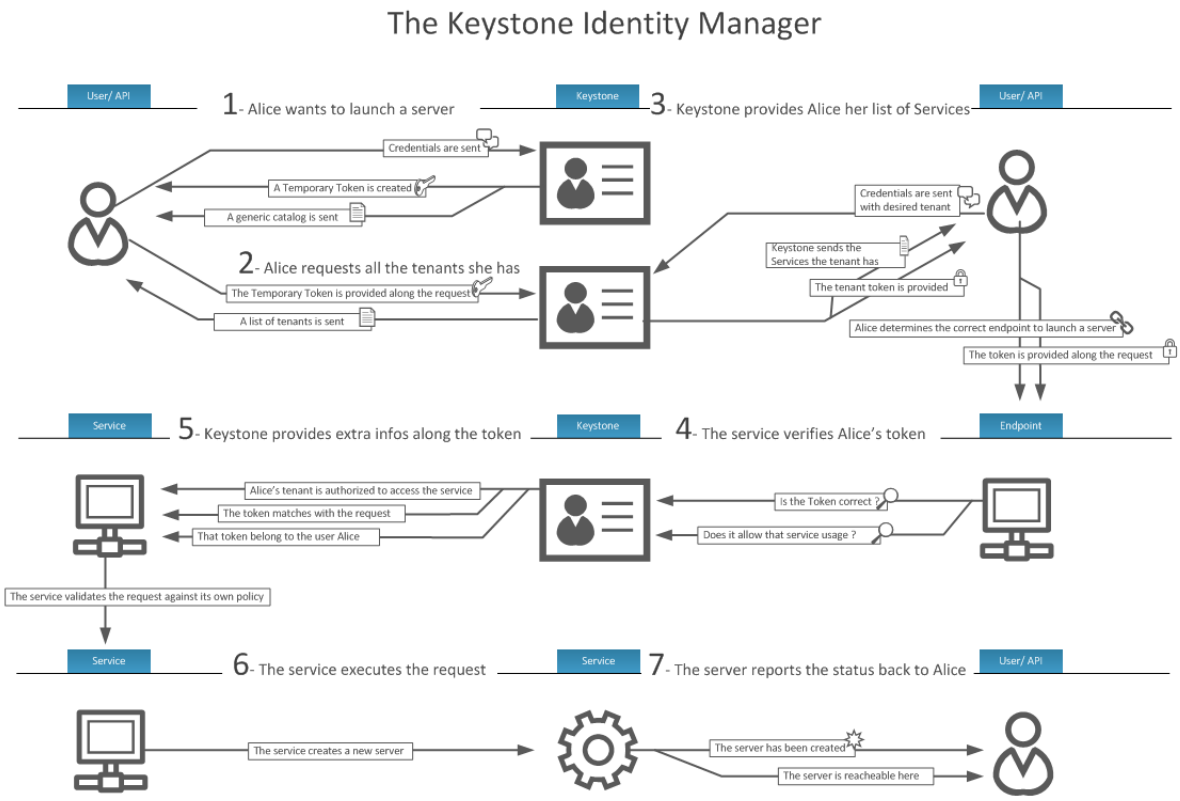
Authentication

In the context of the identity service, authentication is the act of confirming the identity of a user or the truth of a claim. The identity service will confirm that incoming request are being made by the user who claims to be making the call by validating a set of claims that the user is making. These claims are initially in the form of a set of credentials (username & password, or username and API key). After initial confirmation, the identity service will issue the user a token which the user can then provide to demonstrate that their identity has been authenticated when making subsequent requests.

Token

A token is an arbitrary bit of text that is used to access resources. Each token has a scope which describes which resources are accessible with it. A token may be revoked at anytime and is valid for a finite duration.

	While the identity service supports token-based authentication in this release, the intention is for it to support additional protocols in the future. The intent is for it to be an integration service foremost, and not a aspire to be a full-fledged identity store and management solution.
Tenant	A container used to group or isolate resources and/or identity objects. Depending on the service operator, a tenant may map to a customer, account, organization, or project.
Service	An OpenStack service, such as Compute (Nova), Object Storage (Swift), or Image Service (Glance). A service provides one or more endpoints through which users can access resources and perform (presumably useful) operations.
Endpoint	An network-accessible address, usually described by URL, where a service may be accessed. If using an extension for templates, you can create an endpoint template, which represents the templates of all the consumable services that are available across the regions.
Role	A personality that a user assumes when performing a specific set of operations. A role includes a set of right and privileges. A user assuming that role inherits those rights and privileges. In the identity service, a token that is issued to a user includes the list of roles that user can assume. Services that are being called by that user determine how they interpret the set of roles a user has and which operations or resources each roles grants access to.



User management

The three main concepts of Identity user management are:

- Users
- Tenants
- Roles

A *user* represents a human user, and has associated information such as username, password and email. This example creates a user named "alice":

```
$ keystone user-create --name=alice --pass=mypassword123 --email=
alice@example.com
```

A *tenant* can be thought of as a project, group, or organization. Whenever you make requests to OpenStack services, you must specify a tenant. For example, if you query the Compute service for a list of running instances, you will receive a list of all of the running instances in the tenant you specified in your query. This example creates a tenant named "acme":

```
$ keystone tenant-create --name=acme
```

Note

Because the term *project* was used instead of *tenant* in earlier versions of OpenStack Compute, some command-line tools use `--project_id` instead of `--tenant-id` or `--os-tenant-id` to refer to a tenant ID.

A *role* captures what operations a user is permitted to perform in a given tenant. This example creates a role named "compute-user":

```
$ keystone role-create --name=compute-user
```

Note

It is up to individual services such as the Compute service and Image service to assign meaning to these roles. As far as the Identity service is concerned, a role is simply a name.

The Identity service associates a user with a tenant and a role. To continue with our previous examples, we may wish to assign the "alice" user the "compute-user" role in the "acme" tenant:

```
$ keystone user-list

    +--------+---------+-------------------+--------+
    |   id   | enabled |       email       |  name  |
    +--------+---------+-------------------+--------+
    | 892585 |   True  | alice@example.com | alice  |
    +--------+---------+-------------------+--------+

$ keystone role-list

    +--------+--------------+
    |   id   |     name     |
    +--------+--------------+
    | 9a764e | compute-user |
    +--------+--------------+

$ keystone tenant-list

    +--------+------+---------+
    |   id   | name | enabled |
    +--------+------+---------+
    | 6b8fd2 | acme |   True  |
    +--------+------+---------+

$ keystone user-role-add --user=892585 --role=9a764e --tenant-id=6b8fd2
```

A user can be assigned different roles in different tenants: for example, Alice may also have the "admin" role in the "Cyberdyne" tenant. A user can also be assigned multiple roles in the same tenant.

The `/etc/[SERVICE_CODENAME]/policy.json` controls what users are allowed to do for a given service. For example, `/etc/nova/policy.json` specifies the access policy for the Compute service, `/etc/glance/policy.json` specifies the access policy for the

Image service, and `/etc/keystone/policy.json` specifies the access policy for the Identity service.

The default `policy.json` files in the Compute, Identity, and Image service recognize only the `admin` role: all operations that do not require the `admin` role will be accessible by any user that has any role in a tenant.

If you wish to restrict users from performing operations in, say, the Compute service, you need to create a role in the Identity service and then modify `/etc/nova/policy.json` so that this role is required for Compute operations.

For example, this line in `/etc/nova/policy.json` specifies that there are no restrictions on which users can create volumes: if the user has any role in a tenant, they will be able to create volumes in that tenant.

```
            "volume:create": [],
```

If we wished to restrict creation of volumes to users who had the `compute-user` role in a particular tenant, we would add `"role:compute-user"`, like so:

```
            "volume:create": ["role:compute-user"],
```

If we wished to restrict all Compute service requests to require this role, the resulting file would look like:

```
            {
                "admin_or_owner":  [["role:admin"], ["project_id:
%(project_id)s"]],
                "default": [["rule:admin_or_owner"]],

                "compute:create": ["role":"compute-user"],
                "compute:create:attach_network": ["role":"compute-user"],
                "compute:create:attach_volume": ["role":"compute-user"],
                "compute:get_all": ["role":"compute-user"],

                "admin_api": [["role:admin"]],
                "compute_extension:accounts": [["rule:admin_api"]],
                "compute_extension:admin_actions": [["rule:admin_api"]],
                "compute_extension:admin_actions:pause":
 [["rule:admin_or_owner"]],
                "compute_extension:admin_actions:unpause":
 [["rule:admin_or_owner"]],
                "compute_extension:admin_actions:suspend":
 [["rule:admin_or_owner"]],
                "compute_extension:admin_actions:resume":
 [["rule:admin_or_owner"]],
                "compute_extension:admin_actions:lock": [["rule:admin_api"]],
                "compute_extension:admin_actions:unlock":
 [["rule:admin_api"]],
                "compute_extension:admin_actions:resetNetwork":
 [["rule:admin_api"]],
                "compute_extension:admin_actions:injectNetworkInfo":
 [["rule:admin_api"]],
```

```
                "compute_extension:admin_actions:createBackup":
 [["rule:admin_or_owner"]],
                "compute_extension:admin_actions:migrateLive":
 [["rule:admin_api"]],
                "compute_extension:admin_actions:migrate":
 [["rule:admin_api"]],
                "compute_extension:aggregates": [["rule:admin_api"]],
                "compute_extension:certificates": ["role":"compute-user"],
                "compute_extension:cloudpipe": [["rule:admin_api"]],
                "compute_extension:console_output": ["role":"compute-user"],
                "compute_extension:consoles": ["role":"compute-user"],
                "compute_extension:createserverext": ["role":"compute-user"],
                "compute_extension:deferred_delete": ["role":"compute-user"],
                "compute_extension:disk_config": ["role":"compute-user"],
                "compute_extension:extended_server_attributes":
 [["rule:admin_api"]],
                "compute_extension:extended_status": ["role":"compute-user"],
                "compute_extension:flavorextradata": ["role":"compute-user"],
                "compute_extension:flavorextraspecs": ["role":"compute-user"],
                "compute_extension:flavormanage": [["rule:admin_api"]],
                "compute_extension:floating_ip_dns": ["role":"compute-user"],
                "compute_extension:floating_ip_pools": ["role":"compute-
user"],
                "compute_extension:floating_ips": ["role":"compute-user"],
                "compute_extension:hosts": [["rule:admin_api"]],
                "compute_extension:keypairs": ["role":"compute-user"],
                "compute_extension:multinic": ["role":"compute-user"],
                "compute_extension:networks": [["rule:admin_api"]],
                "compute_extension:quotas": ["role":"compute-user"],
                "compute_extension:rescue": ["role":"compute-user"],
                "compute_extension:security_groups": ["role":"compute-user"],
                "compute_extension:server_action_list": [["rule:admin_api"]],
                "compute_extension:server_diagnostics": [["rule:admin_api"]],
                "compute_extension:simple_tenant_usage:show":
 [["rule:admin_or_owner"]],
                "compute_extension:simple_tenant_usage:list":
 [["rule:admin_api"]],
                "compute_extension:users": [["rule:admin_api"]],
                "compute_extension:virtual_interfaces": ["role":"compute-
user"],
                "compute_extension:virtual_storage_arrays": ["role":"compute-
user"],
                "compute_extension:volumes": ["role":"compute-user"],
                "compute_extension:volumetypes": ["role":"compute-user"],

                "volume:create": ["role":"compute-user"],
                "volume:get_all": ["role":"compute-user"],
                "volume:get_volume_metadata": ["role":"compute-user"],
                "volume:get_snapshot": ["role":"compute-user"],
                "volume:get_all_snapshots": ["role":"compute-user"],

                "network:get_all_networks": ["role":"compute-user"],
                "network:get_network": ["role":"compute-user"],
                "network:delete_network": ["role":"compute-user"],
                "network:disassociate_network": ["role":"compute-user"],
                "network:get_vifs_by_instance": ["role":"compute-user"],
                "network:allocate_for_instance": ["role":"compute-user"],
                "network:deallocate_for_instance": ["role":"compute-user"],
                "network:validate_networks": ["role":"compute-user"],
```

```
                "network:get_instance_uuids_by_ip_filter": ["role":"compute-
user"],

                "network:get_floating_ip": ["role":"compute-user"],
                "network:get_floating_ip_pools": ["role":"compute-user"],
                "network:get_floating_ip_by_address": ["role":"compute-user"],
                "network:get_floating_ips_by_project": ["role":"compute-
user"],
                "network:get_floating_ips_by_fixed_address": ["role":"compute-
user"],
                "network:allocate_floating_ip": ["role":"compute-user"],
                "network:deallocate_floating_ip": ["role":"compute-user"],
                "network:associate_floating_ip": ["role":"compute-user"],
                "network:disassociate_floating_ip": ["role":"compute-user"],

                "network:get_fixed_ip": ["role":"compute-user"],
                "network:add_fixed_ip_to_instance": ["role":"compute-user"],
                "network:remove_fixed_ip_from_instance": ["role":"compute-
user"],
                "network:add_network_to_project": ["role":"compute-user"],
                "network:get_instance_nw_info": ["role":"compute-user"],

                "network:get_dns_domains": ["role":"compute-user"],
                "network:add_dns_entry": ["role":"compute-user"],
                "network:modify_dns_entry": ["role":"compute-user"],
                "network:delete_dns_entry": ["role":"compute-user"],
                "network:get_dns_entries_by_address": ["role":"compute-user"],
                "network:get_dns_entries_by_name": ["role":"compute-user"],
                "network:create_private_dns_domain": ["role":"compute-user"],
                "network:create_public_dns_domain": ["role":"compute-user"],
                "network:delete_dns_domain": ["role":"compute-user"]
            }
```

Service management

The two main concepts of Identity service management are:

- Services
- Endpoints

The Identity service also maintains a user that corresponds to each service (e.g., a user named *nova*, for the Compute service) and a special service tenant, which is called *service*.

The commands for creating services and endpoints are described in a later section.

Installing and Configuring the Identity Service

Install the Identity service on any server that is accessible to the other servers you intend to use for OpenStack services, as root:

```
# apt-get install keystone
```

After installing, you need to delete the sqlite database it creates, then change the configuration to point to a MySQL database. This configuration enables easier scaling

scenarios since you can bring up multiple Keystone front ends when needed, and configure them all to point back to the same database. Plus a database backend has built-in data replication features and documentation surrounding high availability and data redundancy configurations.

Delete the `keystone.db` file created in the `/var/lib/keystone` directory.

```
# rm /var/lib/keystone/keystone.db
```

Configure the production-ready backend data store rather than using the catalog supplied by default for the ability to backup the service and endpoint data. This example shows MySQL.

Install MySQL as root:

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

During the install, you'll be prompted for the mysql root password. Enter a password of your choice and verify it.

Use **sed** to edit `/etc/mysql/my.cnf` to change `bind-address` from localhost (127.0.0.1) to any (0.0.0.0) and restart the mysql service, as root.

```
# sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf
# service mysql restart
```

The following sequence of commands will create a MySQL database named "keystone" and a MySQL user named "keystone" with full access to the "keystone" MySQL database.

On Fedora, RHEL, and CentOS, you can configure the Keystone database with the **openstack-db** command.

To manually create the database, start the **mysql** command line client by running:

```
$ mysql -u root -p
```

Enter the mysql root user's password when prompted.

To configure the MySQL database, create the keystone database.

```
mysql> CREATE DATABASE keystone;
```

Create a MySQL user for the newly-created keystone database that has full control of the keystone database.

Note

Choose a secure password for the keystone user and replace all references to *[YOUR_KEYSTONEDB_PASSWORD]* with this password.

```
mysql> GRANT ALL ON keystone.* TO 'keystone'@'%' IDENTIFIED BY
 '[YOUR_KEYSTONEDB_PASSWORD]';
```

Enter quit at the `mysql>` prompt to exit MySQL.

```
mysql> quit
```

Reminder

Recall that this document assumes the Cloud Controller node has an IP address of `192.168.206.130`.

Once Keystone is installed, it is configured via a primary configuration file (`/etc/keystone/keystone.conf`), and by initializing data into keystone using the command line client. By default, Keystone's data store is sqlite. To change the data store to mysql, change the line defining "connection" in `/etc/keystone/keystone.conf` like so:

```
connection = mysql://keystone:[YOUR_KEYSTONEDB_PASSWORD]@192.168.206.130/
keystone
```

Also, ensure that the proper service token is used in the `keystone.conf` file. An example is provided in the Appendix or you can generate a random string. The sample token is:

```
admin_token = 012345SECRET99TOKEN012345
```

Next, restart the keystone service so that it picks up the new database configuration.

```
# sudo service keystone restart
```

Lastly, initialize the new keystone database, as root:

```
# keystone-manage db_sync
```

Configuring Services to work with Keystone

Once Keystone is installed and running, you set up users and tenants and services to be configured to work with it. You can either follow the manual steps or use a script.

Note

The parameters –os-token and –os-endpoint are valid for the keystoneclient available after October 2012. Use –token and –endpoint with the keystoneclient released with the Folsom packaging. This install guide documents installing the client from packages, but you can use the client from another computer with the CLI Guide instructions for pip install.

Setting up tenants, users, and roles - manually

You need to minimally define a tenant, user, and role to link the tenant and user as the most basic set of details to get other services authenticating and authorizing with the Identity service.

Scripted method available

These are the manual, unscripted steps using the keystone client. A scripted method is available at Setting up tenants, users, and roles - scripted.

First, create a default tenant, we'll name it `demo` in this example.

```
$ keystone --os-token 012345SECRET99TOKEN012345 --os-endpoint http://192.168.
206.130:35357/v2.0 tenant-create --name demo --description "Default Tenant"
```

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | Default Tenant                   |
| enabled     | True                             |
| id          | b5815b046cfe47bb891a7b64119e7f80 |
| name        | demo                             |
+-------------+----------------------------------+
```

Create a default user named `admin`.

```
$ keystone --os-token 012345SECRET99TOKEN012345 --os-endpoint http://192.168.
206.130:35357/v2.0 user-create --tenant-id b5815b046cfe47bb891a7b64119e7f80 --
name admin --pass secretword
```

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| email    |                                  |
| enabled  | True                             |
| id       | a4c2d43f80a549a19864c89d759bb3fe |
| name     | admin                            |
| tenantId | b5815b046cfe47bb891a7b64119e7f80 |
+----------+----------------------------------+
```

Create an administrative role based on keystone's default `policy.json` file, `admin`.

```
$ keystone --os-token 012345SECRET99TOKEN012345 --os-endpoint http://192.168.
206.130:35357/v2.0 role-create --name admin
```

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| id       | e3d9d157cc95410ea45d23bbbc2e5c10 |
| name     | admin                            |
+----------+----------------------------------+
```

Grant the `admin` role to the `admin` user in the `demo` tenant with "user-role-add".

```
$ keystone --os-token 012345SECRET99TOKEN012345 --os-endpoint
 http://192.168.206.130:35357/v2.0 user-role-add --user-id
 a4c2d43f80a549a19864c89d759bb3fe --tenant-id b5815b046cfe47bb891a7b64119e7f80
 --role-id e3d9d157cc95410ea45d23bbbc2e5c10
```

There is no output to this command.

Create a Service Tenant. This tenant contains all the services that we make known to the service catalog.

```
$ keystone --os-token 012345SECRET99TOKEN012345 --os-endpoint http://192.168.
206.130:35357/v2.0 tenant-create --name service --description "Service Tenant"
```

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | Service Tenant                   |
| enabled     | True                             |
| id          | eb7e0c10a99446cfa14c244374549e9d |
| name        | service                          |
+-------------+----------------------------------+
```

Create a Glance Service User in the Service Tenant. You'll do this for any service you add to be in the Keystone service catalog.

```
$ keystone --os-token 012345SECRET99TOKEN012345 --os-endpoint http://192.168.
206.130:35357/v2.0 user-create --tenant-id eb7e0c10a99446cfa14c244374549e9d --
name glance --pass glance
```

```
     +----------+----------------------------------+
     | Property |              Value               |
     +----------+----------------------------------+
     | email    |                                  |
     | enabled  | True                             |
     | id       | 46b2667a7807483d983e0b4037a1623b |
     | name     | glance                           |
     | tenantId | eb7e0c10a99446cfa14c244374549e9d |
     +----------+----------------------------------+
```

Grant the `admin` role to the `glance` user in the `service` tenant.

```
$ keystone --os-token 012345SECRET99TOKEN012345 --os-endpoint
 http://192.168.206.130:35357/v2.0 user-role-add --user-id
 46b2667a7807483d983e0b4037a1623b --tenant-id eb7e0c10a99446cfa14c244374549e9d
 --role-id e3d9d157cc95410ea45d23bbbc2e5c10
```

There is no output to this command.

Create a Nova Service User in the Service Tenant.

```
$ keystone --os-token 012345SECRET99TOKEN012345 --os-endpoint http://192.168.
206.130:35357/v2.0 user-create --tenant-id eb7e0c10a99446cfa14c244374549e9d --
name nova --pass nova
```

```
     +----------+----------------------------------+
     | Property |              Value               |
     +----------+----------------------------------+
     | email    |                                  |
     | enabled  | True                             |
     | id       | 54b3776a8707834d983e0b4037b1345c |
     | name     | nova                             |
     | tenantId | eb7e0c10a99446cfa14c244374549e9d |
     +----------+----------------------------------+
```

Grant the `admin` role to the `nova` user in the `service` tenant.

```
$ keystone --os-token 012345SECRET99TOKEN012345 --os-endpoint
 http://192.168.206.130:35357/v2.0 user-role-add --user-id
 54b3776a8707834d983e0b4037b1345c --tenant-id eb7e0c10a99446cfa14c244374549e9d
 --role-id e3d9d157cc95410ea45d23bbbc2e5c10
```

There is no output to this command.

Create an EC2 Service User in the Service Tenant.

```
$ keystone --os-token 012345SECRET99TOKEN012345 --os-endpoint http://192.168.
206.130:35357/v2.0 user-create --tenant-id eb7e0c10a99446cfa14c244374549e9d --
name ec2 --pass ec2
```

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| email    |                                  |
| enabled  | True                             |
| id       | 32e7668b8707834d983e0b4037b1345c |
| name     | ec2                              |
| tenantId | eb7e0c10a99446cfa14c244374549e9d |
+----------+----------------------------------+
```

Grant the `admin` role to the `ec2` user in the `service` tenant.

```
$ keystone --os-token 012345SECRET99TOKEN012345 --os-endpoint
 http://192.168.206.130:35357/v2.0 user-role-add --user-id
 32e7668b8707834d983e0b4037b1345c --tenant-id eb7e0c10a99446cfa14c244374549e9d
 --role-id e3d9d157cc95410ea45d23bbbc2e5c10
```

There is no output to this command.

Create an Object Storage Service User in the Service Tenant.

```
$ keystone --os-token 012345SECRET99TOKEN012345 --os-endpoint http://192.168.
206.130:35357/v2.0 user-create --tenant-id eb7e0c10a99446cfa14c244374549e9d --
name swift --pass swiftpass
```

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| email    |                                  |
| enabled  | True                             |
| id       | 4346677b8909823e389f0b4037b1246e |
| name     | swift                            |
| tenantId | eb7e0c10a99446cfa14c244374549e9d |
+----------+----------------------------------+
```

Grant the `admin` role to the `swift` user in the `service` tenant.

```
$ keystone --os-token 012345SECRET99TOKEN012345 --os-endpoint
 http://192.168.206.130:35357/v2.0 user-role-add --user-id
 4346677b8909823e389f0b4037b1246e --tenant-id eb7e0c10a99446cfa14c244374549e9d
 --role-id e3d9d157cc95410ea45d23bbbc2e5c10
```

There is no output to this command.

Next you create definitions for the services.

Defining Services

Keystone also acts as a service catalog to let other OpenStack systems know where relevant API endpoints exist for OpenStack Services. The OpenStack Dashboard, in particular, uses the service catalog heavily - and this **must** be configured for the OpenStack Dashboard to properly function.

There are two alternative ways of defining services with keystone:

1. Using a template file

2. Using a database backend

While using a template file is simpler, it is not recommended except for development environments such as DevStack. The template file does not enable CRUD operations on the service catalog through keystone commands, but you can use the service-list command when using the template catalog. A database backend can provide better reliability, availability, and data redundancy. This section describes how to populate the Keystone service catalog using the database backend. Your `/etc/keystone/keystone.conf` file should contain the following lines if it is properly configured to use the database backend.

```
[catalog]
driver = keystone.catalog.backends.sql.Catalog
```

Elements of a Keystone service catalog entry

For each service in the catalog, you must perform two keystone operations:

1. Use the **keystone service-create** command to create a database entry for the service, with the following attributes:

   `--name` Name of the service (e.g., `nova`, `ec2`, `glance`, `keystone`)

   `--type` Type of service (e.g., `compute`, `ec2`, `image`, `identity`)

   `--description` A description of the service, (e.g., `"Nova Compute Service"`)

2. Use the **keystone endpoint-create** command to create a database entry that describes how different types of clients can connect to the service, with the following attributes:

   `--region` the region name you've given to the OpenStack cloud you are deploying (e.g., RegionOne)

   `--service-id` The ID field returned by the **keystone service-create** (e.g., `935fd37b6fa74b2f9fba6d907fa95825`)

   `--publicurl` The URL of the public-facing endpoint for the service (e.g., `http://192.168.206.130:9292/v1` or `http://192.168.206.130:8774/v2/%(tenant_id)s`)

   `--internalurl` The URL of an internal-facing endpoint for the service.

   This typically has the same value as `publicurl`.

   `--adminurl` The URL for the admin endpoint for the service. The Keystone and EC2 services use different endpoints for `adminurl` and `publicurl`, but for other services these endpoints will be the same.

Keystone allows some URLs to contain special variables, which are automatically substituted with the correct value at runtime. Some examples in this document employ the `tenant_id` variable, which we use when specifying the Volume and Compute service endpoints. Variables can be specified using either `%(`*`varname`*`)s` or `$(`*`varname`*`)s` notation. In this document, we always use the `%(`*`varname`*`)s` notation (e.g., `%(tenant_id)s`) since $ is interpreted as a special character by Unix shells.

Creating keystone services and service endpoints

Here we define the services and their endpoints.

Define the Identity service:

```
$ keystone --os-token 012345SECRET99TOKEN012345 \
 --os-endpoint http://192.168.206.130:35357/v2.0/ \
 service-create \
 --name=keystone \
 --type=identity \
 --description="Keystone Identity Service"

+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | Keystone Identity Service        |
| id          | 15c11a23667e427e91bc31335b45f4bd |
| name        | keystone                         |
| type        | identity                         |
+-------------+----------------------------------+
$ keystone --os-token 012345SECRET99TOKEN012345 \
 --os-endpoint http://192.168.206.130:35357/v2.0/ \
 endpoint-create \
 --region RegionOne \
 --service-id=15c11a23667e427e91bc31335b45f4bd \
 --publicurl=http://192.168.206.130:5000/v2.0 \
 --internalurl=http://192.168.206.130:5000/v2.0 \
 --adminurl=http://192.168.206.130:35357/v2.0
+-------------+----------------------------------+
|   Property  |             Value                |
+-------------+----------------------------------+
| adminurl    | http://192.168.206.130:35357/v2.0 |
| id          | 11f9c625a3b94a3f8e66bf4e5de2679f  |
| internalurl | http://192.168.206.130:5000/v2.0  |
| publicurl   | http://192.168.206.130:5000/v2.0  |
| region      | RegionOne                         |
| service_id  | 15c11a23667e427e91bc31335b45f4bd  |
+-------------+----------------------------------+
```

Define the Compute service, which requires a separate endpoint for each tenant. Here we use the `service` tenant from the previous section.

Note

The `%(tenant_id)s` and single quotes around the `publicurl`, `internalurl`, and `adminurl` must be typed exactly as shown for both the Compute endpoint and the Volume endpoint.

```
$ keystone --os-token 012345SECRET99TOKEN012345 \
--os-endpoint http://192.168.206.130:35357/v2.0/ \
service-create \
 --name=nova \
 --type=compute \
 --description="Nova Compute Service"
```

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | Nova Compute Service             |
| id          | abc0f03c02904c24abdcc3b7910e2eed |
| name        | nova                             |
| type        | compute                          |
+-------------+----------------------------------+

$ keystone --os-token 012345SECRET99TOKEN012345 \
--os-endpoint http://192.168.206.130:35357/v2.0/ \
endpoint-create \
 --region RegionOne \
 --service-id=abc0f03c02904c24abdcc3b7910e2eed \
 --publicurl='http://192.168.206.130:8774/v2/%(tenant_id)s' \
 --internalurl='http://192.168.206.130:8774/v2/%(tenant_id)s' \
 --adminurl='http://192.168.206.130:8774/v2/%(tenant_id)s'
+-------------+-----------------------------------------+
|   Property  |                                Value    |
+-------------+-----------------------------------------+
| adminurl    | http://192.168.206.130:8774/v2/%(tenant_id)s |
| id          | 935fd37b6fa74b2f9fba6d907fa95825         |
| internalurl | http://192.168.206.130:8774/v2/%(tenant_id)s |
| publicurl   | http://192.168.206.130:8774/v2/%(tenant_id)s |
| region      | RegionOne                               |
| service_id  | abc0f03c02904c24abdcc3b7910e2eed         |
+-------------+-----------------------------------------+
```

Define the Volume service, which also requires a separate endpoint for each tenant.

```
$ keystone --os-token 012345SECRET99TOKEN012345 \
 --os-endpoint http://192.168.206.130:35357/v2.0/ \
service-create \
 --name=volume \
 --type=volume \
 --description="Nova Volume Service"


+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | Nova Volume Service              |
| id          | 1ff4ece13c3e48d8a6461faebd9cd38f |
| name        | volume                           |
| type        | volume                           |
+-------------+----------------------------------+

$ keystone --os-token 012345SECRET99TOKEN012345 \
 --os-endpoint http://192.168.206.130:35357/v2.0/ \
 endpoint-create \
 --region RegionOne \
 --service-id=1ff4ece13c3e48d8a6461faebd9cd38f \
 --publicurl='http://192.168.206.130:8776/v1/%(tenant_id)s' \
 --internalurl='http://192.168.206.130:8776/v1/%(tenant_id)s' \
 --adminurl='http://192.168.206.130:8776/v1/%(tenant_id)s'


+-------------+-----------------------------------------+
```

```
|   Property  |                  Value                    |
+-------------+-------------------------------------------+
| adminurl    | http://192.168.206.130:8776/v1/%(tenant_id)s |
| id          | 1ff4ece13c3e48d8a6461faebd9cd38f          |
| internalurl | http://192.168.206.130:8776/v1/%(tenant_id)s |
| publicurl   | http://192.168.206.130:8776/v1/%(tenant_id)s |
| region      | RegionOne                                 |
| service_id  | 8a70cd235c7d4a05b43b2dffb9942cc0          |
+-------------+-------------------------------------------+
```

Define the Image service:

```
$ keystone --os-token 012345SECRET99TOKEN012345 \
 --os-endpoint http://192.168.206.130:35357/v2.0/ \
 service-create \
 --name=glance \
 --type=image \
 --description="Glance Image Service"


+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | Glance Image Service             |
| id          | 7d5258c490144c8c92505267785327c1 |
| name        | glance                           |
| type        | image                            |
+-------------+----------------------------------+

$ keystone --os-token 012345SECRET99TOKEN012345 \
 --os-endpoint http://192.168.206.130:35357/v2.0/ \
 endpoint-create \
 --region RegionOne \
 --service-id=7d5258c490144c8c92505267785327c1 \
 --publicurl=http://192.168.206.130:9292/v1 \
 --internalurl=http://192.168.206.130:9292/v1 \
 --adminurl=http://192.168.206.130:9292/v1


+-------------+----------------------------------+
|   Property  |             Value                |
+-------------+----------------------------------+
| adminurl    | http://192.168.206.130:9292/v1   |
| id          | 3c8c0d749f21490b90163bfaed9befe7 |
| internalurl | http://192.168.206.130:9292/v1   |
| publicurl   | http://192.168.206.130:9292/v1   |
| region      | RegionOne                        |
| service_id  | 7d5258c490144c8c92505267785327c1 |
+-------------+----------------------------------+
```

Define the EC2 compatibility service:

```
$ keystone --os-token 012345SECRET99TOKEN012345 \
 --os-endpoint http://192.168.206.130:35357/v2.0/ \
 service-create \
 --name=ec2 \
 --type=ec2 \
```

```
 --description="EC2 Compatibility Layer"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | EC2 Compatibility Layer          |
| id          | 181cdad1d1264387bcc411e1c6a6a5fd |
| name        | ec2                              |
| type        | ec2                              |
+-------------+----------------------------------+

$ keystone --os-token 012345SECRET99TOKEN012345 \
 --os-endpoint http://192.168.206.130:35357/v2.0/ \
 endpoint-create \
 --region RegionOne \
 --service-id=181cdad1d1264387bcc411e1c6a6a5fd \
 --publicurl=http://192.168.206.130:8773/services/Cloud \
 --internalurl=http://192.168.206.130:8773/services/Cloud \
 --adminurl=http://192.168.206.130:8773/services/Admin


+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
| adminurl    | http://192.168.206.130:8773/services/Admin |
| id          | d2a3d7490c61442f9b2c8c8a2083c4b6        |
| internalurl | http://192.168.206.130:8773/services/Cloud |
| publicurl   | http://192.168.206.130:8773/services/Cloud |
| region      | RegionOne                               |
| service_id  | 181cdad1d1264387bcc411e1c6a6a5fd        |
+-------------+-----------------------------------------+
```

Define the Object Storage service:

```
$ keystone  --os-token 012345SECRET99TOKEN012345 \
 --os-endpoint http://192.168.206.130:35357/v2.0/ \
 service-create \
 --name=swift \
 --type=object-store \
 --description="Object Storage Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | Object Storage Service           |
| id          | 272efad2d1234376cbb911c1e5a5a6ed |
| name        | swift                            |
| type        | object-store                     |
+-------------+----------------------------------+

$ keystone --os-token 012345SECRET99TOKEN012345 \
 --os-endpoint http://192.168.206.130:35357/v2.0/ \
 endpoint-create \
 --region RegionOne \
 --service-id=272efad2d1234376cbb911c1e5a5a6ed \
 --publicurl 'http://192.168.206.130:8888/v1/AUTH_%(tenant_id)s' \
 --adminurl 'http://192.168.206.130:8888/v1' \
 --internalurl 'http://192.168.206.130:8888/v1/AUTH_%(tenant_id)s'
```

```
+-------------+----------------------------------------------------+
|   Property  |                       Value                        |
+-------------+----------------------------------------------------+
| adminurl    | http://192.168.206.130:8888/v1                     |
| id          | e32b3c4780e51332f9c128a8c208a5a4                   |
| internalurl | http://192.168.206.130:8888/v1/AUTH_%(tenant_id)s  |
| publicurl   | http://192.168.206.130:8888/v1/AUTH_%(tenant_id)s  |
| region      | RegionOne                                          |
| service_id  | 272efad2d1234376cbb911c1e5a5a6ed                   |
+-------------+----------------------------------------------------+
```

Setting up Tenants, Users, Roles, and Services - Scripted

The Keystone project offers a bash script for populating tenants, users, roles and services at https://github.com/openstack/keystone/blob/master/tools/sample_data.sh with sample data. This script uses 127.0.0.1 for all endpoint IP addresses. This script also defines services for you.

Troubleshooting the Identity Service (Keystone)

To begin troubleshooting, look at the logs in the `/var/log/keystone.log` file (the location of log files is configured in the `/etc/keystone/logging.conf` file). It shows all the components that have come in to the WSGI request, and will ideally have an error in that log that explains why an authorization request failed. If you're not seeing the request at all in those logs, then run keystone with "–debug" where –debug is passed in directly after the CLI command prior to parameters.

Verifying the Identity Service Installation

Verify that authentication is behaving as expected by using your established username and password to generate an authentication token:

```
       $ keystone --os-username=admin --os-password=secretword --os-auth-url=
http://192.168.206.130:35357/v2.0 token-get

+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| expires  |       2012-10-04T16:08:03Z       |
|    id    | 960ad732a0eb4b2a88516f18384c1fba |
| user_id  | a4c2d43f80a549a19864c89d759bb3fe |
+----------+----------------------------------+
```

You should receive a token in response, paired with your user ID.

This verifies that keystone is running on the expected endpoint, and that your user account is established with the expected credentials.

Next, verify that authorization is behaving as expected by requesting authorization on a tenant:

```
       $ keystone --os-username=admin --os-password=secretword --os-tenant-
name=demo --os-auth-url=http://192.168.206.130:35357/v2.0 token-get

+-----------+----------------------------------+
|  Property |              Value               |
+-----------+----------------------------------+
|  expires  |       2012-10-04T16:10:14Z       |
|     id    | 8787f264d2a34607b37aa8d58d956afa |
| tenant_id | c1ac0f7f0e55448fa3940fa6b8b54911 |
|  user_id  | a4c2d43f80a549a19864c89d759bb3fe |
+-----------+----------------------------------+
```

You should receive a new token in response, this time including the ID of the tenant you specified.

This verifies that your user account has an explicitly defined role on the specified tenant, and that the tenant exists as expected.

You can also set your `--os-*` variables in your environment to simplify CLI usage. First, set up a `keystonerc` file with the admin credentials and admin endpoint:

```
       export OS_USERNAME=admin
       export OS_PASSWORD=secretword
       export OS_TENANT_NAME=demo
       export OS_AUTH_URL=http://192.168.206.130:35357/v2.0
```

Save and source the file.

```
$ source keystonerc
```

Verify that your `keystonerc` is configured correctly by performing the same command as above, but without any `--os-*` arguments.

```
       $ keystone token-get

+-----------+----------------------------------+
|  Property |              Value               |
+-----------+----------------------------------+
|  expires  |       2012-10-04T16:12:38Z       |
|     id    | 03a13f424b56440fb39278b844a776ae |
| tenant_id | c1ac0f7f0e55448fa3940fa6b8b54911 |
|  user_id  | a4c2d43f80a549a19864c89d759bb3fe |
+-----------+----------------------------------+
```

You should receive a new token in response, reflecting the same tenant and user ID values as above.

This verifies that you have configured your environment variables correctly.

Finally, verify that your admin account has authorization to perform administrative commands.

Reminder

Unlike basic authentication/authorization, which can be performed against either port 5000 or 35357, administrative commands MUST be performed against the admin API port: 35357). This means that you MUST use port 35357 in your OS_AUTH_URL or `--os-auth-url` setting.

```
       $ keystone user-list

+----------------------------------+---------+-------+--------+
|                id                | enabled | email |  name  |
+----------------------------------+---------+-------+--------+
| 318003c9a97342dbab6ff81675d68364 | True    | None  | swift  |
| 3a316b32f44941c0b9ebc577feaa5b5c | True    | None  | nova   |
| ac4dd12ebad84e55a1cd964b356ddf65 | True    | None  | glance |
| a4c2d43f80a549a19864c89d759bb3fe | True    | None  | admin  |
| ec47114af7014afd9a8994cbb6057a8b | True    | None  | ec2    |
+----------------------------------+---------+-------+--------+
```

This verifies that your user account has the `admin` role, as defined in keystone's `policy.json` file.

6. Installing OpenStack Compute and Image Service

The OpenStack Compute and Image services work together to provide access to virtual servers and images through REST APIs.

Installing and Configuring the Image Service

Install the Image service, as root:

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

Note

10/10/12: When using the Ubuntu Cloud Archive, you need to re-install the python-keystoneclient after installing the glance packages listed above, otherwise you see an error.

After installing, you need to delete the sqlite database it creates, then change the configuration to point to the MySQL database.

Delete the `glance.sqlite` file created in the `/var/lib/glance/` directory.

```
# rm /var/lib/glance/glance.sqlite
```

Configuring the Image Service database backend

Configure the backend data store. For MySQL, create a glance MySQL database and a glance MySQL user. Grant the "glance" user full access to the glance MySQL database.

Start the MySQL command line client by running:

```
$ mysql -u root -p
```

Enter the MySQL root user's password when prompted.

To configure the MySQL database, create the glance database.

```
mysql> CREATE DATABASE glance;
```

Create a MySQL user for the newly-created glance database that has full control of the database.

```
mysql> GRANT ALL ON glance.* TO 'glance'@'%' IDENTIFIED BY
 '[YOUR_GLANCEDB_PASSWORD]';
```

Enter `quit` at the `mysql>` prompt to exit MySQL.

```
mysql> quit
```

Edit the Glance configuration files and paste ini middleware files

The Image service has a number of options that you can use to configure the Glance API server, optionally the Glance Registry server, and the various storage backends that Glance can use to store images. By default, the storage backend is in file, specified in the glance-api.conf config file in the section [DEFAULT].

The glance-api service implements versions 1 and 2 of the OpenStack Images API. By default, both are enabled by setting these configuration options to "true" in the `glance-api.conf` file.

```
enable_v1_api=True
```

```
enable_v2_api=True
```

Disable either version of the Images API by setting the option to False in the `glance-api.conf` file.

Note

In order to use the v2 API, you must copy the necessary SQL configuration from your glance-registry service to your glance-api configuration file. The following instructions assume that you want to use the v2 Image API for your installation. The v1 API is implemented on top of the glance-registry service while the v2 API is not.

Most configuration is done via configuration files, with the Glance API server (and possibly the Glance Registry server) using separate configuration files. When installing through an operating system package management system, sample configuration files are installed in /etc/glance. You configure the PasteDeploy configuration, which controls the deployment of the WSGI application for each component, in files named *component*-paste.ini, such as glance-api-paste.ini.

This walkthrough installs the image service using a file backend and the Identity service (Keystone) for authentication.

Update `/etc/glance/glance-api-paste.ini` and configure the `admin_*` values under `[filter:authtoken]`.

```
[filter:authtoken]
admin_tenant_name = service
admin_user = glance
admin_password = glance
```

Add the admin and service identifiers and flavor=keystone to the end of `/etc/glance/glance-api.conf` as shown below.

```
[keystone_authtoken]
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = glance
```

```
[paste_deploy]
# Name of the paste configuration file that defines the available pipelines
config_file = /etc/glance/glance-api-paste.ini

# Partial name of a pipeline in your paste configuration file with the
# service name removed. For example, if your paste section name is
# [pipeline:glance-api-keystone], you would configure the flavor below
# as 'keystone'.
flavor=keystone
```

Ensure that `/etc/glance/glance-api.conf` points to the MySQL database rather than sqlite.

```
sql_connection = mysql://glance:[YOUR_GLANCEDB_PASSWORD]@192.168.206.130/
glance
```

Restart glance-api to pick up these changed settings.

```
service glance-api restart
```

Update the last sections of `/etc/glance/glance-registry.conf` to reflect the values you set earlier for admin user and the service tenant, plus enable the Identity service with flavor=keystone.

```
[keystone_authtoken]
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = glance

[paste_deploy]
# Name of the paste configuration file that defines the available pipelines
config_file = /etc/glance/glance-registry-paste.ini

# Partial name of a pipeline in your paste configuration file with the
# service name removed. For example, if your paste section name is
# [pipeline:glance-api-keystone], you would configure the flavor below
# as 'keystone'.
flavor=keystone
```

Update `/etc/glance/glance-registry-paste.ini` by enabling the Identity service, keystone:

```
# Use this pipeline for keystone auth
[pipeline:glance-registry-keystone]
pipeline = authtoken context registryapp
```

Ensure that `/etc/glance/glance-registry.conf` points to the MySQL database rather than sqlite.

```
sql_connection = mysql://glance:[YOUR_GLANCEDB_PASSWORD]@192.168.206.130/
glance
```

Restart glance-registry to pick up these changed settings.

```
service glance-registry restart
```

Note

Any time you change the .conf files, restart the corresponding service.

On Ubuntu 12.04, the database tables are under version control and you must do these steps on a new install to prevent the Image service from breaking possible upgrades, as root:

```
# glance-manage version_control 0
```

Now you can populate or migrate the database.

```
# glance-manage db_sync
```

Restart glance-registry and glance-api services, as root:

```
# service glance-registry restart
# service glance-api restart
```

Note

This guide does not configure image caching, refer to http://docs.openstack.org/developer/glance/ for more information.

Troubleshooting the Image Service (Glance)

To begin troubleshooting, look at the logs in the `/var/log/glance/registry.log` or `/var/log/glance/api.log`.

Verifying the Image Service Installation

To validate the Image service client is installed, enter **`glance help`** at the command line.

Obtain a test image.

```
mkdir /tmp/images
    cd /tmp/images/
    wget http://smoser.brickies.net/ubuntu/ttylinux-uec/ttylinux-uec-amd64-12.
1_2.6.35-22_1.tar.gz
    tar -zxvf ttylinux-uec-amd64-12.1_2.6.35-22_1.tar.gz
```

Upload the kernel.

Note

This example shows inputting –os-username, –os-password, –os-tenant-name, –os-auth-url on the command line for reference. You could also use the OS_* environment variables by setting them in an openrc file:

```
export OS_USERNAME=admin
export OS_TENANT_NAME=openstackDemo
export OS_PASSWORD=secretword
export OS_AUTH_URL=http://192.168.206.130:5000/v2.0/
export OS_REGION_NAME=RegionOne
```

Then you would source these environment variables by running **`source openrc`**.

```
glance --os-username=admin --os-password=secretword --os-tenant-name=demo --
os-auth-url=http://192.168.206.130:5000/v2.0 \
image-create \
--name="tty-linux-kernel" \
--disk-format=aki \
--container-format=aki < ttylinux-uec-amd64-12.1_2.6.35-22_1-vmlinuz
```

```
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 79adac8c671790504a142ab2392cf927     |
| container_format | aki                                  |
| created_at       | 2012-11-01T20:48:25                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | aki                                  |
| id               | 599907ff-296d-4042-a671-d015e34317d2 |
| is_public        | False                                |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | tty-linux-kernel                     |
| owner            | 71bcb9b723c04eeb9b5ace0ac9ce418b     |
| protected        | False                                |
| size             | 4731440                              |
| status           | active                               |
| updated_at       | 2012-11-01T20:48:25                  |
+------------------+--------------------------------------+
```

Upload the initrd.

```
glance --os-username=admin --os-password=secretword --os-tenant-name=demo --
os-auth-url=http://192.168.206.130:5000/v2.0 \
image-create \
--name="tty-linux-ramdisk" \
--disk-format=ari \
--container-format=ari < ttylinux-uec-amd64-12.1_2.6.35-22_1-loader
```

```
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 0754dbc6abcbfec42b345948654a05ad     |
| container_format | ari                                  |
| created_at       | 2012-11-01T20:48:27                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | ari                                  |
| id               | 7d9f0378-1640-4e43-8959-701f248d999d |
| is_public        | False                                |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | tty-linux-ramdisk                    |
| owner            | 71bcb9b723c04eeb9b5ace0ac9ce418b     |
| protected        | False                                |
| size             | 2254249                              |
| status           | active                               |
| updated_at       | 2012-11-01T20:48:27                  |
+------------------+--------------------------------------+
```

Upload the image.

```
glance --os-username=admin --os-password=secretword --os-tenant-name=demo --
os-auth-url=http://192.168.206.130:5000/v2.0 \
image-create \
--name="tty-linux" \
--disk-format=ami \
--container-format=ami \
--property kernel_id=599907ff-296d-4042-a671-d015e34317d2 \
--property ramdisk_id=7d9f0378-1640-4e43-8959-701f248d999d < ttylinux-uec-
amd64-12.1_2.6.35-22_1.img
```

```
+----------------------+--------------------------------------+
| Property             | Value                                |
+----------------------+--------------------------------------+
| Property 'kernel_id' | 599907ff-296d-4042-a671-d015e34317d2 |
| Property 'ramdisk_id' | 7d9f0378-1640-4e43-8959-701f248d999d |
| checksum             | 2f81976cae15c16ef0010c51e3a6c163     |
| container_format     | ami                                  |
| created_at           | 2012-11-01T20:48:30                  |
| deleted              | False                                |
| deleted_at           | None                                 |
| disk_format          | ami                                  |
| id                   | 21b421e5-44d4-4903-9db0-4f134fdd0793 |
| is_public            | False                                |
| min_disk             | 0                                    |
| min_ram              | 0                                    |
| name                 | tty-linux                            |
| owner                | 71bcb9b723c04eeb9b5ace0ac9ce418b     |
| protected            | False                                |
| size                 | 25165824                             |
| status               | active                               |
| updated_at           | 2012-11-01T20:48:30                  |
+----------------------+--------------------------------------+
```

Now a glance image-list should show three images.

```
glance --os-username=admin --os-password=secretword --os-tenant-name=demo --
os-auth-url=http://192.168.206.130:5000/v2.0 image-list
+--------------------------------------+---------------------------------
+-------------+-----------------+----------+--------+
| ID                                   | Name                            |
 Disk Format | Container Format | Size     | Status |
+--------------------------------------+---------------------------------
+-------------+-----------------+----------+--------+
| 21b421e5-44d4-4903-9db0-4f134fdd0793 | tty-linux                       | ami
         | ami              | 25165824 | active |
| 7d9f0378-1640-4e43-8959-701f248d999d | tty-linux-ramdisk               | ari
         | ari              | 2254249  | active |
| 599907ff-296d-4042-a671-d015e34317d2 | tty-linux-kernel                | aki
         | aki              | 4731440  | active |
+--------------------------------------+---------------------------------
+-------------+-----------------+----------+--------+
```

Configuring the Hypervisor

For production environments the most tested hypervisors are KVM and Xen-based hypervisors. KVM runs through libvirt, Xen runs best through XenAPI calls. KVM is selected

by default and requires the least additional configuration. This guide offers information on both but the specific walkthrough configuration options set up KVM.

KVM

KVM is configured as the default hypervisor for Compute.

Note

There are several sections about hypervisor selection in this document. If you are reading this document linearly, you do not want to load the KVM module prior to installing nova-compute. The nova-compute service depends on qemu-kvm which installs `/lib/udev/rules.d/45-qemu-kvm.rules`, which sets the correct permissions on the /dev/kvm device node.

To enable KVM explicitly, add the following configuration options `/etc/nova/nova.conf`:

```
compute_driver=libvirt.LibvirtDriver
libvirt_type=kvm
```

The KVM hypervisor supports the following virtual machine image formats:

- Raw
- QEMU Copy-on-write (qcow2)
- VMWare virtual machine disk format (vmdk)

The rest of this section describes how to enable KVM on your system. You may also wish to consult distribution-specific documentation:

- Fedora: Getting started with virtualization from the Fedora project wiki.
- Ubuntu: KVM/Installation from the Community Ubuntu documentation.
- Debian: Virtualization with KVM from the Debian handbook.
- RHEL: Installing virtualization packages on an existing Red Hat Enterprise Linux system from the Red Hat Enterprise Linux Virtualization Host Configuration and Guest Installation Guide.
- openSUSE: Installing KVM from the openSUSE Virtualization with KVM manual.
- SLES: Installing KVM from the SUSE Linux Enterprise Server Virtualization with KVM manual.

Checking for hardware virtualization support

The processors of your compute host need to support virtualization technology (VT) to use KVM.

If you are running on Ubuntu, use the **kvm-ok** command to check if your processor has VT support, it is enabled in the BIOS, and KVM is installed properly, as root:

```
# kvm-ok
```

If KVM is enabled, the output should look something like:

```
INFO: /dev/kvm exists
KVM acceleration can be used
```

If KVM is not enabled, the output should look something like:

```
INFO: Your CPU does not support KVM extensions
KVM acceleration can NOT be used
```

In the case that KVM acceleration is not supported, Compute should be configured to use a different hypervisor, such as QEMU or Xen.

On distributions that don't have **kvm-ok**, you can check if your processor has VT support by looking at the processor flags in the `/proc/cpuinfo` file. For Intel processors, look for the `vmx` flag, and for AMD processors, look for the `svm` flag. A simple way to check is to run the following command and see if there is any output:

```
$ egrep '(vmx|svm)' --color=always /proc/cpuinfo
```

Some systems require that you enable VT support in the system BIOS. If you believe your processor supports hardware acceleration but the above command produced no output, you may need to reboot your machine, enter the system BIOS, and enable the VT option.

Enabling KVM

KVM requires the `kvm` and either `kvm-intel` or `kvm-amd` modules to be loaded. This may have been configured automatically on your distribution when KVM is installed.

You can check that they have been loaded using **lsmod**, as follows, with expected output for Intel-based processors:

```
$ lsmod | grep kvm
kvm_intel             137721  9
kvm                   415459  1 kvm_intel
```

The following sections describe how to load the kernel modules for Intel-based and AMD-based processors if they were not loaded automatically by your distribution's KVM installation process.

Intel-based processors

If your compute host is Intel-based, run the following as root to load the kernel modules:

```
# modprobe kvm
# modprobe kvm-intel
```

Add the following lines to `/etc/modules` so that these modules will load on reboot:

```
kvm
kvm-intel
```

AMD-based processors

If your compute host is AMD-based, run the following as root to load the kernel modules:

```
# modprobe kvm
# modprobe kvm-amd
```

Add the following lines to `/etc/modules` so that these modules will load on reboot:

```
kvm
kvm-amd
```

Specifying the CPU model of KVM guests

The Compute service allows you to control the guest CPU model that is exposed to KVM virtual machines. Use cases include:

- To maximise performance of virtual machines by exposing new host CPU features to the guest
- To ensure a consistent default CPU across all machines, removing reliance of variable QEMU defaults

In libvirt, the CPU is specified by providing a base CPU model name (which is a shorthand for a set of feature flags), a set of additional feature flags, and the topology (sockets/cores/threads). The libvirt KVM driver provides a number of standard CPU model names. Examples of model names include:

```
"486", "pentium", "pentium2", "pentiumpro", "coreduo", "n270",
"pentiumpro", "qemu32", "kvm32", "cpu64-rhel5", "cpu64-rhel5",
"kvm64", "pentiumpro", "Conroe" "Penryn", "Nehalem", "Westmere",
"pentiumpro", "cpu64-rhel5", "cpu64-rhel5", "Opteron_G1",
"Opteron_G2", "Opteron_G3", "Opteron_G4"
```

These models are defined in the file `/usr/share/libvirt/cpu_map.xml`. Check this file to determine which models are supported by your local installation.

There are two Compute configuration options that determine the type of CPU model exposed to the hypervisor when using KVM, `libvirt_cpu_mode` and `libvirt_cpu_model`.

The `libvirt_cpu_mode` option can take one of four values: `none`, `host-passthrough`, `host-model` and `custom`.

Host model (default for KVM & QEMU)

If your `nova.conf` contains `libvirt_cpu_mode=host-model`, libvirt will identify the CPU model in `/usr/share/libvirt/cpu_map.xml` which most closely matches the host, and then request additional CPU flags to complete the match. This should give close to maximum functionality/performance, which maintaining good reliability/compatibility if the guest is migrated to another host with slightly different host CPUs.

Host passthrough

If your `nova.conf` contains `libvirt_cpu_mode=host-passthrough`, libvirt will tell KVM to passthrough the host CPU with no modifications. The difference to host-model, instead of just matching feature flags, every last detail of the host CPU is matched. This gives absolutely best performance, and can be important to some apps which check low level CPU details, but it comes at a cost with respect to migration: the guest can only be migrated to an exactly matching host CPU.

Custom

If your nova.conf contains `libvirt_cpu_mode=custom`, you can explicitly specify one of the supported named model using the libvirt_cpu_model configuration option. For example, to configure the KVM guests to expose Nehalem CPUs, your nova.conf should contain:

```
libvirt_cpu_mode=custom
libvirt_cpu_model=Nehalem
```

None (default for all libvirt-driven hypervisors other than KVM & QEMU)

If your nova.conf contains `libvirt_cpu_mode=none`, then libvirt will not specify any CPU model at all. It will leave it up to the hypervisor to choose the default model. This setting is equivalent to the Compute service behavior prior to the Folsom release.

Troubleshooting

Trying to launch a new virtual machine instance fails with the ERR0R state, and the following error appears in `/var/log/nova/nova-compute.log`

```
libvirtError: internal error no supported architecture for os type 'hvm'
```

This is a symptom that the KVM kernel modules have not been loaded.

If you cannot start VMs after installation without rebooting, it's possible the permissions are not correct. This can happen if you load the KVM module before you've installed nova-compute. To check the permissions, run **`ls -l /dev/kvm`** to see whether the group is set to kvm. If not, run **`sudo udevadm trigger`**.

QEMU

From the perspective of the Compute service, the QEMU hypervisor is very similar to the KVM hypervisor. Both are controlled through libvirt, both support the same feature set, and all virtual machine images that are compatible with KVM are also compatible with QEMU. The main difference is that QEMU does not support native virtualization. Consequently, QEMU has worse performance than KVM and is a poor choice for a production deployment.

The typical uses cases for QEMU are

- Running on older hardware that lacks virtualization support.
- Running the Compute service inside of a virtual machine for development or testing purposes, where the hypervisor does not support native virtualization for guests.

KVM requires hardware support for acceleration. If hardware support is not available (e.g., if you are running Compute inside of a VM and the hypervisor does not expose the required hardware support), you can use QEMU instead. KVM and QEMU have the same level of support in OpenStack, but KVM will provide better performance. To enable QEMU:

```
compute_driver=libvirt.LibvirtDriver
libvirt_type=qemu
```

For some operations you may also have to install the **guestmount** utility:

```
$> sudo apt-get install guestmount
```

The QEMU hypervisor supports the following virtual machine image formats:

- Raw
- QEMU Copy-on-write (qcow2)
- VMWare virtual machine disk format (vmdk)

Xen, XenAPI, XenServer and XCP

The recommended way to use Xen with OpenStack is through the XenAPI driver. To enable the XenAPI driver, add the following configuration options `/etc/nova/nova.conf`:

```
compute_driver=xenapi.XenAPIDriver
xenapi_connection_url=http://your_xenapi_management_ip_address
xenapi_connection_username=root
xenapi_connection_password=your_password
```

The above connection details are used by the OpenStack Compute service to contact your hypervisor and are the same details you use to connect XenCenter, the XenServer management console, to your XenServer or XCP box. Note these settings are generally unique to each hypervisor host as the use of the host internal management network ip address (169.254.0.1) will cause features such as live-migration to break.

OpenStack with XenAPI supports the following virtual machine image formats:

- Raw
- VHD (in a gzipped tarball)

It is possible to manage Xen using libvirt. This would be necessary for any Xen-based system that isn't using the XCP toolstack, such as SUSE Linux or Oracle Linux. Unfortunately, this is not well tested or supported as of the Essex release. To experiment using Xen through libvirt add the following configuration options `/etc/nova/nova.conf`:

```
compute_driver=libvirt.LibvirtDriver
libvirt_type=xen
```

The rest of this section describes Xen, XCP, and XenServer, the differences between them, and how to use them with OpenStack. Xen's architecture is different from KVM's in important ways, and we discuss those differences and when each might make sense in your OpenStack cloud.

Xen terminology

Xen is a hypervisor. It provides the fundamental isolation between virtual machines. Xen is open source (GPLv2) and is managed by Xen.org, an cross-industry organization.

Xen is a component of many different products and projects. The hypervisor itself is very similar across all these projects, but the way that it is managed can be different, which can cause confusion if you're not clear which toolstack you are using. Make sure you know what toolstack you want before you get started.

Xen Cloud Platform (XCP) is an open source (GPLv2) toolstack for Xen. It is designed specifically as platform for enterprise and cloud computing, and is well integrated with OpenStack. XCP is available both as a binary distribution, installed from an iso, and from Linux distributions, such as xcp-xapi in Ubuntu. The current versions of XCP avalible in linux distributions do not yet include all the features available in the binary distribution of XCP.

Citrix XenServer is a commercial product. It is based on XCP, and exposes the same toolstack and managment API. As an analogy, think of XenServer being based on XCP in the way that Red Hat Enterprise Linux is based on Fedora. XenServer has a free version (which is very similar to XCP) and paid-for versions with additional features enabled. Citrix provide support for XenServer, but as of July 2012, they do not provide any support for XCP. For a comparison between these products see the XCP Feature Matrix.

Both XenServer and XCP include Xen, Linux, and the primary control daemon known as **xapi**.

The API shared between XCP and XenServer is called **XenAPI**. OpenStack usually refers to XenAPI, to indicate that the integration works equally well on XCP and XenServer. Sometimes, a careless person will refer to XenServer specifically, but you can be reasonably confident that anything that works on XenServer will also work on the latest version of XCP. Read the XenAPI Object Model Overview for definitions of XenAPI specifc terms such as SR, VDI, VIF and PIF.

Privileged and unprivileged domains

A Xen host will run a number of virtual machines, VMs, or domains (the terms are synonymous on Xen). One of these is in charge of running the rest of the system, and is known as "domain 0", or "dom0". It is the first domain to boot after Xen, and owns the storage and networking hardware, the device drivers, and the primary control software. Any other VM is unprivileged, and are known as a "domU" or "guest". All customer VMs are unprivileged of course, but you should note that on Xen the OpenStack control software (nova-compute) also runs in a domU. This gives a level of security isolation between the privileged system software and the OpenStack sofware (much of which is customer-facing). This architecture is described in more detail later.

There is an ongoing project to split domain 0 into multiple privileged domains known as **driver domains** and **stub domains**. This would give even better separation between critical components. This technology is what powers Citrix XenClient RT, and is likly to be added into XCP in the next few years. However, the current architecture just has three levels of separation: dom0, the OpenStack domU, and the completely unprivileged customer VMs.

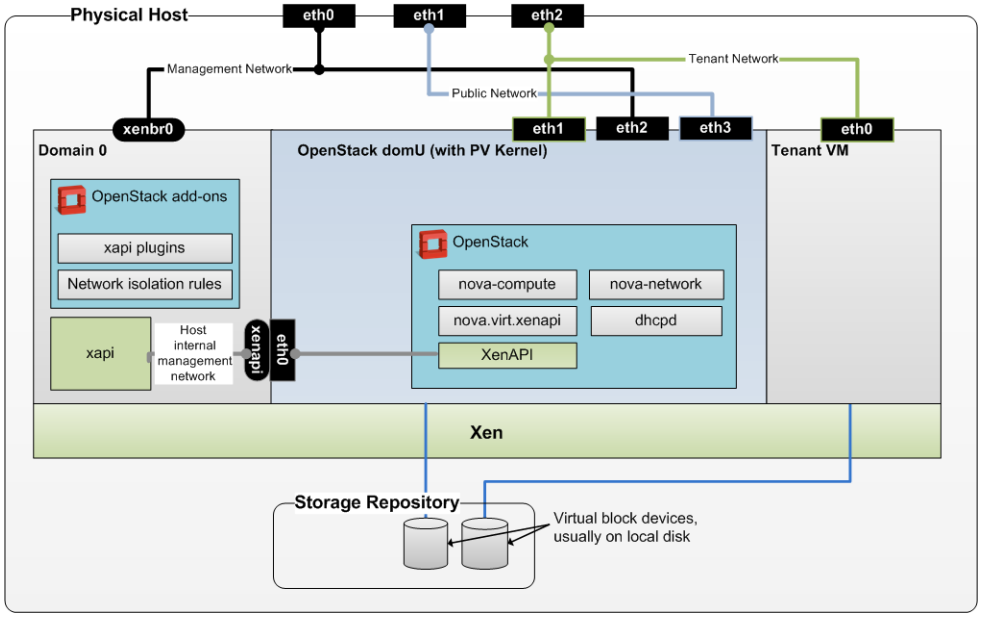
Paravirtualized versus hardware virtualized domains

A Xen virtual machine can be **paravirtualized (PV)** or **hardware virtualized (HVM)**. This refers to the interaction between Xen, domain 0, and the guest VM's kernel. PV guests are aware of the fact that they are virtualized and will co-operate with Xen and domain 0; this gives them better performance characteristics. HVM guests are not aware of their environment, and the hardware has to pretend that they are running on an unvirtualized machine. HVM guests have the advantage that there is no need to modify the guest operating system, which is essential when running Windows.

In OpenStack, customer VMs may run in either PV or HVM mode. However, the OpenStack domU (that's the one running nova-compute) **must** be running in PV mode.

XenAPI deployment architecture

When you deploy OpenStack on XCP or XenServer you will get something similar to this:



Key things to note:

- The hypervisor: Xen
- Domain 0: runs xapi and some small pieces from OpenStack (some xapi plugins and network isolation rules). The majority of this is provided by XenServer or XCP (or yourself using Kronos).
- OpenStack domU: The nova-compute code runs in a paravirtualized virtual machine, running on the host under management. Each host runs a local instance of nova-compute. It will often also be running nova-network (depending on your network mode). In this case, nova-network is managing the addresses given to the tenant VMs through DHCP.
- Nova uses the XenAPI Python library to talk to xapi, and it uses the Host Internal Management Network to reach from the domU to dom0 without leaving the host.

Some notes on the networking:

- The above diagram assumes FlatDHCP networking (the DevStack default).
- There are three main OpenStack networks: Management traffic (RabbitMQ, MySQL, etc), Tenant network traffic (controlled by nova-network) and Public traffic (floating IPs, public API end points).
- Each network that leaves the host has been put through a separate physical network interface. This is the simplest model, but it's not the only one possible. You may choose to isolate this traffic using VLANs instead, for example.

XenAPI pools

Before OpenStack 2012.1 ("Essex"), all XenServer machines used with OpenStack are standalone machines, usually only using local storage.

However in 2012.1 and later, the host-aggregates feature allows you to create pools of XenServer hosts (configuring shared storage is still an out of band activity). This move will enable live migration when using shared storage.

Installing XenServer and XCP

When you want to run OpenStack with XCP or XenServer, you first need to install the software on an appropriate server. Please note, Xen is a type 1 hypervisor. This means when your server starts the first software that runs is Xen. This means the software you install on your compute host is XenServer or XCP, not the operating system you wish to run the OpenStack code on. The OpenStack services will run in a VM you install on top of XenServer.

Before you can install your system you must decide if you want to install Citrix XenServer (either the free edition, or one of the paid editions) or Xen Cloud Platform from Xen.org. You can download the software from the following locations:

- http://www.citrix.com/XenServer/download
- http://www.xen.org/download/xcp/index.html

When installing many servers, you may find it easier to perform PXE boot installations of XenServer or XCP. You can also package up any post install changes you wish to make to your XenServer by creating your own XenServer supplimental pack.

It is also possible to get XCP by installing the **xcp-xenapi** package on Debian based distributions. However, this is not as mature or feature complete as above distributions. This will modify your boot loader to first boot Xen, then boot your exsiting OS on top of Xen as Dom0. It is in Dom0 that the xapi deamon will run. You can find more details on the Xen.org wiki: http://wiki.xen.org/wiki/Project_Kronos

Post install steps

Now you have installed XenServer or XCP, it is time to start running OpenStack. Before you can start running OpenStack you must ensure:

- Ensure you are using the EXT type of storage repository (SR). Features that require access to VHD files (such as copy on write, snapshot and migration) do not work when using the LVM SR. Storage repositry (SR) is a XenAPI specific term relating to the physical storage on which virtual disks are stored.
- Enable passwordless SSH login between all your XenServer hosts if you want to use the resize or migration functionality.
- Create the directory "/images" if you want resize or migration to work.

You are now ready to install OpenStack onto your XenServer system. This process involves the following steps:

- Install the VIF isolation rules to help prevent mac and ip address spoofing.

- Install the XenAPI plugins.
- Create a Paravirtualised virtual machine that can run the OpenStack compute code.
- Install and configure the nova-compute in the above virtual machine.

For further information on how to perform these steps look at how DevStack performs the last three steps when doing developer deployments. For more information on DevStack, take a look at the DevStack and XenServer Readme. More information on the first step can be found in the XenServer mutli-tenancy protection doc. More information on how to install the XenAPI plugins can be found in the XenAPI plugins Readme.

Further reading

Here are some of the resources avaiable to learn more about Xen:

- Citrix XenServer official documentation: http://docs.vmd.citrix.com/XenServer.
- What is Xen? by Xen.org: http://xen.org/files/Marketing/WhatisXen.pdf.
- Xen Hypervisor project: http://xen.org/products/xenhyp.html.
- XCP project: http://xen.org/products/cloudxen.html.
- Further XenServer and OpenStack information: http://wiki.openstack.org/XenServer.

Pre-configuring the network

These instructions are for using the FlatDHCP networking mode with a single network interface. More complex configurations are described in the networking section, but this configuration is known to work. These configuration options should be set on all compute nodes.

Set your network interface in promiscuous mode so that it can receive packets that are intended for virtual machines. As root:

```
# ip link set eth0 promisc on
```

Set up your /etc/network/interfaces file with these settings:

- eth0: public IP, gateway
- br100: no ports, stp off, fd 0, first address from fixed_range set in nova.conf files.

Here's an Ubuntu/Debian example:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp

# Bridge network interface for VM networks
auto br100
iface br100 inet static
address 192.168.100.1
```

```
netmask 255.255.255.0
bridge_stp off
bridge_fd 0
```

Also install bridge-utils:

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

Ensure that you set up the bridge, although if you use flat_network_bridge=br100 in your `nova.conf` file, nova will set up the bridge for you when you run the nova-manage network command.

```
sudo brctl addbr br100
```

Lastly, restart networking to have these changes take effect. (This method is deprecated but "`restart networking`" doesn't always work.)

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

Configuring the SQL Database (MySQL) on the Cloud Controller

Start the mysql command line client by running:

```
mysql -u root -p
```

Enter the mysql root user's password when prompted.

To configure the MySQL database, create the nova database.

```
mysql> CREATE DATABASE nova;
```

Create a MySQL user and password for the newly-created nova database that has full control of the database.

```
mysql> GRANT ALL ON nova.* TO 'nova'@'%' IDENTIFIED BY
'[YOUR_NOVADB_PASSWORD]';
```

Enter quit at the mysql> prompt to exit MySQL.

```
mysql> quit
```

The command to populate the database is described later in the documentation, in the Section entitled Configuring the Database for Compute.

Configuring the SQL Database (PostgreSQL) on the Cloud Controller

Optionally, if you choose not to use MySQL, you can install and configure PostgreSQL for all your databases. Here's a walkthrough for the Nova database:

```
$ sudo apt-get install postgresql postgresql-client
```

Start the PostgreSQL command line client by running:

```
sudo su - postgres
```

Enter the postgresql root user's password if prompted.

To configure the database, create the nova database.

```
postgres> psql
postgres=# CREATE USER novadbadmin;
postgres=# ALTER USER novadbadmin WITH PASSWORD '[YOUR_NOVADB_PASSWORD]';
postgres=# CREATE DATABASE nova;
postgres=# GRANT ALL PRIVILEGES ON DATABASE nova TO novadbadmin;
postgres=# \q
postgres> exit
```

The database is created and we have a privileged user that controls the database. Now we have to install the packages that will help Nova access the database.

```
$ sudo apt-get install python-sqlalchemy python-psycopg2
```

Configure the `/etc/nova/nova.conf` file, to ensure it knows to use the PostgreSQL database:

```
sql_connection = postgres://novadbadmin:[[YOUR_NOVADB_PASSWORD]]@127.0.0.1/
nova
```

The command to populate the database is described later in the documentation, in the section entitled Configuring the Database for Compute.

Installing the Cloud Controller

Install the messaging queue server, RabbitMQ.

You have the option of installing Apache Qpid, refer to the Compute Administration Manual for configuration instructions, including adding rpc_backend=nova.rpc.impl_qpid to your `nova.conf`.

```
sudo apt-get install rabbitmq-server
```

Install the required nova- packages, and dependencies are automatically installed.

```
sudo apt-get install nova-compute nova-volume nova-novncproxy novnc nova-api
 nova-ajax-console-proxy nova-cert nova-consoleauth nova-doc nova-scheduler
 nova-network
```

If you see the error:

```
E: Unable to locate package nova-novncproxy
```

ensure that you have installed the Ubuntu Cloud Archive packages by adding the following to `/etc/apt/sources.list.d/folsom.list`:

```
deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/folsom
 main
```

Prior to running apt-get update and apt-get upgrade, install the keyring:

```
sudo apt-get install ubuntu-cloud-keyring
```

Configuring OpenStack Compute

Many of the settings for Compute services are stored in the `/etc/nova/nova.conf` file by default on every node running any of the nova-* services. This section describes the

relevant settings for getting a minimal install running. Refer to the OpenStack Compute Administration Manual for guidance on more configuration options.

In general, you can use the same `nova.conf` file across the controller and compute nodes. However, the following configuration options need to be changed on each compute host:

- `my_ip`
- `vncserver_listen`
- `vncserver_proxyclient_address`

For the above configuration options, you must use the IP address of the specific compute host, not the cloud controller.

Note

In the Essex release, the `nova.conf` file format changed from flags (`--name=value`) to INI file format (`name=value`). The Compute service will function properly with either format.

The default `nova.conf` format for Essex packages varies by distribution. Fedora packages use the new INI style format, and Ubuntu 12.04 packages use the old flag format. By convention, this guide uses the new INI style format.

You can automatically convert a `nova.conf` file from flag format to INI format using **nova-manage**:

```
$ nova-manage config convert --help
Usage: nova-manage config convert <args> [options]


Options:
  -h, --help        show this help message and exit
  --infile=<path>  old-style flagfile to convert to config
  --outfile=<path>  path for output file. Writes config to stdout if
 not
                    specified
```

The packages automatically do these steps for a user named nova, but if you are installing as another user you should ensure that the `nova.conf` file should have its owner set to `root:nova`, and mode set to `0640`, since the file contains your MySQL server's username and password.

Note

If you are installing as another user, you should set permissions correctly. This packaged install ensures that the nova user belongs to the nova group and that the .conf file permissions are set, but here are the manual commands, which should be run as root:

```
# groupadd nova
# usermod -g nova nova
# chown -R username:nova /etc/nova
# chmod 640 /etc/nova/nova.conf
```

The hypervisor is set either by editing `/etc/nova/nova.conf` or referring to `nova-compute.conf` in the `nova.conf` file. The hypervisor defaults to `kvm`, but if you are working within a VM already, switch to `qemu` on the `libvirt_type=` line. To use Xen, refer to the overview in this book for where to install nova components.

Ensure the database connection defines your backend data store by adding a `sql_connection` line to `nova.conf`: `sql_connection=mysql://[user]:[pass]@[primary IP]/[db name]`, such as `sql_connection=mysql://nova:yourpassword@192.168.206.130/nova`.

You must change `/etc/nova/api-paste.ini` under [filter:authtoken] to set admin_tenant_name, admin_user, admin_password. Most distributions have variables in place in this file.

```
############
# Metadata #
############
[composite:metadata]
use = egg:Paste#urlmap
/: meta

[pipeline:meta]
pipeline = ec2faultwrap logrequest metaapp

[app:metaapp]
paste.app_factory = nova.api.metadata.handler:MetadataRequestHandler.factory

#######
# EC2 #
#######

[composite:ec2]
use = egg:Paste#urlmap
/services/Cloud: ec2cloud

[composite:ec2cloud]
use = call:nova.api.auth:pipeline_factory
noauth = ec2faultwrap logrequest ec2noauth cloudrequest validator ec2executor
keystone = ec2faultwrap logrequest ec2keystoneauth cloudrequest validator
 ec2executor

[filter:ec2faultwrap]
paste.filter_factory = nova.api.ec2:FaultWrapper.factory

[filter:logrequest]
paste.filter_factory = nova.api.ec2:RequestLogging.factory

[filter:ec2lockout]
paste.filter_factory = nova.api.ec2:Lockout.factory

[filter:ec2keystoneauth]
paste.filter_factory = nova.api.ec2:EC2KeystoneAuth.factory

[filter:ec2noauth]
paste.filter_factory = nova.api.ec2:NoAuth.factory

[filter:cloudrequest]
controller = nova.api.ec2.cloud.CloudController
paste.filter_factory = nova.api.ec2:Requestify.factory
```

```
[filter:authorizer]
paste.filter_factory = nova.api.ec2:Authorizer.factory

[filter:validator]
paste.filter_factory = nova.api.ec2:Validator.factory

[app:ec2executor]
paste.app_factory = nova.api.ec2:Executor.factory

##############
# Openstack #
##############

[composite:osapi_compute]
use = call:nova.api.openstack.urlmap:urlmap_factory
/: oscomputeversions
/v1.1: openstack_compute_api_v2
/v2: openstack_compute_api_v2

[composite:osapi_volume]
use = call:nova.api.openstack.urlmap:urlmap_factory
/: osvolumeversions
/v1: openstack_volume_api_v1

[composite:openstack_compute_api_v2]
use = call:nova.api.auth:pipeline_factory
noauth = faultwrap sizelimit noauth ratelimit osapi_compute_app_v2
keystone = faultwrap sizelimit authtoken keystonecontext ratelimit
 osapi_compute_app_v2
keystone_nolimit = faultwrap sizelimit authtoken keystonecontext
 osapi_compute_app_v2

[composite:openstack_volume_api_v1]
use = call:nova.api.auth:pipeline_factory
noauth = faultwrap sizelimit noauth ratelimit osapi_volume_app_v1
keystone = faultwrap sizelimit authtoken keystonecontext ratelimit
 osapi_volume_app_v1
keystone_nolimit = faultwrap sizelimit authtoken keystonecontext
 osapi_volume_app_v1

[filter:faultwrap]
paste.filter_factory = nova.api.openstack:FaultWrapper.factory

[filter:noauth]
paste.filter_factory = nova.api.openstack.auth:NoAuthMiddleware.factory

[filter:ratelimit]
paste.filter_factory = nova.api.openstack.compute.
limits:RateLimitingMiddleware.factory

[filter:sizelimit]
paste.filter_factory = nova.api.sizelimit:RequestBodySizeLimiter.factory

[app:osapi_compute_app_v2]
paste.app_factory = nova.api.openstack.compute:APIRouter.factory

[pipeline:oscomputeversions]
pipeline = faultwrap oscomputeversionapp
```

```
[app:osapi_volume_app_v1]
paste.app_factory = nova.api.openstack.volume:APIRouter.factory

[app:oscomputeversionapp]
paste.app_factory = nova.api.openstack.compute.versions:Versions.factory

[pipeline:osvolumeversions]
pipeline = faultwrap osvolumeversionapp
##########
# Shared #
##########

[filter:keystonecontext]
paste.filter_factory = nova.api.auth:NovaKeystoneContext.factory

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
```

Add these settings to `/etc/nova/nova.conf` for the network configuration assumptions made for this installation scenario. You can place comments in the `nova.conf` file by entering a new line with a # sign at the beginning of the line. To see a listing of all possible configuration option settings, see the reference in the OpenStack Compute Administration Manual.

```
auth_strategy=keystone
network_manager=nova.network.manager.FlatDHCPManager
fixed_range=192.168.100.0/24
public_interface=eth0
flat_interface=eth0
flat_network_bridge=br100
```

Here is an example `nova.conf` with commented sections:

```
[DEFAULT]

# LOGS/STATE
verbose=True
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova
rootwrap_config=/etc/nova/rootwrap.conf

# AUTHENTICATION
auth_strategy=keystone
[keystone_authtoken]
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = nova
signing_dirname = /tmp/keystone-signing-nova

# SCHEDULER
compute_scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler
```

```
# VOLUMES
volume_driver=nova.volume.driver.ISCSIDriver
volume_group=nova-volumes
volume_name_template=volume-%s
iscsi_helper=tgtadm

# DATABASE
sql_connection=mysql://nova:yourpassword@192.168.206.130/nova

# COMPUTE
libvirt_type=qemu
compute_driver=libvirt.LibvirtDriver
instance_name_template=instance-%08x
api_paste_config=/etc/nova/api-paste.ini

# COMPUTE/APIS: if you have separate configs for separate services
# this flag is required for both nova-api and nova-compute
allow_resize_to_same_host=True

# APIS
osapi_compute_extension=nova.api.openstack.compute.contrib.standard_extensions
ec2_dmz_host=192.168.206.130
s3_host=192.168.206.130

# RABBITMQ
rabbit_host=192.168.206.130

# GLANCE
image_service=nova.image.glance.GlanceImageService
glance_api_servers=192.168.206.130:9292

# NETWORK
network_manager=nova.network.manager.FlatDHCPManager
force_dhcp_release=True
dhcpbridge_flagfile=/etc/nova/nova.conf
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
# Change my_ip to match each host
my_ip=192.168.206.130
public_interface=br100
vlan_interface=eth0
flat_network_bridge=br100
flat_interface=eth0
fixed_range=192.168.100.0/24

# NOVNC CONSOLE
novncproxy_base_url=http://192.168.206.130:6080/vnc_auto.html
# Change vncserver_proxyclient_address and vncserver_listen to match each
 compute host
vncserver_proxyclient_address=192.168.206.130
vncserver_listen=192.168.206.130
```

Note

The `my_ip` configuration option will be different for each host, edit it accordingly.

Stop the nova- services prior to running db sync, by running stop commands as root. Otherwise your logs show errors because the database has not yet been populated:

```
# stop nova-api
# stop nova-compute
# stop nova-network
# stop nova-scheduler
# stop nova-novncproxy
# stop nova-volume
```

Configuring the Database for Compute

Create the tables in your backend data store by running the following command:

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

If you see any response, you can look in /var/log/nova/nova-manage.log to see the problem. No response means the command completed correctly and your nova database is now populated.

Restart all services in total, just to cover the entire spectrum:

```
sudo start nova-api
sudo start nova-compute
sudo start nova-network
sudo start nova-scheduler
sudo start nova-novncproxy
sudo start nova-volume
sudo start libvirt-bin
sudo /etc/init.d/rabbitmq-server restart
```

All nova services are now installed and started. If the "start" command doesn't work, your services may not be running correctly (or not at all). Review the logs in /var/log/nova to look for clues.

Creating the Network for Compute VMs

You must run the command that creates the network and the bridge using the br100 specified in the nova.conf file to create the network that the virtual machines use. This example shows the network range using 192.168.100.0/24 as the fixed range for our guest VMs, but you can substitute the range for the network you have available. We're labeling it with "private" in this case.

```
nova-manage network create private --multi_host=T --
fixed_range_v4=192.168.100.0/24 --bridge_interface=br100 --num_networks=1 --
network_size=256
```

Verifying the Compute Installation

You can ensure all the Compute services are running by using the **nova-manage** command, as root:

```
# nova-manage service list
```

In return you should see "smiley faces" rather than three X symbols. Here's an example.

```
Binary           Host     Zone    Status     State Updated_At
nova-compute     myhost    nova    enabled    :-)   2012-04-02 14:06:15
nova-cert        myhost    nova    enabled    :-)   2012-04-02 14:06:16
nova-volume      myhost    nova    enabled    :-)   2012-04-02 14:06:14
nova-scheduler   myhost    nova    enabled    :-)   2012-04-02 14:06:11
nova-network     myhost    nova    enabled    :-)   2012-04-02 14:06:13
nova-consoleauth myhost    nova    enabled    :-)   2012-04-02 14:06:10
```

Note

If you see three X symbols and are running services on separate hosts, ensure that ntp is synchronizing time correctly and that all servers match their time. Out-of-sync time stamps are the most common cause of the XXX state.

You can find the version of the installation by using the **nova-manage** command, as root:

```
# nova-manage version list
```

The version number 2012.2 corresponds with the Folsom release of Compute.

```
2012.2 (2012.1-LOCALBRANCH:LOCALREVISION)
```

Defining Compute and Image Service Credentials

Create an `openrc` file that can contain these variables that are used by the **nova** (Compute) and **glance** (Image) command-line interface clients. In this document, we store the `openrc` file in the `~/creds` directory:

```
$ mkdir ~/creds
$ nano ~/creds/openrc
```

In the `openrc` file you create, paste these values:

```
export OS_USERNAME=admin
export OS_TENANT_NAME=openstackDemo
export OS_PASSWORD=secretword
export OS_AUTH_URL=http://192.168.206.130:5000/v2.0/
export OS_REGION_NAME=RegionOne
```

Next, ensure these are used in your environment. If you see 401 Not Authorized errors on commands using tokens, ensure that you have properly sourced your credentials and that all the pipelines are accurate in the configuration files.

```
$ source ~/creds/openrc
```

Verify your credentials are working by using the **nova** client to list the available images:

```
$ nova image-list

+--------------------------------------+-------------------+--------+--------+
|                  ID                  |        Name       | Status | Server |
+--------------------------------------+-------------------+--------+--------+
| 58b71381-1d8a-4cbb-94f2-c67bfced35f5 | tty-linux-kernel  | ACTIVE |        |
| 5ca3d2fa-fd89-4edb-af41-ae96136e48ef | tty-linux-ramdisk | ACTIVE |        |
```

```
| cc63afc1-9a83-4978-8d75-f19d874fdf94 | tty-linux         | ACTIVE |        |
+--------------------------------------+-------------------+--------+--------+
```

Note that the ID values on your installation will be different.

Installing Additional Compute Nodes

There are many different ways to perform a multinode install of Compute in order to scale out your deployment and run more compute nodes, enabling more virtual machines to run simultaneously.

Ensure that the networking on each node is configured as documented in the Pre-configuring the network section.

In this case, you can install all the nova- packages and dependencies as you did for the Cloud Controller node, or just install nova-compute. Your installation can run any nova-services anywhere, so long as the service can access `nova.conf` so it knows where the Rabbitmq or Qpid messaging server is installed.

When running in a high-availability mode for networking, the compute node is where you configure the compute network, the networking between your instances. Learn more about high-availability for networking in the Compute Adminstration manual.

Because you may need to query the database from the compute node and learn more information about instances, the nova client and MySQL client or PostgresSQL client packages should be installed on any additional compute nodes.

Copy the `nova.conf` from your controller node to all additional compute nodes. As mentioned in the section entitled Configuring OpenStack Compute, modify the following configuration options so that they match the IP address of the compute host:

- `my_ip`
- `vncserver_listen`
- `vncserver_proxyclient_address`

7. Registering Virtual Machine Images

To test your deployment, download some virtual machine images that are known to work with OpenStack. CirrOS is a small test image that is often used for testing OpenStack deployments. You can find the most recent CirrOS image on the CirrOS Launchpad home page under "Downloads". As of this writing the most recent image is version 0.3.0. A 64-bit version in QCOW2 format (compatible with KVM or QEMU hypervisors) can be downloaded at https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-disk.img. Once you launch the image, log in with the following credentials:

- Username: `cirros`
- Password: `cubswin:)`

The 64-bit CirrOS QCOW2 image is the image we'll use for this walkthrough. More detailed information about how to obtain and create images can be found in the OpenStack Compute Administration Guide in the "Image Management" chapter.

Create a directory called `stackimages` to house your image files:

```
$ mkdir stackimages
```

Download the CirrOS image into your `stackimages` directory.

```
$ wget -c https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-
x86_64-disk.img -O stackimages/cirros.img
```

Verify that your **glance** client can access the Image service by requesting a list of installed images:

```
$ glance index
ID                                   Name                           Disk
 Format          Container Format     Size
------------------------------------ ------------------------------
 -------------------- -------------------- --------------
```

If you get the following error, make sure that the environment variables set in `~/openrc`

```
Failed to show index. Got error:
You are not authenticated.
Details: 401 Unauthorized

This server could not verify that you are authorized to access the document
 you requested. Either you supplied the wrong credentials (e.g., bad
 password), or your browser does not understand how to supply the credentials
 required.

 Authentication required
```

Add the CirrOS image to the Image service using the **glance add** command, passing the image file through standard input:

```
$ glance add name=cirros-0.3.0-x86_64 --disk-format=qcow2 --container-format=
bare < stackimages/cirros.img
Added new image with ID: f4addd24-4e8a-46bb-b15d-fae2591f1a35
```

Note

The returned image ID is generated dynamically, and therefore will be different on your deployment than in this example.

The rationale for the arguments is:

`name=cirros-0.3.0-x86_64` The `name` field is an arbitrary label. In this example the name encodes the distribution, version, and architecture: `cirros-0.3.0-x864_64`.

`disk-format=qcow2` The `disk-format` field specifies the format of the image file. In this case, the image file format is QCOW2, which can be verified using the **file** command:

```
$ file stackimages/cirros.img
stackimages/cirros.img: QEMU QCOW Image (v2),
 41126400 bytes
```

Other valid formats are `raw`, `vhd`, `vmdk`, `vdi`, `iso`, `aki`, `ari` and `ami`.

`container-format=bare` The `container-format` field is required by the **glance add** command but isn't actually used by any of the OpenStack services, so the value specified here has no effect on system behavior. We specify `bare` to indicate that the image file is not in a file format that contains metadata about the virtual machine.

Because the value is not used anywhere, it safe to always specify `bare` as the container format, although the command will accept other formats: `ovf`, `aki`, `ari`, `ami`.

Confirm it was uploaded by listing the images in the Image service:

```
$ glance index
ID                                   Name                           Disk
 Format          Container Format     Size
------------------------------------ ------------------------------
 -------------------- -------------------- --------------
f4addd24-4e8a-46bb-b15d-fae2591f1a35 cirros-0.3.0-x86_64            qcow2
           bare                       9761280
```

The **nova image-list** command will also list the images in the Image service:

```
$ nova image-list
+--------------------------------------
+----------------------------------------+--------
+-------------------------------------+
|                  ID                  |                  Name
  | Status |                Server                |
+--------------------------------------
+----------------------------------------+--------
+-------------------------------------+
| f4addd24-4e8a-46bb-b15d-fae2591f1a35 | cirros-0.3.0-x86_64
  | ACTIVE |                                       |
```

```
+--------------------------------------
+----------------------------------------+--------
+--------------------------------------+
```

8. Running Virtual Machine Instances

Security groups: Enabling SSH and ICMP (ping)

The Compute service uses the concept of security groups to control what network protocols (TCP, UDP, ICMP), ports, and IP addresses are permitted to access instances. Each tenant manages its own list of security groups and starts off with a security group called `default`. If no security group is specified upon boot, the virtual machine will be associated with the `default` security group.

Security groups can be listed by the **nova secgroup-list** command.

```
$ nova secgroup-list
+---------+-------------+
|   Name  | Description |
+---------+-------------+
| default | default     |
+---------+-------------+
```

In this example, we will use the nova **secgroup-add-rule** command to enable access to TCP port 22 (so we can SSH to instances) Allow access to port 22 from all IP addresses (specified in CIDR notation as `0.0.0.0/0`) with the following command:

```
$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
```

When specifying rules for TCP and UDP protocols, you may specify a range of port consecutive addresses in a single rule (e.g., from port 5901 to port 5999). In this case, only a single port is being enabled, so we specify the start port as 22 and the end port as 22.

To be able to ping virtual machine instances, you must specify a rule to allow ICMP traffic. When specifying ICMP rules, instead of specifying a begin and end port, you specify a pemitted ICMP code and ICMP type. You can also specify `-1` for the code to enable all codes and `-1` for the type to enable all ICMP types. Allow access to all codes and types of ICMP traffic from all IP addresses with the following command:

```
$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
```

Adding a keypair

The Compute service can inject an SSH public key into an account on the instance, assuming the virtual machine image being used supports this. To add a keypair to the Compute service, use the **nova keypair-add** command. This command can be used to either generate a new keypair, or to upload an existing public key. The following example uploads an existing public key, located at `~/.ssh/id_rsa.pub`, and gives the keypair the name `mykey`.

```
$ nova keypair-add --pub_key ~/.ssh/id_rsa.pub mykey
```

List the keypairs by doing:

```
$ nova keypair-list
+-------+-------------------------------------------------+
```

```
|  Name |                   Fingerprint                    |
| mykey | c3:d2:b5:d3:ec:4a:29:b0:22:32:6e:34:dd:91:f9:cf |
+-------+--------------------------------------------------+
```

Confirm that the uploaded keypair matches your local key by checking your key's fingerprint with the **ssh-keygen** command:

```
$ ssh-keygen -l -f ~/.ssh/id_rsa.pub
2048 c3:d2:b5:d3:ec:4a:29:b0:22:32:6e:34:dd:91:f9:cf /home/myaccount/.ssh/
id_rsa.pub (RSA)
```

Confirm all services running

Before trying to start an instance, confirm that all of the necessary services are running, in particular:

`nova-api` The `nova-api` service must be running to respond to the request to boot an instance, as well as to serve as the metadata server so that the instance can retrieve the public key uploaded i a previous section. If the **nova** commands in the previous section succeeded, then the service is running.

`nova-scheduler` The `nova-scheduler` service must be running in order to dispatch requests for a new virtual machine instance to a host running the `nova-compute` service that has sufficient resources.

`nova-compute` The `nova-compute` service must be running in order to interact with the hypervisor to bring up a virtual machine instance.

`nova-network` The `nova-network` service must be running in order to perform networking tasks such as assigning an IP address to the virtual machine instance and implementing the security group rules.

The **nova-manage service list** command can be used to confirm that these services are running properly.

Note

The **nova-manage service list** command does not indicate whether the `nova-api` service is running.

As root:

```
# nova-manage service list
Binary           Host                                 Zone             Status
  State Updated_At
nova-compute     myhost-1                              nova             enabled
   :-)   2012-05-27 12:36:35
nova-network     myhost-1                              nova             enabled
   :-)   2012-05-27 12:36:28
nova-scheduler   myhost-1                              nova             enabled
   :-)   2012-05-27 12:36:33
```

If any of the services are missing in your configuration, or the `State` column does not show a smiley face, then your Compute service will not be able to launch an instance.

Starting an instance

To start an instance, we need to specify a *flavor*, also known as an *instance type*, which indicates the size of an instance. Use the **nova flavor-list** command to view the list of available flavors:

```
$ nova flavor-list
+----+-----------+-----------+------+-----------+------+-------+-------------+
| ID |    Name   | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor |
+----+-----------+-----------+------+-----------+------+-------+-------------+
| 1  | m1.tiny   | 512       | 0    | 0         |      | 1     | 1.0         |
| 2  | m1.small  | 2048      | 15   | 15        |      | 1     | 1.0         |
| 3  | m1.medium | 4096      | 25   | 25        |      | 2     | 1.0         |
| 4  | m1.large  | 8192      | 45   | 45        |      | 4     | 1.0         |
| 5  | m1.xlarge | 16384     | 85   | 85        |      | 8     | 1.0         |
+----+-----------+-----------+------+-----------+------+-------+-------------+
```

We also need to specify the image. Use the **nova image-list** to retrieve the ID of the CirrOS image.

```
$ nova image-list
+--------------------------------------
+-----------------------------------------+--------
+--------------------------------------+
|                  ID                  |                  Name
  | Status |                Server                |
+--------------------------------------
+-----------------------------------------+--------
+--------------------------------------+
| f4addd24-4e8a-46bb-b15d-fae2591f1a35 | cirros-0.3.0-x86_64
  | ACTIVE |                                      |
+--------------------------------------
+-----------------------------------------+--------
+--------------------------------------+
```

Use the `nova boot` command to launch a new virtual machine instance. We'll use an `m1.small` instance in this example, using the CirrOS image, and the `mykey` keypair we added. We also need to give this virtual machine instance a name, we'll call it `cirros`. We will explicitly specify the `default` security group in this example, although this isn't strictly necessary since the `default` group will be used if no security group is specified.

```
$ nova boot --flavor 2 --image f4addd24-4e8a-46bb-b15d-fae2591f1a35 --key_name
 mykey --security_group default cirros
+------------------------------------
+------------------------------------------------------------+
|               Property             |                          Value
                 |
+------------------------------------
+------------------------------------------------------------+
| OS-DCF:diskConfig                  | MANUAL
                 |
| OS-EXT-SRV-ATTR:host               | host-1
                 |
| OS-EXT-SRV-ATTR:hypervisor_hostname | None
                 |
| OS-EXT-SRV-ATTR:instance_name      | instance-00000001
                 |
| OS-EXT-STS:power_state             | 0
                 |
```

```
| OS-EXT-STS:task_state                | scheduling
                  |
| OS-EXT-STS:vm_state                  | building
                  |
| accessIPv4                           |
                  |
| accessIPv6                           |
                  |
| adminPass                            | RG3W2bpZDbCo
                  |
| config_drive                         |
                  |
| created                              | 2012-05-27T13:00:33Z
                  |
| flavor                               | m1.small
                  |
| hostId                               |
 a2fd457e034c030506bac5c790c38d9519ea7a03b6861474a712c6b7 |
| id                                   | c6bbbf26-b40a-47e7-8d5c-eb17bf65c485
                  |
| image                                | cirros-0.3.0-x86_64
                  |
| key_name                             | mykey
                  |
| metadata                             | {}
                  |
| name                                 | cirros
                  |
| progress                             | 0
                  |
| status                               | BUILD
                  |
| tenant_id                            | b5815b046cfe47bb891a7b64119e7f80
                  |
| updated                              | 2012-05-27T13:00:33Z
                  |
| user_id                              | a4c2d43f80a549a19864c89d759bb3fe
                  |
+--------------------------------------
+------------------------------------------------------------+
```

Check the progress of the instance with the **nova list** command. When the instance has booted, the command output will look something like:

```
$ nova list
+--------------------------------------+----------------------+--------
+-----------------------------------------+
|                  ID                  |          Name         | Status |
            Networks                |
+--------------------------------------+----------------------+--------
+-----------------------------------------+
| c6bbbf26-b40a-47e7-8d5c-eb17bf65c485 | cirros                | ACTIVE |
 private=192.168.100.5                    |
+--------------------------------------+----------------------+--------
+-----------------------------------------+
```

You can view the boot messages of the instances using the **nova console-log** command:

```
$ nova console-log
...
```

```
Starting network...
udhcpc (v1.18.5) started
Sending discover...
Sending select for 192.168.100.5...
Lease of 192.168.100.5 obtained, lease time 120
deleting routers
route: SIOCDELRT: No such process
adding dns 192.168.100.4
cloud-setup: checking http://169.254.169.254/2009-04-04/meta-data/instance-id
cloud-setup: successful after 1/30 tries: up 1.45. iid=i-00000001
Starting dropbear sshd: generating rsa key... generating dsa key... OK
===== cloud-final: system completely up in 1.77 seconds ====
  instance-id: i-00000001
  public-ipv4:
  local-ipv4 : 192.168.100.5
cloud-userdata: user data not a script
  ____               ____  ____
 / __/ __ ____ ____ / __ \/ __/
/ /__ / // __// __// /_/ /\ \
\___//_//_/  /_/   \____/___/
 http://launchpad.net/cirros


login as 'cirros' user. default password: 'cubswin:)'. use 'sudo' for root.
cirros login:
```

You should be able to ping your instance:

```
$ ping -c5 192.168.100.5
PING 192.168.100.5 (192.168.100.5) 56(84) bytes of data.
64 bytes from 192.168.100.5: icmp_req=1 ttl=64 time=0.270 ms
64 bytes from 192.168.100.5: icmp_req=2 ttl=64 time=0.228 ms
64 bytes from 192.168.100.5: icmp_req=3 ttl=64 time=0.244 ms
64 bytes from 192.168.100.5: icmp_req=4 ttl=64 time=0.203 ms
64 bytes from 192.168.100.5: icmp_req=5 ttl=64 time=0.210 ms

--- 192.168.100.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.203/0.231/0.270/0.024 ms
```

You should be able to ssh to your instance as the `cirros` user, using either the ssh keypair you uploaded or using the password `cubswin:)`

```
$ ssh cirros@192.168.100.5
The authenticity of host '192.168.100.5 (192.168.100.5)' can't be established.
RSA key fingerprint is c2:0a:95:d4:e7:e1:a6:a2:6a:99:4d:b8:f9:66:13:64.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.100.5' (RSA) to the list of known hosts.
cirros@192.168.100.5's password: cubswin:)
$
```

Bringing down an instance

Bring down your instance using the **nova delete** command:

```
$ nova delete c6bbbf26-b40a-47e7-8d5c-eb17bf65c485
```

9. Installing OpenStack Object Storage

The OpenStack Object Storage services work together to provide object storage and retrieval through a REST API.

System Requirements

Hardware: OpenStack Object Storage specifically is designed to run on commodity hardware.

Table 9.1. Hardware Recommendations

Server	Recommended Hardware	Notes
Object Storage object servers	Processor: dual quad core Memory: 8 or 12 GB RAM Disk space: optimized for cost per GB Network: one 1 GB Network Interface Card (NIC)	The amount of disk space depends on how much you can fit into the rack efficiently. You want to optimize these for best cost per GB while still getting industry-standard failure rates. At Rackspace, our storage servers are currently running fairly generic 4U servers with 24 2T SATA drives and 8 cores of processing power. RAID on the storage drives is not required and not recommended. Swift's disk usage pattern is the worst case possible for RAID, and performance degrades very quickly using RAID 5 or 6. As an example, Rackspace runs Cloud Files storage servers with 24 2T SATA drives and 8 cores of processing power. Most services support either a worker or concurrency value in the settings. This allows the services to make effective use of the cores available.
Object Storage container/account servers	Processor: dual quad core Memory: 8 or 12 GB RAM Network: one 1 GB Network Interface Card (NIC)	Optimized for IOPS due to tracking with SQLite databases.
Object Storage proxy server	Processor: dual quad core Network: one 1 GB Network Interface Card (NIC)	Higher network throughput offers better performance for supporting many API requests. Optimize your proxy servers for best CPU performance. The Proxy Services are more CPU and network I/O intensive. If you are using 10g networking to the proxy, or are terminating SSL traffic at the proxy, greater CPU power will be required.

Operating System: OpenStack Object Storage currently runs on Ubuntu, RHEL, CentOS, or Fedora and the large scale deployment at Rackspace runs on Ubuntu 10.04 LTS.

Networking: 1000 Mbps are suggested. For OpenStack Object Storage, an external network should connect the outside world to the proxy servers, and the storage network is intended to be isolated on a private network or multiple private networks.

Database: For OpenStack Object Storage, a SQLite database is part of the OpenStack Object Storage container and account management process.

Permissions: You can install OpenStack Object Storage either as root or as a user with sudo permissions if you configure the sudoers file to enable all the permissions.

Object Storage Network Planning

For both conserving network resources and ensuring that network administrators understand the needs for networks and public IP addresses for providing access to the APIs and storage network as necessary, this section offers recommendations and required minimum sizes. Throughput of at least 1000 Mbps is suggested.

This document refers to two networks. One is a Public Network for connecting to the Proxy server, and the second is a Storage Network that is not accessible from outside the cluster, to which all of the nodes are connected. All of the OpenStack Object Storage services, as well as the rsync daemon on the Storage nodes are configured to listen on their STORAGE_LOCAL_NET IP addresses.

Public Network (Publicly routable IP range): This network is utilized for providing Public IP accessibility to the API endpoints within the cloud infrastructure.

Minimum size: 8 IPs (CIDR /29)

Storage Network (RFC1918 IP Range, not publicly routable): This network is utilized for all inter-server communications within the Object Storage infrastructure.

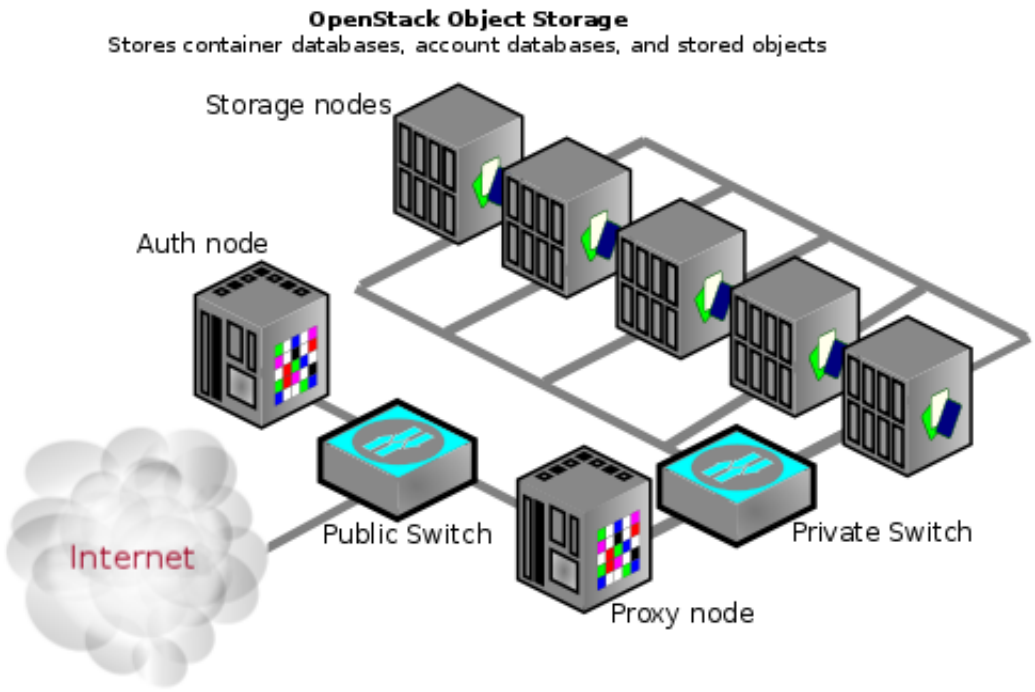
Recommended size: 255 IPs (CIDR /24)

Example Object Storage Installation Architecture

- node - a host machine running one or more OpenStack Object Storage services
- Proxy node - node that runs Proxy services
- Auth node - an optionally separate node that runs the Auth service separately from the Proxy services
- Storage node - node that runs Account, Container, and Object services
- Ring - a set of mappings of OpenStack Object Storage data to physical devices

To increase reliability, you may want to add additional Proxy servers for performance.

This document describes each Storage node as a separate zone in the ring. It is recommended to have a minimum of 5 zones. A zone is a group of nodes that is as isolated as possible from other nodes (separate servers, network, power, even geography). The ring guarantees that every replica is stored in a separate zone. This diagram shows one possible configuration for a minimal installation.



Installing OpenStack Object Storage on Ubuntu

Though you can install OpenStack Object Storage for development or testing purposes on a single server, a multiple-server installation enables the high availability and redundancy you want in a production distributed object storage system.

If you would like to perform a single node installation on Ubuntu for development purposes from source code, use the Swift All In One instructions or DevStack. See http://swift.openstack.org/development_saio.html for manual instructions or http://devstack.org for all-in-one including authentication and a dashboard.

Before You Begin

Have a copy of the Ubuntu Server installation media on hand if you are installing on a new server.

This document demonstrates installing a cluster using the following types of nodes:

- One Proxy node which runs the swift-proxy-server processes and may also run the optional swauth or tempauth services, this walkthrough uses the Identity service code-named Keystone. The proxy server serves proxy requests to the appropriate Storage nodes.

- Five Storage nodes that run the swift-account-server, swift-container-server, and swift-object-server processes which control storage of the account databases, the container databases, as well as the actual stored objects.

Note

Fewer Storage nodes can be used initially, but a minimum of 5 is recommended for a production cluster.

General Installation Steps

1. Install the baseline operating system, such as Ubuntu Server (10.04 through 12.04) or RHEL, CentOS, or Fedora, on all nodes.

2. Install the swift service and openSSH.

```
# apt-get install swift openssh-server  rsync memcached python-netifaces
 python-xattr python-memcache
```

3. Create and populate configuration directories on all nodes:

```
#mkdir -p /etc/swift
#chown -R swift:swift /etc/swift/
```

4. Create /etc/swift/swift.conf:

```
[swift-hash]
# random unique string that can never change (DO NOT LOSE)
swift_hash_path_suffix = fLIbertYgibbitZ
```

Note

The suffix value in `/etc/swift/swift.conf` should be set to some random string of text to be used as a salt when hashing to determine mappings in the ring. This file should be the same on every node in the cluster!

Next, set up your storage nodes, proxy node, and an auth node, in this walkthrough we'll use the OpenStack Identity Service, Keystone, for the common auth piece.

Installing and Configuring the Storage Nodes

Note

OpenStack Object Storage should work on any modern filesystem that supports Extended Attributes (XATTRS). We currently recommend XFS as it demonstrated the best overall performance for the swift use case after considerable testing and benchmarking at Rackspace. It is also the only filesystem that has been thoroughly tested.

1. Install Storage node packages:

```
# apt-get install swift-account swift-container swift-object xfsprogs
```

2. For every device on the node, setup the XFS volume (/dev/sdb is used as an example):

```
#fdisk /dev/sdb
```

(set up a single partition)

```
mkfs.xfs -i size=1024 /dev/sdb1
echo "/dev/sdb1 /srv/node/sdb1 xfs noatime,nodiratime,nobarrier,logbufs=8 0
 0" >> /etc/fstab
mkdir -p /srv/node/sdb1
mount /srv/node/sdb1
chown -R swift:swift /srv/node
```

3. Create /etc/rsyncd.conf:

```
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = <STORAGE_LOCAL_NET_IP>

[account]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/account.lock

[container]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/container.lock

[object]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/object.lock
```

4. Edit the following line in /etc/default/rsync:

```
RSYNC_ENABLE = true
```

5. Start rsync daemon:

```
#service rsync start
```

Note

The rsync daemon requires no authentication, so it should be run on a local, private network.

6. Create /etc/swift/account-server.conf:

```
[DEFAULT]
bind_ip = <STORAGE_LOCAL_NET_IP>
workers = 2

[pipeline:main]
pipeline = account-server
```

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

[account-replicator]

[account-auditor]

[account-reaper]
```

7. Create /etc/swift/container-server.conf:

```
[DEFAULT]
bind_ip = <STORAGE_LOCAL_NET_IP>
workers = 2

[pipeline:main]
pipeline = container-server

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

[container-replicator]

[container-updater]

[container-auditor]
```

8. Create /etc/swift/object-server.conf:

```
[DEFAULT]
bind_ip = <STORAGE_LOCAL_NET_IP>
workers = 2

[pipeline:main]
pipeline = object-server

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

[object-replicator]

[object-updater]

[object-auditor]

[object-expirer]
```

9. Start the storage services:

```
swift-init object-server start
swift-init object-replicator start
swift-init object-updater start
swift-init object-auditor start
swift-init container-server start
swift-init container-replicator start
swift-init container-updater start
swift-init container-auditor start
swift-init account-server start
swift-init account-replicator start
swift-init account-auditor start
```

Don't Panic

If you are following these instructions in a linear manner, at this point the ring files may not be present on the storage nodes. This may cause some of the services such as the *-replicator to fail to start. After you have created the ring files and distributed them to the storage nodes, a service restart should allow these to start.

Installing and Configuring the Proxy Node

The proxy server takes each request and looks up locations for the account, container, or object and routes the requests correctly. The proxy server also handles API requests. You enable account management by configuring it in the `proxy-server.conf` file.

Note

It is assumed that all commands are run as the root user.

1. Install swift-proxy service:

```
# apt-get install swift-proxy memcached
```

2. Create self-signed cert for SSL:

```
#cd /etc/swift
#openssl req -new -x509 -nodes -out cert.crt -keyout cert.key
```

3. Modify memcached to listen on the default interfaces. Preferably this should be on a local, non-public network. Edit the following line in /etc/memcached.conf, changing:

```
-l 127.0.0.1
to
-l <PROXY_LOCAL_NET_IP>
```

4. Restart the memcached server:

```
#service memcached restart
```

5. Create /etc/swift/proxy-server.conf:

```
[DEFAULT]
bind_port = 8888
user = <user>

[pipeline:main]
pipeline = healthcheck cache swift3 authtoken keystone proxy-server

[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true

[filter:keystone]
```

```
paste.filter_factory = keystone.middleware.swift_auth:filter_factory
operator_roles = Member,admin, swiftoperator

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
# Delaying the auth decision is required to support token-less
# usage for anonymous referrers ('.r:*').
delay_auth_decision = 10
service_port = 5000
service_host = 127.0.0.1
auth_port = 35357
auth_host = 127.0.0.1
auth_protocol = http
auth_uri = http://127.0.0.1:5000/
auth_token = 012345SECRET99TOKEN012345
admin_token = 012345SECRET99TOKEN012345
admin_tenant_name = service
admin_user = swift
admin_password = swift

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

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

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

Note

If you run multiple memcache servers, put the multiple IP:port listings in the [filter:cache] section of the proxy-server.conf file like:

```
10.1.2.3:11211,10.1.2.4:11211
```

Only the proxy server uses memcache.

6. Create the account, container and object rings. The builder command is basically creating a builder file with a few parameters. The parameter with the value of 18 represents 2 ^ 18th, the value that the partition will be sized to. Set this “partition power” value based on the total amount of storage you expect your entire ring to use. The value of 3 represents the number of replicas of each object, with the last value being the number of hours to restrict moving a partition more than once.

```
# cd /etc/swift
# swift-ring-builder account.builder create 18 3 1
# swift-ring-builder container.builder create 18 3 1
# swift-ring-builder object.builder create 18 3 1
```

7. For every storage device on each node add entries to each ring:

```
# swift-ring-builder account.builder add z<ZONE>-
<STORAGE_LOCAL_NET_IP>:6002/<DEVICE> 100
# swift-ring-builder container.builder add z<ZONE>-
<STORAGE_LOCAL_NET_IP_1>:6001/<DEVICE> 100
```

```
# swift-ring-builder object.builder add z<ZONE>-
<STORAGE_LOCAL_NET_IP_1>:6000/<DEVICE> 100
```

For example, if you were setting up a storage node with a partition in Zone 1 on IP 10.0.0.1. The mount point of this partition is /srv/node/sdb1, and the path in rsyncd.conf is /srv/node/, the DEVICE would be sdb1 and the commands would look like:

```
# swift-ring-builder account.builder add z1-10.0.0.1:6002/sdb1 100
# swift-ring-builder container.builder add z1-10.0.0.1:6001/sdb1 100
# swift-ring-builder object.builder add z1-10.0.0.1:6000/sdb1 100
```

Note

Assuming there are 5 zones with 1 node per zone, ZONE should start at 1 and increment by one for each additional node.

8. Verify the ring contents for each ring:

```
# swift-ring-builder account.builder
# swift-ring-builder container.builder
# swift-ring-builder object.builder
```

9. Rebalance the rings:

```
# swift-ring-builder account.builder rebalance
# swift-ring-builder container.builder rebalance
# swift-ring-builder object.builder rebalance
```

Note

Rebalancing rings can take some time.

10. Copy the account.ring.gz, container.ring.gz, and object.ring.gz files to each of the Proxy and Storage nodes in /etc/swift.

11. Make sure all the config files are owned by the swift user:

```
# chown -R swift:swift /etc/swift
```

12. Start Proxy services:

```
# swift-init proxy start
```

OpenStack Object Storage Post Installation

Now that the proxy server and storage servers are configured, you want to restart the services.

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

Verify the Installation

You can run these commands from the proxy server or any server with access to the Identity Service.

1. First, export the swift admin password (setup previously) in a variable so it can be re-used.

```
$ export ADMINPASS=secretword
```

Note

If you do not wish to have the swift admin password stored in your shell's history, you may perform the following:

```
$ export SWIFT_PROXY_CONF="/etc/swift/proxy-server.conf export
 ADMINPASS=$( grep super_admin_key ${SWIFT_PROXY_CONF} | awk
 '{ print $NF }' )
```

2. Run the swift CLI, swift, with the correct Identity service URL. Export the information for ADMINPASS using `$ export ADMINPASS=secretword`.

```
$ swift -V 2.0 -A http://<AUTH_HOSTNAME>:5000/v2.0 -U demo:admin -K
 $ADMINPASS stat
```

3. Get an X-Storage-Url and X-Auth-Token:

```
$ curl -k -v -H 'X-Storage-User: demo:admin' -H 'X-Storage-Pass: $ADMINPASS'
 http://<AUTH_HOSTNAME>:5000/auth/v2.0
```

4. Check that you can HEAD the account:

```
$ curl -k -v -H 'X-Auth-Token: <token-from-x-auth-token-above>' <url-from-x-
storage-url-above>
```

5. Use swift to upload a few files named 'bigfile[1-2].tgz' to a container named 'myfiles':

```
$ swift -A http://<AUTH_HOSTNAME>:5000/v2.0 -U demo:admin -K $ADMINPASS
 upload myfiles bigfile1.tgz
$ swift -A http://<AUTH_HOSTNAME>:5000/v2.0 -U demo:admin -K $ADMINPASS
 upload myfiles bigfile2.tgz
```

6. Use swift to download all files from the 'myfiles' container:

```
$ swift -A http://<AUTH_HOSTNAME>:5000/v2.0 -U demo:admin -K $ADMINPASS
 download myfiles
```

Note

If you are using swauth in preference to the OpenStack Identity service, you should use the `default_swift_cluster` variable to connect to your swift cluster. Please follow the swauth documentation to verify your installation.

Adding an Additional Proxy Server

For reliability's sake you may want to have more than one proxy server. You can set up the additional proxy node in the same manner that you set up the first proxy node but with additional configuration steps.

Once you have more than two proxies, you also want to load balance between the two, which means your storage endpoint (what clients use to connect to your storage) also changes. You can select from different strategies for load balancing. For example, you could use round robin dns, or a software or hardware load balancer (like pound) in front of the two proxies, and point your storage url to the load balancer.

Configure an initial proxy node for the initial setup, and then follow these additional steps for more proxy servers.

1. Update the list of memcache servers in /etc/swift/proxy-server.conf for all the added proxy servers. If you run multiple memcache servers, use this pattern for the multiple IP:port listings:

   ```
   10.1.2.3:11211,10.1.2.4:11211
   ```

   in each proxy server's conf file.:

   ```
   [filter:cache]
   use = egg:swift#memcache
   memcache_servers = <PROXY_LOCAL_NET_IP>:11211
   ```

2. Next, copy all the ring information to all the nodes, including your new proxy nodes, and ensure the ring info gets to all the storage nodes as well.

3. After you sync all the nodes, make sure the admin has the keys in /etc/swift and the ownership for the ring file is correct.

Note

If you are using swauth in preference to the OpenStack Identity service, there are additional steps to follow for the addition of a second proxy server. Please follow the swauth documentation Installation section, paying close attention to the `default_swift_cluster` variable.

10. Installing the OpenStack Dashboard

OpenStack has components that provide a view of the OpenStack installation such as a Django-built website that serves as a dashboard.

About the Dashboard

You can use a dashboard interface with an OpenStack Compute installation with a web-based console provided by the Openstack-Dashboard project. It provides web-based interactions with the OpenStack Compute cloud controller through the OpenStack APIs. For more information about the Openstack-Dashboard project, please visit: https://github.com/openstack/horizon/. These instructions are for an example deployment configured with an Apache web server.

System Requirements for the Dashboard

Because Apache does not serve content from a root user, you must use another user with sudo privileges and run as that user.

You should have a running OpenStack Compute installation with the Identity Service, Keystone, enabled for identity management.

The dashboard needs to be installed on the node that can contact the Identity Service.

You should know the URL of your Identity endpoint and the Compute endpoint.

You must know the credentials of a valid Identity service user.

You must have git installed. It's straightforward to install it with sudo apt-get install git-core.

Python 2.6 is required, and these instructions have been tested with Ubuntu 10.10. It should run on any system with Python 2.6 or 2.7 that is capable of running Django including Mac OS X (installing prerequisites may differ depending on platform).

Optional components:

- An Image Store (*Glance*) endpoint.
- An Object Store (*Swift*) endpoint.
- A Quantum (networking) endpoint.

Installing the OpenStack Dashboard

Here are the overall steps for creating the OpenStack dashboard.

1. Install the OpenStack Dashboard framework including Apache and related modules.
2. Configure the Dashboard.
3. Restart and run the Apache server.

Install the OpenStack Dashboard, as root:

```
# apt-get install -y memcached libapache2-mod-wsgi openstack-dashboard
```

Next, modify the variable CACHE_BACKEND in `/etc/openstack-dashboard/local_settings.py` to match the ones set in `/etc/memcached.conf`. Open `/etc/openstack-dashboard/local_settings.py` and look for this line:

```
CACHE_BACKEND = 'memcached://127.0.0.1:11211/'
```

Note

The address and port in the new value need to be equal to the ones set in `/etc/memcached.conf`.

If you change the memcached settings, restart the Apache web server for the changes to take effect.

Note

This guide has selected memcache as a session store for OpenStack Dashboard. There are other options available, each with benefits and drawbacks. Refer to the OpenStack Dashboard Session Storage section for more information.

Note

In order to change the timezone you can use either dashboard or inside /etc/openstack-dashboard/local_settings.py you can change below mentioned parameter.

```
TIME_ZONE = "UTC"
```

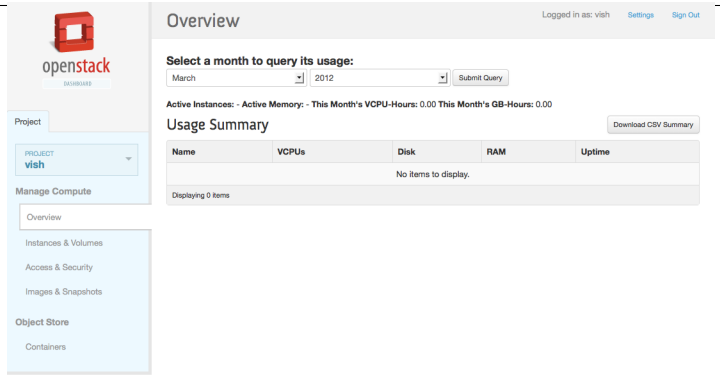
Configuring the Dashboard

A full example `local_settings.py` file is included in the Appendix of the OpenStack Install and Deploy manual. Here are some common options:

- SWIFT_ENABLED: If an Object Storage (Swift) endpoint is available and configured in the Identity service catalog, set `SWIFT_ENABLED = True`.
- QUANTUM_ENABLED: If a Network Connection (Quantum) service is available and configured in the Identity service catalog, set `QUANTUM_ENABLED = True`. Else keep it FALSE if you are not using Quantum.

Validating the Dashboard Install

To validate the Dashboard installation, point your browser at http://192.168.206.130. Note that you cannot use VNC Console from a Chrome browser. You need both Flash installed and a Firefox browser. Once you connect to the Dashboard with the URL, you should see a login window. Enter the credentials for users you created with the Identity Service, Keystone. For example, enter "admin" for the username and "secretword" as the password.

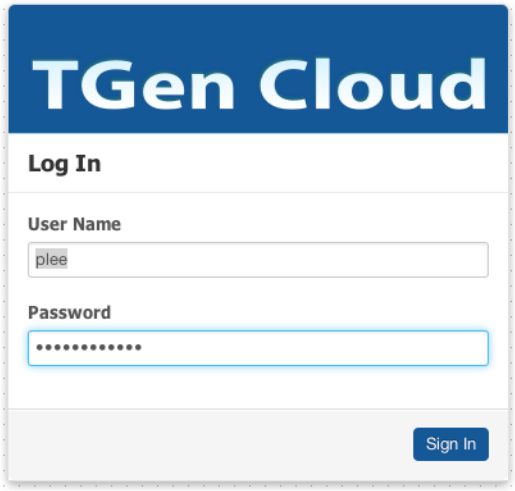


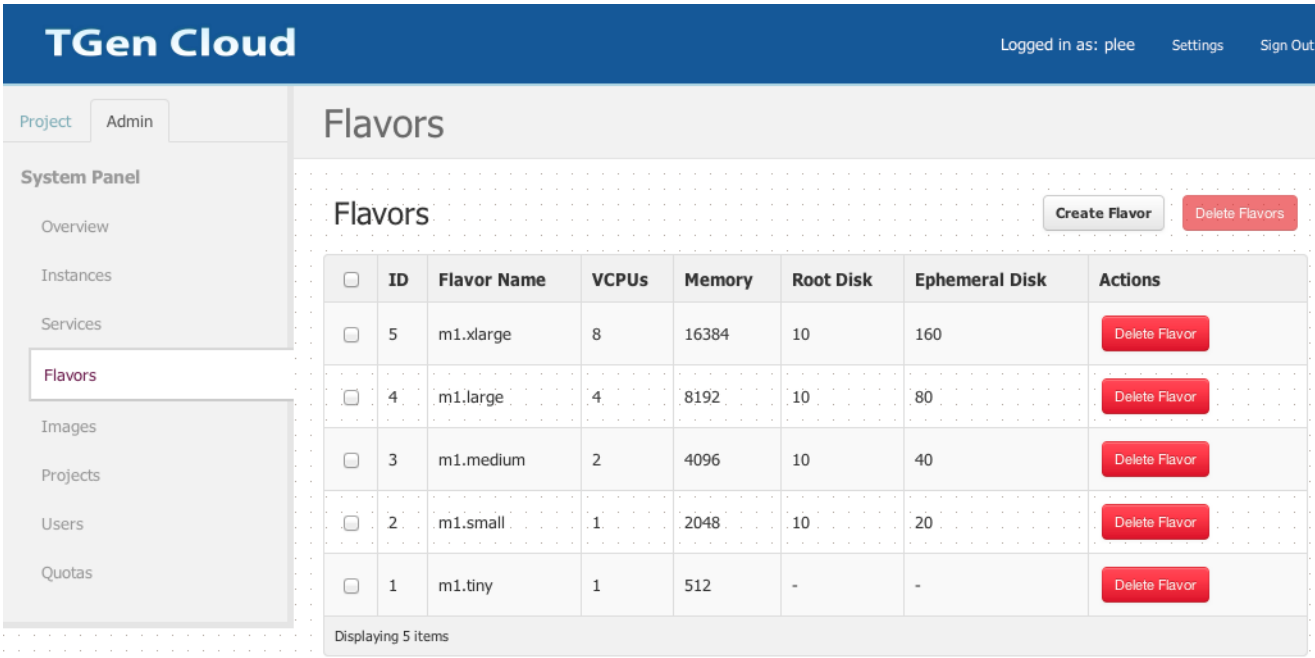
How To Custom Brand The OpenStack Dashboard (Horizon)

Adapted from a blog post by Preston Lee.

When deploying OpenStack "Essex" on Ubuntu Server 12.04, you can have the `openstack-dashboard` package installed to provide the web-based "Horizon" GUI component newly added for the Essex release. Canonical also provides an `openstack-dashboard-ubuntu-theme` package that brands the Python-based Django GUI.

The Horizon documents briefly mention branding customization to give you a head start, but here are more specific steps. Here's a custom-branded Horizon dashboard with custom colors, logo, and site title:





Once you know where to make the appropriate changes, it's super simple. Step-by-step:

1. Create a graphical logo with a transparent background. The text "TGen Cloud" in this example is actually rendered via .png files of multiple sizes created with a graphics

program. Use a 200×27 for the logged-in banner graphic, and 365×50 for the login screen graphic.

2. Set the HTML title (shown at the top of the browser window) by adding the following line to `/etc/openstack-dashboard/local_settings.py`: `SITE_BRANDING = "Example, Inc. Cloud"`

3. Upload your new graphic files to:

```
/usr/share/openstack-dashboard/openstack_dashboard/static/dashboard/img/
```

4. Create a new CSS stylesheet — we'll call ours custom.css — in the directory:

```
/usr/share/openstack-dashboard/openstack_dashboard/static/dashboard/css/
```

5. Edit your CSS file using the following as a starting point for customization, which simply overrides the Ubuntu customizations made in the ubuntu.css file.

   Change the colors and image file names as appropriate, though the relative directory paths should be the same.

```
/*
* New theme colors for dashboard that override the defaults:
*  dark blue: #355796 / rgb(53, 87, 150)
*  light blue: #BAD3E1 / rgb(186, 211, 225)
*
* By Preston Lee <plee@tgen.org>
*/
h1.brand {
background: #355796 repeat-x top left;
border-bottom: 2px solid #BAD3E1;
}
h1.brand a {
background: url(../img/my_cloud_logo_small.png) top left no-repeat;
}
#splash .login {
background: #355796 url(../img/my_cloud_logo_medium.png) no-repeat center
 35px;
}
#splash .login .modal-header {
border-top: 1px solid #BAD3E1;
}
.btn-primary {
background-image: none !important;
background-color: #355796 !important;
border: none !important;
box-shadow: none;
}
.btn-primary:hover,
.btn-primary:active {
border: none;
box-shadow: none;
background-color: #BAD3E1 !important;
text-decoration: none;
}
```

6. Open the following HTML template in an editor:

```
/usr/share/openstack-dashboard/openstack_dashboard/templates/_stylesheets.
html
```

7. Add a line to include your new stylesheet pointing to custom.css: (I've highlighted the new line in *bold*.)

```
...
 <link href='{{ STATIC_URL }}bootstrap/css/bootstrap.min.css' media='screen'
 rel='stylesheet' />
 <link href='{{ STATIC_URL }}dashboard/css/{% choose_css %}' media='screen'
 rel='stylesheet' />
 <link href='{{ STATIC_URL }}dashboard/css/custom.css' media='screen' rel=
'stylesheet' />
 ...
```

8. Restart apache just for good measure: `sudo service apache2 restart`

9. Reload the dashboard in your browser and fine tune your CSS appropriate.

You're done!

OpenStack Dashboard Session Storage

Horizon uses Django's sessions framework for handling user session data; however that's not the end of the story. There are numerous session backends available, which are controlled through the SESSION_ENGINE setting in your local_settings.py file. What follows is a quick discussion of the pros and cons of each of the common options as they pertain to deploying Horizon specifically.

Local Memory Cache

Local memory storage is the quickest and easiest session backend to set up, as it has no external dependencies whatsoever. However, it has two significant drawbacks:

1. No shared storage across processes or workers.

2. No persistence after a process terminates.

The local memory backend is enabled as the default for Horizon solely because it has no dependencies. It is not recommended for production use, or even for serious development work. Enabled by:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
    'BACKEND': 'django.core.cache.backends.locmem.LocMemCache'
}
```

Memcached

External caching using an application such as memcached offers persistence and shared storage, and can be very useful for small-scale deployment and/or development. However, for distributed and high-availability scenarios memcached has inherent problems which are beyond the scope of this documentation.

Memcached is an extremely fast and efficient cache backend for cases where it fits the deployment need, but it's not appropriate for all scenarios.

Requirements:

1. Memcached service running and accessible.

2. Python memcached module installed.

Enabled by:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
    'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache'
    'LOCATION': 'my_memcached_host:11211',
}
```

Database

Database-backed sessions are scalable (using an appropriate database strategy), persistent, and can be made high-concurrency and highly-available.

The downside to this approach is that database-backed sessions are one of the slower session storages, and incur a high overhead under heavy usage. Proper configuration of your database deployment can also be a substantial undertaking and is far beyond the scope of this documentation. To enable, follow the below steps to initialise the database and configure it for use

Start the mysql command line client by running:

```
$ mysql -u root -p
```

Enter the MySQL root user's password when prompted.

To configure the MySQL database, create the dash database.

```
mysql> CREATE DATABASE dash;
```

Create a MySQL user for the newly-created dash database that has full control of the database.

```
mysql> GRANT ALL ON dash.* TO 'dash'@'%' IDENTIFIED BY
        'yourpassword';
```

Enter quit at the `mysql>` prompt to exit MySQL.

In the `/etc/openstack-dashboard/local_settings.py` file, change these options:

```
SESSION_ENGINE = 'django.core.cache.backends.db.DatabaseCache'
DATABASES = {
    'default': {
        # Database configuration here
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'dash',
        'USER': 'dash',
        'PASSWORD': 'yourpassword',
        'HOST': 'localhost',
        'default-character-set': 'utf8'
    }
}
```

After configuring the `local_settings.py` as shown, you can run the **manage.py syncdb** command to populate this newly-created database.

```
$ /usr/share/openstack-dashboard/manage.py syncdb
```

As a result, you should see the following at the end of what returns:

```
Installing custom SQL ...
Installing indexes ...
DEBUG:django.db.backends:(0.008) CREATE INDEX `django_session_c25c2c28` ON
 `django_session` (`expire_date`);; args=()
No fixtures found.
```

If you want to avoid a warning when restarting apache2, create a blackhole directory in the dashboard directory like so:

```
#  sudo mkdir -p /var/lib/dash/.blackhole
```

Restart Apache to pick up the default site and symbolic link settings.

```
# /etc/init.d/apache2 restart
```

Restart the nova-api service to ensure the API server can connect to the Dashboard and to avoid an error displayed in the Dashboard.

```
sudo restart nova-api
```

Cached Database

To mitigate the performance issues of database queries, you can also consider using Django's cached_db session backend which utilizes both your database and caching infrastructure to perform write-through caching and efficient retrieval. You can enable this hybrid setting by configuring both your database and cache as discussed above and then using:

```
SESSION_ENGINE = "django.contrib.sessions.backends.cached_db"
```

Cookies

If you're using Django 1.4 or later, a new session backend is available to you which avoids server load and scaling problems: the signed_cookies backend!

This backend stores session data in a cookie which is stored by the user's browser. The backend uses a cryptographic signing technique to ensure session data is not tampered with during transport (this is not the same as encryption, session data is still readable by an attacker).

The pros of this session engine are that it doesn't require any additional dependencies or infrastructure overhead, and it scales indefinitely as long as the quantity of session data being stored fits into a normal cookie.

The biggest downside is that it places session data into storage on the user's machine and transports it over the wire. It also limits the quantity of session data which can be stored.

For a thorough discussion of the security implications of this session backend, please read the Django documentation on cookie-based sessions.

Overview of VNC Proxy

The VNC Proxy is an OpenStack component that allows users of the Compute service to access their instances through VNC clients. In Essex and beyond, there is support for both libvirt and XenServer using both Java and websocket clients.

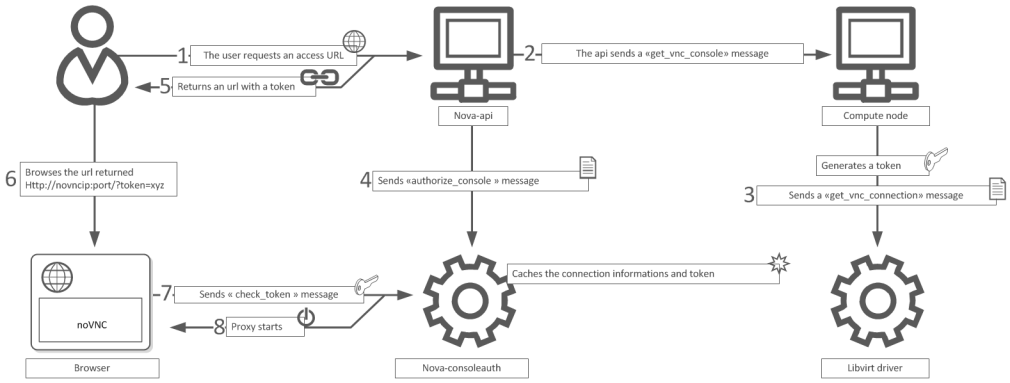
The VNC console connection works as follows:

1. User connects to API and gets an access_url like `http://ip:port/?token=xyz`.
2. User pastes URL in browser or as client parameter.
3. Browser/Client connects to proxy.
4. Proxy talks to nova-consoleauth to authorize the user's token, and then maps the token to the *private* host and port of an instance's VNC server. The compute host specifies the address the proxy should use to connect via the `nova.conf` option vncserver_proxyclient_address. In this way, the vnc proxy works as a bridge between the public network, and the private host network.
5. Proxy initiates connection to VNC server, and continues proxying until the session ends.

The proxy also performs the required function of tunneling the VNC protocol over Websockets so that the noVNC client has a way to talk VNC. Note that in general, the VNC proxy performs multiple functions:

- Bridges between public network (where clients live) and private network (where vncservers live).
- Mediates token authentication.
- Transparently deals with hypervisor-specific connection details to provide a uniform client experience.

Figure 10.1. NoVNC Process



About nova-consoleauth

Both client proxies leverage a shared service to manage token auth called nova-consoleauth. This service must be running in order for either proxy to work. Many proxies of either type can be run against a single nova-consoleauth service in a cluster configuration.

The nova-consoleauth shared service should not be confused with nova-console, which is a XenAPI-specific service that is not used by the most recent VNC proxy architecture.

Typical Deployment

A typical deployment will consist of the following components:

- One nova-consoleauth process. Typically this runs on the controller host.
- One or more nova-novncproxy services. This supports browser-based novnc clients. For simple deployments, this service typically will run on the same machine as nova-api, since it proxies between the public network and the private compute host network.
- One or more nova-xvpvncproxy services. This supports the special Java client discussed in this document. For simple deployments, this service typically will run on the same machine as nova-api, since it proxies between the public network and the private compute host network.
- One or more compute hosts. These compute hosts must have correctly configured configuration options, as described below.

Getting an Access URL

Nova provides the ability to create access_urls through the os-consoles extension. Support for accessing this URL is provided by novaclient:

```
$ nova get-vnc-console [server_id] [novnc|xvpvnc]
```

Specify 'novnc' to retrieve a URL suitable for pasting into a web browser. Specify 'xvpvnc' for a URL suitable for pasting into the Java client.

So to request a web browser URL:

```
$ nova get-vnc-console [server_id] novnc
```

Important nova-compute Options

To enable vncproxy in your cloud, in addition to running one or both of the proxies and nova-consoleauth, you need to configure the following options in `nova.conf` on your compute hosts.

- `[no]vnc_enabled` - Defaults to enabled. If this option is disabled your instances will launch without VNC support.
- `vncserver_listen` - Defaults to `127.0.0.1`. This is the address that vncservers will bind, and should be overridden in production deployments as a private address. Applies

to libvirt only. For multi-host libvirt deployments this should be set to a host management IP on the same network as the proxies.

Note

If you intend to support live migration, you cannot specify a specific IP address for `vncserver_listen`, because that IP address will not exist on the destination host. The result is that live migration will fail and the following error will appear in the `libvirtd.log` file in the destination host:

```
error: qemuMonitorIORead:513 : Unable to read from monitor:
 Connection reset by peer
```

If you wish to support live migration in your deployment, you must specify a value of `0.0.0.0` for `vncserver_listen`.

- `vncserver_proxyclient_address` - Defaults to `127.0.0.1`. This is the address of the compute host that nova will instruct proxies to use when connecting to instance vncservers. For all-in-one XenServer domU deployments this can be set to 169.254.0.1. For multi-host XenServer domU deployments this can be set to a dom0 management ip on the same network as the proxies. For multi-host libvirt deployments this can be set to a host management IP on the same network as the proxies.
- `novncproxy_base_url=[base url for client connections]` - This is the public base URL to which clients will connect. "?token=abc" will be added to this URL for the purposes of auth. When using the system as described in this document, an appropriate value is "http://$SERVICE_HOST:6080/vnc_auto.html" where *SERVICE_HOST* is a public hostname.
- `xvpvncproxy_base_url`=[base url for client connections] - This is the public base URL to which clients will connect. "?token=abc" will be added to this URL for the purposes of auth. When using the system as described in this document, an appropriate value is "http://$SERVICE_HOST:6081/console" where SERVICE_HOST is a public hostname.

Accessing VNC Consoles with a Java client

To enable support for the OpenStack Java VNC client in Compute, we provide the **nova-xvpvncproxy** service, which you should run to enable this feature.

- `xvpvncproxy_port=[port]` - port to bind (defaults to 6081)
- `xvpvncproxy_host=[host]` - host to bind (defaults to 0.0.0.0)

As a client, you will need a special Java client, which is a version of TightVNC slightly modified to support our token auth:

```
$ git clone https://github.com/cloudbuilders/nova-xvpvncviewer
$ cd nova-xvpvncviewer
$ make
```

Then, to create a session, first request an access URL using **python-novaclient** and then run the client like so. To retrieve access URL:

```
$ nova get-vnc-console [server_id] xvpvnc
```

To run client:

```
$ java -jar VncViewer.jar [access_url]
```

nova-novncproxy (novnc)

You will need the novnc package installed, which contains the nova-novncproxy service. As root:

```
# apt-get install novnc
```

The service should start automatically on install. To restart it:

```
# service novnc restart
```

The configuration option parameter should point to your `nova.conf` configuration file that includes the message queue server address and credentials.

By default, **nova-novncproxy** binds on `0.0.0.0:6080`.

In order to connect the service to your nova deployment, add the two following configuration options into your `nova.conf` file :

- `vncserver_listen=`*`0.0.0.0`*

  This configuration option allow you to specify the address for the vnc service to bind on, make sure it is assigned one of the compute node interfaces. This address will be the one used by your domain file.

  ```
  <graphics type="vnc" autoport="yes" keymap="en-us"
  listen="0.0.0.0"/>
  ```

  ### Note

  In order to have the live migration working, make sure to use the *`0.0.0.0`*address.

- `vncserver_ proxyclient_ address =`*`127.0.0.1`*

  This is the address of the compute host that nova will instruct proxies to use when connecting to instance vncservers.

Note

The previous vnc proxy implementation, called nova-vncproxy, has been deprecated.

Accessing a VNC console through a web browser

Retrieving an access_url for a web browser is similar to the flow for the Java client. To retrieve the access URL:

```
$ nova get-vnc-console [server_id] novnc
```

Then, paste the URL into your web browser.

Additionally, you can use the OpenStack Dashboard (codenamed Horizon), to access browser-based VNC consoles for instances.

Frequently asked questions about VNC access to VMs

- **Q: What has changed since Diablo?**

  A: Previously, VNC support was done differently for libvirt and XenAPI. Now, there is unified multi-hypervisor support. To support this change, configuration options have been added and changed. Also, a new required service called nova-consoleauth has been added. If you are upgrading from Diablo, you will have to take these changes into consideration when upgrading.

  If you are using Diablo, please see the documentation that shipped with your code, as this information will not be relevant.

- **Q: What happened to Diablo's nova-vncproxy?**

  A: nova-vncproxy was removed from the nova source tree. The Essex analog for this process is nova-novncproxy, which is provided by an external project.

- **Q: Why is nova-vncproxy no longer part of nova?**

  A: In Diablo, we shipped a websocket proxy (nova-vncproxy) with nova, but it had poor browser support. This nova-vncproxy code was dependent on external noVNC code, so changes to that system involved updating 2 projects. Due to the rapid evolution of websocket tech, and the tight dependence of the websocket proxy on javscript and html components, we decided to keep that code all in one place.

- **Q: What is the difference between nova-xvpvncproxy and nova-novncproxy?**

  A: nova-xvpvncproxy which ships with nova, is a new proxy that supports a simple Java client. nova-novncproxy uses noVNC to provide vnc support through a web browser.

- **Q: I want VNC support in the Dashboard. What services do I need?**

  A: You need nova-novncproxy, nova-consoleauth, and correctly configured compute hosts.

- **Q: When I use nova get-vnc-console or click on the VNC tab of the Dashboard, it hangs. Why?**

  A: Make sure you are running nova-consoleauth (in addition to nova-novncproxy). The proxies rely on nova-consoleauth to validate tokens, and will wait for a reply from them until a timeout is reached.

- **Q: My vnc proxy worked fine during my All-In-One test, but now it doesn't work on multi host. Why?**

  A: The default options work for an All-In-One install, but changes must be made on your compute hosts once you start to build a cluster. As an example, suppose you have two servers:

```
PROXYSERVER (public_ip=172.24.1.1, management_ip=192.168.1.1)
COMPUTESERVER (management_ip=192.168.1.2)
```

Your nova-compute configuration file would need the following values:

```
# These flags help construct a connection data structure
vncserver_proxyclient_address=192.168.1.2
novncproxy_base_url=http://172.24.1.1:6080/vnc_auto.html
xvpvncproxy_base_url=http://172.24.1.1:6081/console

# This is the address where the underlying vncserver (not the proxy)
# will listen for connections.
vncserver_listen=192.168.1.2
```

Note that novncproxy_base_url and novncproxy_base_url use a public ip; this is the url that is ultimately returned to clients, who generally will not have access to your private network. Your PROXYSERVER must be able to reach vncserver_proxyclient_address, as that is the address over which the vnc connection will be proxied.

See "Important nova-compute Options" for more information.

- **Q: My noVNC does not work with recent versions of web browsers. Why?**

  A: Make sure you have python-numpy installed, which is required to support a newer version of the WebSocket protocol (HyBi-07+). Also, if you are using Diablo's nova-vncproxy, note that support for this protocol is not provided.

- **Q: How do I adjust the dimensions of the VNC window image in horizon?**

  A: These values are hard-coded in a Django HTML template. To alter them, you must edit the template file `_detail_vnc.html`. The location of this file will vary based on Linux distribution. On Ubuntu 12.04, the file can be found at `/usr/share/pyshared/horizon/dashboards/nova/templates/nova/instances_and_volumes/instances/_detail_vnc.html`.

  Modify the `width` and `height` parameters:

```
<iframe src="{{ vnc_url }}" width="720" height="430"></iframe>
```

Appendix A. Appendix: Configuration File Examples

Included for your reference are all configuration files.

keystone.conf

The Identity service's configuration file is found in /etc/keystone/keystone.conf. This file needs to be modified after installing to use SQL for endpoint data and to replace the ADMIN key with the one created during the installation.

```
[DEFAULT]
bind_host = 0.0.0.0
public_port = 5000
admin_port = 35357
admin_token = 012345SECRET99TOKEN012345
compute_port = 8774
verbose = True
debug = True
log_config = /etc/keystone/logging.conf

# ================= Syslog Options ============================
# Send logs to syslog (/dev/log) instead of to file specified
# by `log-file`
use_syslog = False

# Facility to use. If unset defaults to LOG_USER.
# syslog_log_facility = LOG_LOCAL0

[sql]
connection = mysql://keystone:yourpassword@192.168.127.130/keystone
idle_timeout = 200
min_pool_size = 5
max_pool_size = 10
pool_timeout = 200

[ldap]
#url = ldap://localhost
#tree_dn = dc=example,dc=com
#user_tree_dn = ou=Users,dc=example,dc=com
#role_tree_dn = ou=Roles,dc=example,dc=com
#tenant_tree_dn = ou=Groups,dc=example,dc=com
#user = dc=Manager,dc=example,dc=com
#password = freeipa4all
#suffix = cn=example,cn=com

[identity]
driver = keystone.identity.backends.sql.Identity

[catalog]
driver = keystone.catalog.backends.sql.Catalog

[token]
driver = keystone.token.backends.sql.Token
```

```
# Amount of time a token should remain valid (in seconds)
expiration = 86400

[policy]
driver = keystone.policy.backends.rules.Policy

[ec2]
driver = keystone.contrib.ec2.backends.sql.Ec2

[filter:debug]
paste.filter_factory = keystone.common.wsgi:Debug.factory

[filter:token_auth]
paste.filter_factory = keystone.middleware:TokenAuthMiddleware.factory

[filter:admin_token_auth]
paste.filter_factory = keystone.middleware:AdminTokenAuthMiddleware.factory

[filter:xml_body]
paste.filter_factory = keystone.middleware:XmlBodyMiddleware.factory

[filter:json_body]
paste.filter_factory = keystone.middleware:JsonBodyMiddleware.factory

[filter:crud_extension]
paste.filter_factory = keystone.contrib.admin_crud:CrudExtension.factory

[filter:ec2_extension]
paste.filter_factory = keystone.contrib.ec2:Ec2Extension.factory

[app:public_service]
paste.app_factory = keystone.service:public_app_factory

[app:admin_service]
paste.app_factory = keystone.service:admin_app_factory

[pipeline:public_api]
pipeline = token_auth admin_token_auth xml_body json_body debug ec2_extension
 public_service

[pipeline:admin_api]
pipeline = token_auth admin_token_auth xml_body json_body debug ec2_extension
 crud_extension admin_service

[app:public_version_service]
paste.app_factory = keystone.service:public_version_app_factory

[app:admin_version_service]
paste.app_factory = keystone.service:admin_version_app_factory

[pipeline:public_version_api]
pipeline = xml_body public_version_service

[pipeline:admin_version_api]
pipeline = xml_body admin_version_service

[composite:main]
use = egg:Paste#urlmap
/v2.0 = public_api
/ = public_version_api
```

```
[composite:admin]
use = egg:Paste#urlmap
/v2.0 = admin_api
/ = admin_version_api
```

glance-registry.conf

The Image service's registry, which stores the metadata about images, is found in /etc/glance/glance-registry.conf. This file needs to be modified after installing.

```
[DEFAULT]
# Show more verbose log output (sets INFO log level output)
verbose = True

# Show debugging output in logs (sets DEBUG log level output)
debug = False

# Address to bind the registry server
bind_host = 0.0.0.0

# Port the bind the registry server to
bind_port = 9191

# Log to this file. Make sure you do not set the same log
# file for both the API and registry servers!
log_file = /var/log/glance/registry.log

# Backlog requests when creating socket
backlog = 4096

# TCP_KEEPIDLE value in seconds when creating socket.
# Not supported on OS X.
#tcp_keepidle = 600

# SQLAlchemy connection string for the reference implementation
# registry server. Any valid SQLAlchemy connection string is fine.
# See: http://www.sqlalchemy.org/docs/05/reference/sqlalchemy/
connections.html#sqlalchemy.create_engine
sql_connection = mysql://glance:YOUR_GLANCEDB_PASSWORD@192.168.206.130/glance

# Period in seconds after which SQLAlchemy should reestablish its connection
# to the database.
#
# MySQL uses a default `wait_timeout` of 8 hours, after which it will drop
# idle connections. This can result in 'MySQL Gone Away' exceptions. If you
# notice this, you can lower this value to ensure that SQLAlchemy reconnects
# before MySQL can drop the connection.
sql_idle_timeout = 3600

# Limit the api to return `param_limit_max` items in a call to a container. If
# a larger `limit` query param is provided, it will be reduced to this value.
api_limit_max = 1000

# If a `limit` query param is not provided in an api request, it will
# default to `limit_param_default`
limit_param_default = 25

# Role used to identify an authenticated user as administrator
```

```
#admin_role = admin

# ================= Syslog Options ============================

# Send logs to syslog (/dev/log) instead of to file specified
# by `log_file`
use_syslog = False

# Facility to use. If unset defaults to LOG_USER.
#syslog_log_facility = LOG_LOCAL1

# ================= SSL Options ===============================

# Certificate file to use when starting registry server securely
#cert_file = /path/to/certfile

# Private key file to use when starting registry server securely
#key_file = /path/to/keyfile

# CA certificate file to use to verify connecting clients
#ca_file = /path/to/cafile

[keystone_authtoken]
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = admin
admin_password = secretword

[paste_deploy]
# Name of the paste configuration file that defines the available pipelines
config_file = /etc/glance/glance-registry-paste.ini

# Partial name of a pipeline in your paste configuration file with the
# service name removed. For example, if your paste section name is
# [pipeline:glance-api-keystone], you would configure the flavor below
# as 'keystone'.
flavor=keystone
```

glance-registry-paste.ini

The Identity service's API middleware pipeline is found in /etc/glance/glance-registry-paste.ini. This file needs to be modified after installing.

```
# Use this pipeline for no auth - DEFAULT
# [pipeline:glance-registry]
# pipeline = unauthenticated-context registryapp

# Use this pipeline for keystone auth
[pipeline:glance-registry-keystone]
pipeline = authtoken context registryapp

[app:registryapp]
paste.app_factory = glance.registry.api.v1:API.factory

[filter:context]
paste.filter_factory = glance.api.middleware.context:ContextMiddleware.factory
```

```
[filter:unauthenticated-context]
paste.filter_factory =
 glance.api.middleware.context:UnauthenticatedContextMiddleware.factory

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
```

glance-api.conf

The configuration file for the Identity API is found in /etc/glance/glance-api.conf. You need to change this file to look like this example after installing from packages.

```
[DEFAULT]
# Show more verbose log output (sets INFO log level output)
verbose = True

# Show debugging output in logs (sets DEBUG log level output)
debug = False

# Which backend scheme should Glance use by default is not specified
# in a request to add a new image to Glance? Known schemes are determined
# by the known_stores option below.
# Default: 'file'
default_store = file

# List of which store classes and store class locations are
# currently known to glance at startup.
#known_stores = glance.store.filesystem.Store,
#               glance.store.http.Store,
#               glance.store.rbd.Store,
#               glance.store.s3.Store,
#               glance.store.swift.Store,


# Maximum image size (in bytes) that may be uploaded through the
# Glance API server. Defaults to 1 TB.
# WARNING: this value should only be increased after careful consideration
# and must be set to a value under 8 EB (9223372036854775808).
#image_size_cap = 1099511627776

# Address to bind the API server
bind_host = 0.0.0.0

# Port the bind the API server to
bind_port = 9292

# Log to this file. Make sure you do not set the same log
# file for both the API and registry servers!
log_file = /var/log/glance/api.log

# Backlog requests when creating socket
backlog = 4096

# TCP_KEEPIDLE value in seconds when creating socket.
# Not supported on OS X.
#tcp_keepidle = 600

# SQLAlchemy connection string for the reference implementation
# registry server. Any valid SQLAlchemy connection string is fine.
```

```
# See: http://www.sqlalchemy.org/docs/05/reference/sqlalchemy/
connections.html#sqlalchemy.create_engine
# sql_connection = sqlite:///glance.sqlite
# sql_connection = sql_connection = mysql://
glance:YOUR_GLANCEDB_PASSWORD@192.168.206.130/glance

# Period in seconds after which SQLAlchemy should reestablish its connection
# to the database.
#
# MySQL uses a default `wait_timeout` of 8 hours, after which it will drop
# idle connections. This can result in 'MySQL Gone Away' exceptions. If you
# notice this, you can lower this value to ensure that SQLAlchemy reconnects
# before MySQL can drop the connection.
sql_idle_timeout = 3600

# Number of Glance API worker processes to start.
# On machines with more than one CPU increasing this value
# may improve performance (especially if using SSL with
# compression turned on). It is typically recommended to set
# this value to the number of CPUs present on your machine.
workers = 1

# Role used to identify an authenticated user as administrator
#admin_role = admin

# Allow unauthenticated users to access the API with read-only
# privileges. This only applies when using ContextMiddleware.
#allow_anonymous_access = False

# Allow access to version 1 of glance api
#enable_v1_api = True

# Allow access to version 2 of glance api
#enable_v2_api = True

# ================ Syslog Options ============================

# Send logs to syslog (/dev/log) instead of to file specified
# by `log_file`
use_syslog = False

# Facility to use. If unset defaults to LOG_USER.
#syslog_log_facility = LOG_LOCAL0

# ================ SSL Options ==============================

# Certificate file to use when starting API server securely
#cert_file = /path/to/certfile

# Private key file to use when starting API server securely
#key_file = /path/to/keyfile

# CA certificate file to use to verify connecting clients
#ca_file = /path/to/cafile

# ================ Security Options ==========================

# AES key for encrypting store 'location' metadata, including
# -- if used -- Swift or S3 credentials
# Should be set to a random string of length 16, 24 or 32 bytes
```

```
#metadata_encryption_key = <16, 24 or 32 char registry metadata key>

# ============ Registry Options ===============================

# Address to find the registry server
registry_host = 0.0.0.0

# Port the registry server is listening on
registry_port = 9191

# What protocol to use when connecting to the registry server?
# Set to https for secure HTTP communication
registry_client_protocol = http

# The path to the key file to use in SSL connections to the
# registry server, if any. Alternately, you may set the
# GLANCE_CLIENT_KEY_FILE environ variable to a filepath of the key file
#registry_client_key_file = /path/to/key/file

# The path to the cert file to use in SSL connections to the
# registry server, if any. Alternately, you may set the
# GLANCE_CLIENT_CERT_FILE environ variable to a filepath of the cert file
#registry_client_cert_file = /path/to/cert/file

# The path to the certifying authority cert file to use in SSL connections
# to the registry server, if any. Alternately, you may set the
# GLANCE_CLIENT_CA_FILE environ variable to a filepath of the CA cert file
#registry_client_ca_file = /path/to/ca/file

# ============ Notification System Options =====================

# Notifications can be sent when images are create, updated or deleted.
# There are three methods of sending notifications, logging (via the
# log_file directive), rabbit (via a rabbitmq queue), qpid (via a Qpid
# message queue), or noop (no notifications sent, the default)
notifier_strategy = noop

# Configuration options if sending notifications via rabbitmq (these are
# the defaults)
rabbit_host = localhost
rabbit_port = 5672
rabbit_use_ssl = false
rabbit_userid = guest
rabbit_password = guest
rabbit_virtual_host = /
rabbit_notification_exchange = glance
rabbit_notification_topic = glance_notifications
rabbit_durable_queues = False

# Configuration options if sending notifications via Qpid (these are
# the defaults)
qpid_notification_exchange = glance
qpid_notification_topic = glance_notifications
qpid_host = localhost
qpid_port = 5672
qpid_username =
qpid_password =
qpid_sasl_mechanisms =
qpid_reconnect_timeout = 0
qpid_reconnect_limit = 0
```

```
qpid_reconnect_interval_min = 0
qpid_reconnect_interval_max = 0
qpid_reconnect_interval = 0
qpid_heartbeat = 5
# Set to 'ssl' to enable SSL
qpid_protocol = tcp
qpid_tcp_nodelay = True

# ============ Filesystem Store Options ========================

# Directory that the Filesystem backend store
# writes image data to
filesystem_store_datadir = /var/lib/glance/images/

# ============ Swift Store Options =============================

# Version of the authentication service to use
# Valid versions are '2' for keystone and '1' for swauth and rackspace
swift_store_auth_version = 2

# Address where the Swift authentication service lives
# Valid schemes are 'http://' and 'https://'
# If no scheme specified,  default to 'https://'
# For swauth, use something like '127.0.0.1:8080/v1.0/'
swift_store_auth_address = 127.0.0.1:5000/v2.0/

# User to authenticate against the Swift authentication service
# If you use Swift authentication service, set it to 'account':'user'
# where 'account' is a Swift storage account and 'user'
# is a user in that account
swift_store_user = jdoe:jdoe

# Auth key for the user authenticating against the
# Swift authentication service
swift_store_key = a86850deb2742ec3cb41518e26aa2d89

# Container within the account that the account should use
# for storing images in Swift
swift_store_container = glance

# Do we create the container if it does not exist?
swift_store_create_container_on_put = False

# What size, in MB, should Glance start chunking image files
# and do a large object manifest in Swift? By default, this is
# the maximum object size in Swift, which is 5GB
swift_store_large_object_size = 5120

# When doing a large object manifest, what size, in MB, should
# Glance write chunks to Swift? This amount of data is written
# to a temporary disk buffer during the process of chunking
# the image file, and the default is 200MB
swift_store_large_object_chunk_size = 200

# Whether to use ServiceNET to communicate with the Swift storage servers.
# (If you aren't RACKSPACE, leave this False!)
#
# To use ServiceNET for authentication, prefix hostname of
# `swift_store_auth_address` with 'snet-'.
# Ex. https://example.com/v1.0/ -> https://snet-example.com/v1.0/
```

```
swift_enable_snet = False

# If set to True enables multi-tenant storage mode which causes Glance images
# to be stored in tenant specific Swift accounts.
#swift_store_multi_tenant = False

# A list of tenants that will be granted read/write access on all Swift
# containers created by Glance in multi-tenant mode.
#swift_store_admin_tenants = []

# The region of the swift endpoint to be used for single tenant. This setting
# is only necessary if the tenant has multiple swift endpoints.
#swift_store_region =

# ============ S3 Store Options =============================

# Address where the S3 authentication service lives
# Valid schemes are 'http://' and 'https://'
# If no scheme specified,  default to 'http://'
s3_store_host = 127.0.0.1:8080/v1.0/

# User to authenticate against the S3 authentication service
s3_store_access_key = <20-char AWS access key>

# Auth key for the user authenticating against the
# S3 authentication service
s3_store_secret_key = <40-char AWS secret key>

# Container within the account that the account should use
# for storing images in S3. Note that S3 has a flat namespace,
# so you need a unique bucket name for your glance images. An
# easy way to do this is append your AWS access key to "glance".
# S3 buckets in AWS *must* be lowercased, so remember to lowercase
# your AWS access key if you use it in your bucket name below!
s3_store_bucket = <lowercased 20-char aws access key>glance

# Do we create the bucket if it does not exist?
s3_store_create_bucket_on_put = False

# When sending images to S3, the data will first be written to a
# temporary buffer on disk. By default the platform's temporary directory
# will be used. If required, an alternative directory can be specified here.
#s3_store_object_buffer_dir = /path/to/dir

# When forming a bucket url, boto will either set the bucket name as the
# subdomain or as the first token of the path. Amazon's S3 service will
# accept it as the subdomain, but Swift's S3 middleware requires it be
# in the path. Set this to 'path' or 'subdomain' - defaults to 'subdomain'.
#s3_store_bucket_url_format = subdomain

# ============ RBD Store Options =============================

# Ceph configuration file path
# If using cephx authentication, this file should
# include a reference to the right keyring
# in a client.<USER> section
rbd_store_ceph_conf = /etc/ceph/ceph.conf

# RADOS user to authenticate as (only applicable if using cephx)
rbd_store_user = glance
```

```
# RADOS pool in which images are stored
rbd_store_pool = images

# Images will be chunked into objects of this size (in megabytes).
# For best performance, this should be a power of two
rbd_store_chunk_size = 8

# ============ Delayed Delete Options =============================

# Turn on/off delayed delete
delayed_delete = False

# Delayed delete time in seconds
scrub_time = 43200

# Directory that the scrubber will use to remind itself of what to delete
# Make sure this is also set in glance-scrubber.conf
scrubber_datadir = /var/lib/glance/scrubber

# =============== Image Cache Options =============================

# Base directory that the Image Cache uses
image_cache_dir = /var/lib/glance/image-cache/

[keystone_authtoken]
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = admin
admin_password = secretword

[paste_deploy]
# Name of the paste configuration file that defines the available pipelines
config_file = /etc/glance/glance-api-paste.ini

# Partial name of a pipeline in your paste configuration file with the
# service name removed. For example, if your paste section name is
# [pipeline:glance-api-keystone], you would configure the flavor below
# as 'keystone'.
flavor=keystone
```

glance-api-paste.ini

The Identity service's API middleware pipeline is found in /etc/glance/glance-api-paste.ini. You should not need to modify this file.

```
# Use this pipeline for no auth or image caching - DEFAULT
# [pipeline:glance-api]
# pipeline = versionnegotiation unauthenticated-context rootapp

# Use this pipeline for image caching and no auth
# [pipeline:glance-api-caching]
# pipeline = versionnegotiation unauthenticated-context cache rootapp

# Use this pipeline for caching w/ management interface but no auth
# [pipeline:glance-api-cachemanagement]
```

```
# pipeline = versionnegotiation unauthenticated-context cache cachemanage
 rootapp

# Use this pipeline for keystone auth
[pipeline:glance-api-keystone]
pipeline = versionnegotiation authtoken context rootapp

# Use this pipeline for keystone auth with image caching
# [pipeline:glance-api-keystone+caching]
# pipeline = versionnegotiation authtoken context cache rootapp

# Use this pipeline for keystone auth with caching and cache management
# [pipeline:glance-api-keystone+cachemanagement]
# pipeline = versionnegotiation authtoken context cache cachemanage rootapp

[composite:rootapp]
paste.composite_factory = glance.api:root_app_factory
/: apiversions
/v1: apiv1app
/v2: apiv2app

[app:apiversions]
paste.app_factory = glance.api.versions:create_resource

[app:apiv1app]
paste.app_factory = glance.api.v1.router:API.factory

[app:apiv2app]
paste.app_factory = glance.api.v2.router:API.factory

[filter:versionnegotiation]
paste.filter_factory =
 glance.api.middleware.version_negotiation:VersionNegotiationFilter.factory

[filter:cache]
paste.filter_factory = glance.api.middleware.cache:CacheFilter.factory

[filter:cachemanage]
paste.filter_factory =
 glance.api.middleware.cache_manage:CacheManageFilter.factory

[filter:context]
paste.filter_factory = glance.api.middleware.context:ContextMiddleware.factory

[filter:unauthenticated-context]
paste.filter_factory =
 glance.api.middleware.context:UnauthenticatedContextMiddleware.factory

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
delay_auth_decision = true
```

glance-scrubber.conf

An additional configuration file for the Identity service is found in /etc/glance/glance-scrubber.conf.

```
[DEFAULT]
# Show more verbose log output (sets INFO log level output)
```

```
verbose = True

# Show debugging output in logs (sets DEBUG log level output)
debug = False

# Log to this file. Make sure you do not set the same log
# file for both the API and registry servers!
log_file = /var/log/glance/scrubber.log

# Send logs to syslog (/dev/log) instead of to file specified by `log_file`
use_syslog = False

# Delayed delete time in seconds
scrub_time = 43200

# Should we run our own loop or rely on cron/scheduler to run us
daemon = False

# Loop time between checking the registry for new items to schedule for delete
wakeup_time = 300

[app:glance-scrubber]
paste.app_factory = glance.store.scrubber:app_factory
```

nova.conf

The configuration file for Compute (nova) settings is stored in /etc/nova/nova.conf. To see a list of all possible configuration options for this file, see the List of Tables at the beginning of the OpenStack Compute Admin guide .

This guide assumes that the IP address of the machine that runs the Identity service is `192.168.206.130`. If the IP address of the machine on your setup is different, change the following configuration options:

- `service_host`
- `auth_host`
- `auth_uri`

```
[DEFAULT]

# LOGS/STATE
verbose=True
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova
rootwrap_config=/etc/nova/rootwrap.conf

# AUTHENTICATION
auth_strategy=keystone
[keystone_authtoken]
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = nova
signing_dirname = /tmp/keystone-signing-nova
```

```
# SCHEDULER
compute_scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler

# VOLUMES
volume_driver=nova.volume.driver.ISCSIDriver
volume_group=nova-volumes
volume_name_template=volume-%s
iscsi_helper=tgtadm

# DATABASE
sql_connection=mysql://nova:yourpassword@192.168.206.130/nova

# COMPUTE
libvirt_type=qemu
compute_driver=libvirt.LibvirtDriver
instance_name_template=instance-%08x
api_paste_config=/etc/nova/api-paste.ini

# COMPUTE/APIS: if you have separate configs for separate services
# this flag is required for both nova-api and nova-compute
allow_resize_to_same_host=True

# APIS
osapi_compute_extension=nova.api.openstack.compute.contrib.standard_extensions
ec2_dmz_host=192.168.206.130
s3_host=192.168.206.130

# RABBITMQ
rabbit_host=192.168.206.130

# GLANCE
image_service=nova.image.glance.GlanceImageService
glance_api_servers=192.168.206.130:9292

# NETWORK
network_manager=nova.network.manager.FlatDHCPManager
force_dhcp_release=True
dhcpbridge_flagfile=/etc/nova/nova.conf
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
# Change my_ip to match each host
my_ip=192.168.206.130
public_interface=br100
vlan_interface=eth0
flat_network_bridge=br100
flat_interface=eth0
fixed_range=192.168.100.0/24

# NOVNC CONSOLE
novncproxy_base_url=http://192.168.206.130:6080/vnc_auto.html
# Change vncserver_proxyclient_address and vncserver_listen to match each
 compute host
vncserver_proxyclient_address=192.168.206.130
vncserver_listen=192.168.206.130
```

api-paste.ini

The /etc/nova/api-paste.ini file is a middleware configuration file used by the EC2 API and OpenStack Compute API. You should not need to edit it.

```
  ############
# Metadata #
############
[composite:metadata]
use = egg:Paste#urlmap
/: meta

[pipeline:meta]
pipeline = ec2faultwrap logrequest metaapp

[app:metaapp]
paste.app_factory = nova.api.metadata.handler:MetadataRequestHandler.factory

#######
# EC2 #
#######

[composite:ec2]
use = egg:Paste#urlmap
/services/Cloud: ec2cloud

[composite:ec2cloud]
use = call:nova.api.auth:pipeline_factory
noauth = ec2faultwrap logrequest ec2noauth cloudrequest validator ec2executor
keystone = ec2faultwrap logrequest ec2keystoneauth cloudrequest validator
 ec2executor

[filter:ec2faultwrap]
paste.filter_factory = nova.api.ec2:FaultWrapper.factory

[filter:logrequest]
paste.filter_factory = nova.api.ec2:RequestLogging.factory

[filter:ec2lockout]
paste.filter_factory = nova.api.ec2:Lockout.factory

[filter:ec2keystoneauth]
paste.filter_factory = nova.api.ec2:EC2KeystoneAuth.factory

[filter:ec2noauth]
paste.filter_factory = nova.api.ec2:NoAuth.factory

[filter:cloudrequest]
controller = nova.api.ec2.cloud.CloudController
paste.filter_factory = nova.api.ec2:Requestify.factory

[filter:authorizer]
paste.filter_factory = nova.api.ec2:Authorizer.factory

[filter:validator]
paste.filter_factory = nova.api.ec2:Validator.factory

[app:ec2executor]
paste.app_factory = nova.api.ec2:Executor.factory

##############
# Openstack #
##############

[composite:osapi_compute]
```

```
use = call:nova.api.openstack.urlmap:urlmap_factory
/: oscomputeversions
/v1.1: openstack_compute_api_v2
/v2: openstack_compute_api_v2

[composite:osapi_volume]
use = call:nova.api.openstack.urlmap:urlmap_factory
/: osvolumeversions
/v1: openstack_volume_api_v1

[composite:openstack_compute_api_v2]
use = call:nova.api.auth:pipeline_factory
noauth = faultwrap sizelimit noauth ratelimit osapi_compute_app_v2
keystone = faultwrap sizelimit authtoken keystonecontext ratelimit
 osapi_compute_app_v2
keystone_nolimit = faultwrap sizelimit authtoken keystonecontext
 osapi_compute_app_v2

[composite:openstack_volume_api_v1]
use = call:nova.api.auth:pipeline_factory
noauth = faultwrap sizelimit noauth ratelimit osapi_volume_app_v1
keystone = faultwrap sizelimit authtoken keystonecontext ratelimit
 osapi_volume_app_v1
keystone_nolimit = faultwrap sizelimit authtoken keystonecontext
 osapi_volume_app_v1

[filter:faultwrap]
paste.filter_factory = nova.api.openstack:FaultWrapper.factory

[filter:noauth]
paste.filter_factory = nova.api.openstack.auth:NoAuthMiddleware.factory

[filter:ratelimit]
paste.filter_factory =
 nova.api.openstack.compute.limits:RateLimitingMiddleware.factory

[filter:sizelimit]
paste.filter_factory = nova.api.sizelimit:RequestBodySizeLimiter.factory

[app:osapi_compute_app_v2]
paste.app_factory = nova.api.openstack.compute:APIRouter.factory

[pipeline:oscomputeversions]
pipeline = faultwrap oscomputeversionapp

[app:osapi_volume_app_v1]
paste.app_factory = nova.api.openstack.volume:APIRouter.factory

[app:oscomputeversionapp]
paste.app_factory = nova.api.openstack.compute.versions:Versions.factory

[pipeline:osvolumeversions]
pipeline = faultwrap osvolumeversionapp
##########
# Shared #
##########

[filter:keystonecontext]
paste.filter_factory = nova.api.auth:NovaKeystoneContext.factory
```

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
```

Credentials (openrc)

This file contains the credentials used by Compute, Image, and Identity services, you can optionally store in /home/openrc. The important concept to avoid errors is to ensure that it is sourced in the environment from which you issue commands. Run "env | grep OS_" or "env | grep NOVA_" to view what is being used in your environment.

```
export OS_USERNAME=admin
export OS_TENANT_NAME=openstackDemo
export OS_PASSWORD=secretword
export OS_AUTH_URL=http://192.168.206.130:5000/v2.0/
export OS_REGION_NAME=RegionOne
```

Dashboard configuration

This file contains the database and configuration settings for the OpenStack Dashboard.

```
import os

DEBUG = True
TEMPLATE_DEBUG = DEBUG
PROD = False
USE_SSL = False

LOCAL_PATH = os.path.dirname(os.path.abspath(__file__))
#DATABASES = {
#    'default': {
#        'ENGINE': 'django.db.backends.sqlite3',
#        'NAME': os.path.join(LOCAL_PATH, 'dashboard_openstack.sqlite3'),
#    },
#}

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'dash',
        'USER': 'dash',
        'PASSWORD': 'yourpassword',
        'HOST': 'localhost',
        'default-character-set': 'utf8'
    },
}

CACHE_BACKEND = 'dummy://'

SESSION_ENGINE = 'django.contrib.sessions.backends.cached_db'

# Send email to the console by default
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
# Or send them to /dev/null
#EMAIL_BACKEND = 'django.core.mail.backends.dummy.EmailBackend'

# django-mailer uses a different settings attribute
MAILER_EMAIL_BACKEND = EMAIL_BACKEND
```

```
# Configure these for your outgoing email host
# EMAIL_HOST = 'smtp.my-company.com'
# EMAIL_PORT = 25
# EMAIL_HOST_USER = 'djangomail'
# EMAIL_HOST_PASSWORD = 'top-secret!'


OPENSTACK_KEYSTONE_URL = "http://localhost:5000/v2.0/"
# FIXME: this is only needed until keystone fixes its GET /tenants call
# so that it doesn't return everything for admins
OPENSTACK_KEYSTONE_ADMIN_URL = "http://localhost:35357/v2.0"
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "Member"

# NOTE(tres): Available services should come from the service
#             catalog in Keystone.
SWIFT_ENABLED = False

# Configure quantum connection details for networking
QUANTUM_ENABLED = False
QUANTUM_URL = '127.0.0.1'
QUANTUM_PORT = '9696'
QUANTUM_TENANT = '1234'
QUANTUM_CLIENT_VERSION='0.1'

# If you have external monitoring links
EXTERNAL_MONITORING = [
    ['Nagios','http://foo.com'],
    ['Ganglia','http://bar.com'],
]

# If you do not have external monitoring links
# EXTERNAL_MONITORING = []

# Uncomment the following segment to silence most logging
# django.db and boto DEBUG logging is extremely verbose.
#LOGGING = {
#        'version': 1,
#        # set to True will disable all logging except that specified, unless
#        # nothing is specified except that django.db.backends will still log,
#        # even when set to True, so disable explicitly
#        'disable_existing_loggers': False,
#        'handlers': {
#            'null': {
#                'level': 'DEBUG',
#                'class': 'django.utils.log.NullHandler',
#                },
#            'console': {
#                'level': 'DEBUG',
#                'class': 'logging.StreamHandler',
#                },
#            },
#        'loggers': {
#            # Comment or Uncomment these to turn on/off logging output
#            'django.db.backends': {
#                'handlers': ['null'],
#                'propagate': False,
#                },
#            'django_openstack': {
#                'handlers': ['null'],
```

```
#                'propagate': False,
#            },
#        }
#}

# How much ram on each compute host?
COMPUTE_HOST_RAM_GB = 16
```

etc/swift/swift.conf

This file contains the settings to randomize the hash for the ring for Object Storage, code-named swift.

```
[swift-hash]
# random unique string that can never change (DO NOT LOSE)
swift_hash_path_suffix = fLIbertYgibbitZ
```

etc/network/interfaces.conf

These instructions are for using the FlatDHCP networking mode with a single network interface.

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp

# Bridge network interface for VM networks
auto br100
iface br100 inet static
address 192.168.100.1
netmask 255.255.255.0
bridge_stp off
bridge_fd 0
```

etc/swift/proxy-server.conf

This file contains the settings for the Object Storage proxy server, which contains the Identity service settings.

```
[DEFAULT]
bind_port = 8888
user = <user>

[pipeline:main]
pipeline = healthcheck cache swift3 authtoken keystone proxy-server

[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true

[filter:keystone]
```

```
paste.filter_factory = keystone.middleware.swift_auth:filter_factory
operator_roles = Member,admin, swiftoperator

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
# Delaying the auth decision is required to support token-less
# usage for anonymous referrers ('.r:*').
delay_auth_decision = 10
service_port = 5000
service_host = 127.0.0.1
auth_port = 35357
auth_host = 127.0.0.1
auth_protocol = http
auth_uri = http://127.0.0.1:5000/
auth_token = 012345SECRET99TOKEN012345
admin_token = 012345SECRET99TOKEN012345
admin_tenant_name = service
admin_user = swift
admin_password = swift

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

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

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

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]
```

etc/swift/account-server/1.conf

```
[DEFAULT]
devices = /srv/1/node
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 = yes

[account-auditor]

[account-reaper]
```

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]
```

etc/swift/container-server/1.conf

```
[DEFAULT]
devices = /srv/1/node
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 = yes

[container-updater]
```

```
[container-auditor]
[container-sync]
```

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-expirer]
```

etc/swift/object-server/1.conf

```
[DEFAULT]
devices = /srv/1/node
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 = yes

[object-updater]

[object-auditor]

[object-expirer]
```

Appendix B. OpenStack Folsom deployment guide for Ubuntu Precise, Single node installation.

ASSUMPTION: Currently guide uses Nova-Network will be updated for Quantum soon, although we have created database and endpoint for the Quantum service.

Prerequisites

One server with two NICs

- IF1 : Public traffic
- IF2 : Private traffic

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
 address 172.16.30.20
 netmask 255.255.255.0
 gateway 172.16.30.1

# This is an autoconfigured IPv6 interface
iface eth0 inet6 auto

auto eth1
iface eth1 inet static
 address 10.211.55.20
 netmask 255.255.255.0
 gatewaty10.211.55.1
```

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

Download and install the Ubuntu Precise 12.04 LTS x86_64.

Add the following Ubuntu repository read about it on the Canonical blog. (as root):

```
    # echo deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-
updates/folsom main >> /etc/apt/sources.list.d/folsom.list
    # apt-get install ubuntu-cloud-keyring
```

```
    # apt-get update
    # apt-get upgrade
```

Install the required packages.

```
    $  sudo apt-get install vlan bridge-utils ntp mysql-server python-mysqldb
```

Enable the ip_forwarding.

```
     $  sudo vim /etc/sysctl.conf

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

Update the configuration.

```
 $ sudo sysctl -p
```

Edit /etc/ntp.conf

```
# /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help

driftfile /var/lib/ntp/ntp.drift


# Enable this if you want statistics to be logged.
#statsdir /var/log/ntpstats/

statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

# Specify one or more NTP servers.

# Use servers from the NTP Pool Project. Approved by Ubuntu Technical Board
# on 2011-02-08 (LP: #104525). See http://www.pool.ntp.org/join.html for
# more information.
server 0.ubuntu.pool.ntp.org
server 1.ubuntu.pool.ntp.org
server 2.ubuntu.pool.ntp.org
server 3.ubuntu.pool.ntp.org

# Use Ubuntu's ntp server as a fallback.
server ntp.ubuntu.com iburst
server 127.127.1.0
fudge 127.127.1.0 stratum 10

# Access control configuration; see /usr/share/doc/ntp-doc/html/accopt.html
 for
# details.  The web page <http://support.ntp.org/bin/view/Support/
AccessRestrictions>
# might also be helpful.
#
# Note that "restrict" applies to both servers and clients, so a configuration
# that might be intended to block requests from certain clients could also end
```

```
# up blocking replies from your own upstream servers.

# By default, exchange time with everybody, but don't allow configuration.
restrict -4 default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery

# Local users may interrogate the ntp server more closely.
restrict 127.0.0.1
restrict ::1

# Clients from this (example!) subnet have unlimited access, but only if
# cryptographically authenticated.
#restrict 192.168.123.0 mask 255.255.255.0 notrust


# If you want to provide time to your local subnet, change the next line.
# (Again, the address is an example only.)
#broadcast 192.168.123.255

# If you want to listen to time broadcasts on your local subnet, de-comment
 the
# next lines.  Please do this only if you trust everybody on the network!
#disable auth
#broadcastclient
```

```
 $ sudo service ntp restart
```

Edit /etc/mysql/my.cnf and uncomment this line:

```
bind-address            = 0.0.0.0
```

Restart mysql server.

```
$ sudo service mysql restart
```

Creating and adding the databases for all the services.

```
$ mysql -u root -proot -e "create database nova;"
$ mysql -u root -proot -e "create database glance;"
$ mysql -u root -proot -e "create database cinder;"
$ mysql -u root -proot -e "create database keystone;"
$ mysql -u root -proot -e "create database ovs_quantum;"
```

Add mysql permissions for the created databases.

```
 mysql > grant all privileges on nova.* to nova@"localhost" identified by
 "openstack";
 mysql > grant all privileges on nova.* to nova@"%" identified by "openstack";
 mysql > grant all privileges on glance.* to glance@"localhost" identified by
 "openstack";
 mysql > grant all privileges on glance.* to glance@"%" identified by
 "openstack";
 mysql > grant all privileges on cinder.* to cinder@"localhost" identified by
 "openstack";
 mysql > grant all privileges on cinder.* to cinder@"%" identified by
 "openstack";
 mysql > grant all privileges on keystone.* to keystone@"localhost" identified
 by "openstack";
 mysql > grant all privileges on keystone.* to keystone@"%" identified by
 "openstack";
```

```
mysql > grant all privileges on ovs_quantum.* to ovs_quantum@"localhost"
identified by "openstack";
mysql > grant all privileges on ovs_quantum.* to ovs_quantum@"%" identified
by "openstack";
```

Installing and Configuring Identity service

Install the packages.

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

Edit /etc/keystone/keystone.conf and modify Admin token, SQLAlchemy, Catalog accordingly.

```
admin_token = admin
```

```
connection = mysql://keystone:openstack@10.211.55.20/keystone
```

Restart Keystone.

```
$ sudo service keystone restart
```

Populate the database.

```
$ keystone-manage db_sync
$ sudo service keystone restart
```

Update the /home/$user/.bashrc by adding then credentials below.

```
export SERVICE_TOKEN=admin
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=openstack
export OS_AUTH_URL=http://10.211.55.20:5000/v2.0/
export SERVICE_ENDPOINT=http://10.211.55.20:35357/v2.0/
```

Source the environment setting.

```
$ source ./home/$user/.bashrc
```

Add keystone users.

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

Create roles.

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

Create tenants.

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

Create services.

```
$ 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"
$ keystone service-create --name cinder --type volume --description "Cinder
 Service"
$ keystone service-create --name quantum --type network --description
 "OpenStack Networking service"
```

Create endpoints.

```
# For Nova-api

$ keystone endpoint-create --region myregion --service_id
 bbbd1945908f4fae90530e8721df650d --publicurl "http://172.16.30.20:8774/
v2/%(tenant_id)s" --adminurl "http://10.211.55.20:8774/v2/%(tenant_id)s" --
internalurl "http://10.211.55.20:8774/v2/%(tenant_id)s"

# For Nova-volume


$ keystone endpoint-create --region myregion --service_id
 53a8ae206b3645368daa9db4fe149ee5 --publicurl "http://172.16.30.20:8776/
v1/%(tenant_id)s" --adminurl "http://10.211.55.20:8776/v1/%(tenant_id)s" --
internalurl "http://10.211.55.20:8776/v1/%(tenant_id)s"

#For Glance

$ keystone endpoint-create --region myregion --service_id
 4088ac79a42d4495977465a782fbf03f --publicurl "http://10.211.55.20:9292/v1"
 --adminurl "http://10.211.55.20:9292/v1" --internalurl "http://10.211.55.
20:9292/v1"

# For Swift

$ keystone endpoint-create --region myregion --service_id
259703bf8d3c4b5e8aad1179fa8171bd --publicurl "http://172.16.30.20:8080/v1/
AUTH_%(tenant_id)s" --adminurl "http://10.211.55.20:8080/v1" --internalurl
 "http://10.211.55.20:8080/v1/AUTH_%(tenant_id)s"

#For Identity Service

$ keystone endpoint-create --region myregion --service_id
 1f46270f7c774a0786ec6ea590d99b7c --publicurl "http://172.16.30.200:5000/v2.0"
 --adminurl "http://10.211.55.20:35357/v2.0" --internalurl "http://10.211.55.
20:5000/v2.0"
```

```
#For EC2_compatibility

$ keystone endpoint-create --region myregion --service_id
 58e531e33059482f940de8ba9e97e5d1 --publicurl "http://172.16.30.20:8773/
services/Cloud" --adminurl "http://10.211.55.20:8773/services/Admin" --
internalurl "http://10.211.55.20:8773/services/Cloud"

#For Cinder

$ keystone endpoint-create --region myregion --service_id
 65a888cf384d4c68b595196661cee87d --publicurl "http://172.16.30.20:8776/
v1/%(tenant_id)s" --adminurl "http://10.211.55.20:8776/v1/%(tenant_id)s" --
internalurl "http://10.211.55.20:8776/v1/%(tenant_id)s"
```

Retrieve all the i“

```
$ keystone tenant-list

+----------------------------------+---------+---------+
|                id                |   name  | enabled |
+----------------------------------+---------+---------+
| 2a76a11b872e4ca18adb3162924735af | service |   True  |
| 950fe8e5ed5f4659a8556ac836e8943d |  admin  |   True  |
+----------------------------------+---------+---------+

$ keystone user-list

+----------------------------------+---------+---------+--------------------+
|                id                |   name  | enabled |       email        |
+----------------------------------+---------+---------+--------------------+
| 1d64219fcdeb41c3a163a761c61ef280 |   nova  |   True  |  nova@foobar.com   |
| 223c1711de5446f9b99c71803fc488db | quantum |   True  | quantum@foobar.com |
| 45e9461fa61e48f99de1adcd0b38eae7 |  admin  |   True  |  admin@foobar.com  |
| af4a1747e71d48c7834c408678f27316 |  cinder |   True  | cinder@foobar.com  |
| ceade796dee047b8b3488661a29f23cd |  glance |   True  | glance@foobar.com  |
| e3b2c1c3082c4545888329d0862ffcf1 |  swift  |   True  |  swift@foobar.com  |
+----------------------------------+---------+---------+--------------------+

$ keystone role-list

+----------------------------------+--------+
|                id                |  name  |
+----------------------------------+--------+
| de031f37231b4d4cafb0af9f56dba100 | Member |
| e45af7cf33be4dac8070aa8310144ce3 | admin  |
+----------------------------------+--------+

$ keystone service-list
```

```
+----------------------------------+----------+--------------
+------------------------------+
|                id                |   name   |     type     |
 description          |
+----------------------------------+----------+--------------
+------------------------------+
| 1f46270f7c774a0786ec6ea590d99b7c | keystone |   identity   |  OpenStack
 Identity Service  |
| 259703bf8d3c4b5e8aad1179fa8171bd |  swift   | object-store |  OpenStack
 Storage Service   |
| 4088ac79a42d4495977465a782fbf03f |  glance  |    image     |   OpenStack
 Image Service    |
| 53a8ae206b3645368daa9db4fe149ee5 |  volume  |    volume    |   OpenStack
 Volume Service   |
| 58e531e33059482f940de8ba9e97e5d1 |   ec2    |     ec2      |         EC2
 Service          |
| 65a888cf384d4c68b595196661cee87d |  cinder  |    volume    |        Cinder
 Service        |
| 72c2cd62020f4c349e64a383b05daf8b | quantum  |   network    | OpenStack
 Networking service |
| bbbd1945908f4fae90530e8721df650d |   nova   |   compute    |  OpenStack
 Compute Service   |
+----------------------------------+----------+--------------
+------------------------------+
```

A“i“

```
# User admin <> role admin <> tenant admin

$ keystone user-role-add --user_id 45e9461fa61e48f99de1adcd0b38eae7 --role_id
 e45af7cf33be4dac8070aa8310144ce3 --tenant_id 950fe8e5ed5f4659a8556ac836e8943d

# User nova <> role admin <> tenant service

$ keystone user-role-add --user_id 1d64219fcdeb41c3a163a761c61ef280 --role_id
 e45af7cf33be4dac8070aa8310144ce3 --tenant_id 2a76a11b872e4ca18adb3162924735af

# User glance <> role admin <> tenant service

$ keystone user-role-add --user_id e3b2c1c3082c4545888329d0862ffcf1 --role_id
 e45af7cf33be4dac8070aa8310144ce3 --tenant_id 2a76a11b872e4ca18adb3162924735af

# User swift <> role admin <> tenant service

$ keystone user-role-add --user_id e3b2c1c3082c4545888329d0862ffcf1 --role_id
 e45af7cf33be4dac8070aa8310144ce3 --tenant_id 2a76a11b872e4ca18adb3162924735af

# User admin <> role Member <> tenant admin

$ keystone user-role-add --user_id 45e9461fa61e48f99de1adcd0b38eae7 --role_id
 de031f37231b4d4cafb0af9f56dba100 --tenant_id 950fe8e5ed5f4659a8556ac836e8943d

 # User cinder <> role admin <> tenant service

$ keystone user-role-add --user_id af4a1747e71d48c7834c408678f27316 --role_id
 e45af7cf33be4dac8070aa8310144ce3 --tenant_id 2a76a11b872e4ca18adb3162924735af




 # User swift <> role Member <> tenant service
```

```
$ keystone user-role-add --user_id e3b2c1c3082c4545888329d0862ffcf1 --role_id
 de031f37231b4d4cafb0af9f56dba100 --tenant_id 950fe8e5ed5f4659a8556ac836e8943d
```

NOTE: All These ID will differ in your configuration - use the appropriate command and retrieve all the IDs.

Installing and configuring Image Service

Install the packages.

```
$ sudo apt-get install glance glance-api python-glanceclient glance-common
 glance-registry python-glance
```

Edit /etc/glance/glance-api-paste.ini (filter authoken).

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_host = 10.211.55.20
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = admin
admin_password = openstack
```

Edit /etc/glance/glance-registry-paste.ini (filter authtoken).

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_host = 10.211.55.20
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = admin
admin_password = openstack
```

Edit /etc/glance/glance-api.conf.

SQLAlchemy part of config.

```
sql_connection = mysql://glance:openstack@10.211.55.20/glance
```

Append the following lines.

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

Edit /etc/glance/glance-registry.conf.

SQLAlchemy part of config.

```
sql_connection = mysql://glance:openstack@10.211.55.20/glance
```

Append the following lines.

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

Populate the database.

```
$ sudo glance-manage db_sync
```

Restart all services.

```
$ sudo service glance-api restart; sudo service glance-registry restart
```

Testing Glance configuration.

```
$ glance index
```

If nothing is returned, then it is working.

Installing and configuring Compute

Install the packages.

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

Update /etc/nova/nova.conf.

```
[DEFAULT]
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/run/lock/nova
verbose=True
api_paste_config=/etc/nova/api-paste.ini
scheduler_driver=nova.scheduler.simple.SimpleScheduler
s3_host=10.211.55.20
ec2_host=10.211.55.20
ec2_dmz_host=10.211.55.20
rabbit_host=10.211.55.20
cc_host=10.211.55.20
nova_url=http://10.211.55.20:8774/v1.1/
sql_connection=mysql://nova:openstack@10.211.55.20/nova
ec2_url=http://10.211.55.20:8773/services/Cloud

# Auth
use_deprecated_auth=false
auth_strategy=keystone
keystone_ec2_url=http://10.211.55.20:5000/v2.0/ec2tokens
# Imaging service
glance_api_servers=10.211.55.20:9292
image_service=nova.image.glance.GlanceImageService

# Virt driver
connection_type=libvirt
libvirt_type=qemu
libvirt_use_virtio_for_bridges=true
start_guests_on_host_boot=false
resume_guests_state_on_host_boot=false

# Vnc configuration
novnc_enabled=true
novncproxy_base_url=http://10.211.55.20:6080/vnc_auto.html
novncproxy_port=6080
vncserver_proxyclient_address=127.0.0.1
vncserver_listen=0.0.0.0

# Network settings
dhcpbridge_flagfile=/etc/nova/nova.conf
```

```
dhcpbridge=/usr/bin/nova-dhcpbridge
network_manager=nova.network.manager.VlanManager
public_interface=eth0
vlan_interface=eth1
fixed_range=192.168.4.32/27
routing_source_ip=172.16.30.20
network_size=32
force_dhcp_release=True
rootwrap_config=/etc/nova/rootwrap.conf

# Cinder #
volume_api_class=nova.volume.cinder.API
osapi_volume_listen_port=5900
```

Update the file ownership rights.

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

Edit /etc/nova/api-paste.ini (filter authtoken).

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_host = 10.211.55.20
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = openstack
signing_dirname = /tmp/keystone-signing-nova
```

Populate the database.

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

Create the private network.

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

Restart everything.

```
$ cd /etc/init.d/; for i in $( ls nova-* ); do sudo service $i restart; done
$ service open-iscsi restart
$ service novnc restart
```

Check the smiling services.

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

```
Binary           Host                 Zone             Status     State
 Updated_At
nova-consoleauth ubuntu-precise      nova             enabled    :-)
 2012-09-10 15:46:31
nova-scheduler   ubuntu-precise      nova             enabled    :-)
 2012-09-10 15:46:31
nova-compute     ubuntu-precise      nova             enabled    :-)
 2012-09-10 15:46:31
nova-network     ubuntu-precise      nova             enabled    :-)
 2012-09-10 15:46:31
nova-cert        ubuntu-precise      nova             enabled    :-)
 2012-09-10 15:46:31
```

Installing and configuring Dashboard

Install the packages.

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

Disable the quantum endpoint, as of now in our setup we are not using Quantum to do so Edit /etc/openstack-dashboard/local_settings.py - under TEMPLATE_DEBUG.

```
QUANTUM_ENABLED = False
```

Note

In order to change the timezone you can use either dashboard or inside /etc/openstack-dashboard/local_settings.py you can change the parameter below.

```
TIME_ZONE = "UTC"
```

Also be aware that the `local_settings.py` config file has comments that instruct you to configure Apache to redirect according to the location where the Horizon login screen is installed, otherwise you just see the "It works!" text on the root landing page for the web server. Here is the meaningful section in `local_settings.py`:

```
               # Default Ubuntu apache configuration uses /horizon as the
application root.
               # Configure auth redirects here accordingly.
               LOGIN_URL='/horizon/auth/login/'
               LOGIN_REDIRECT_URL='/horizon'
```

Restart the services.

```
$ sudo service apache2 restart; sudo service memcached restart
```

Logging into the dashboard with browser

```
http://127.0.0.1/horizon
```

Installing and configuring Cinder

Install the packages.

```
$ sudo apt-get install cinder-api
cinder-scheduler cinder-volume open-iscsi python-cinderclient tgt
```

Edit /etc/cinder/api-paste.init (filter authtoken).

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
service_protocol = http
service_host = 10.211.55.20
service_port = 5000
auth_host = 10.211.55.20
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = cinder
admin_password = openstack
```

Edit /etc/cinder/cinder.conf.

```
[DEFAULT]
rootwrap_config=/etc/cinder/rootwrap.conf
sql_connection = mysql://cinder:openstack@10.211.55.20/cinder
api_paste_config = /etc/cinder/api-paste.ini

iscsi_helper=tgtadm
volume_name_template = volume-%s
volume_group = cinder-volumes
verbose = True
auth_strategy = keystone
#osapi_volume_listen_port=5900
```

Verify entries in nova.conf.

```
volume_api_class=nova.volume.cinder.API
enabled_apis=ec2,osapi_compute,metadata
#MAKE SURE NO ENTRY FOR osapi_volume anywhere in nova.conf!!!
#Leaving out enabled_apis altogether is NOT sufficient, as it defaults to
 include osapi_volume
```

Setup the tgts file *NOTE: $state_path=/var/lib/cinder/ and $volumes_dir = $state_path/volumes by default and path MUST exist!*.

```
$ sudo sh -c "echo 'include $volumes_dir/*' >> /etc/tgt/conf.d/cinder.conf"
```

Restart the tgt service.

```
$ sudo restart tgt
```

Populate the database.

```
$ sudo cinder-manage db sync
```

Create a 2GB test loopfile.

```
$ sudo dd if=/dev/zero of=cinder-volumes bs=1 count=0 seek=2G
```

Mount it.

```
$ sudo losetup /dev/loop2 cinder-volumes
```

Initialise it as an lvm 'physical volume', then create the lvm 'volume group'

```
$ sudo pvcreate /dev/loop2
$ sudo vgcreate cinder-volumes /dev/loop2
```

Lets check if our volume is created.

```
$ sudo pvscan
```

```
PV /dev/loop1   VG cinder-volumes   lvm2 [2.00 GiB / 1020.00 MiB free]
  Total: 1 [2.00 GiB] / in use: 1 [2.00 GiB] / in no VG: 0 [0   ]
```

Restart the services.

```
$ sudo service cinder-volume restart
```

```
$ sudo service cinder-api restart
$ sudo service cinder-scheduler restart
```

Create a 1 GB test volume.

```
$ cinder create --display_name test 1
$ cinder list
```

```
+--------------------------------------+-----------+--------------+------
+-------------+-------------+
|                  ID                  |   Status  | Display Name | Size |
 Volume Type | Attached to |
+--------------------------------------+-----------+--------------+------
+-------------+-------------+
| 5bbad3f9-50ad-42c5-b58c-9b6b63ef3532 | available |     test     |  1   |
 None    |             |
+--------------------------------------+-----------+--------------+------
+-------------+-------------+
```

Installing and configuring Swift

Install the packages.

```
$ sudo apt-get install swift swift-proxy swift-account swift-container swift-
object xfsprogs curl python-pastedeploy python-swiftclient
```

Create a loopback device.

```
# dd if=/dev/zero of=swift-volume bs=1 count=0 seek=2G
# mkfs.xfs -i size=1024 /srv/swift-volume
```

Create a mountpoint.

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

Add an fstab entry inside /etc/fstab. ($user, replace it with your logged in Linux username)

```
/home/$user/swift-volume /mnt/swift_backend xfs loop,noatime,nodiratime,
nobarrier,logbufs=8 0 0
```

Mount it.

```
$ sudo mount -a
```

Create the backend.

```
# cd /mnt/swift_backend
$ sudo mkdir node1 node2 node3 node4
$ 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
$ sudo mkdir /run/swift
```

Configure rsync, by modifying /etc/default/rsync.

```
RSYNC_ENABLE=true
```

Create, /etc/rsyncd.conf and populate it with contents below.

```
# 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 the service.

```
$ sudo service rsync restart
```

Create /etc/rsyslog.d/10-swift.conf.

```
# Uncomment the following to have a log containing all logs together
#local1,local2,local3,local4,local5.*   /var/log/swift/all.log

# Uncomment the following to have hourly proxy logs for stats processing
#$template HourlyProxyLog,"/var/log/swift/hourly/%$YEAR%%$MONTH%%$DAY%%$HOUR%"
#local1.*;local1.!notice ?HourlyProxyLog

local1.*;local1.!notice /var/log/swift/proxy.log
local1.notice           /var/log/swift/proxy.error
local1.*                ~

local2.*;local2.!notice /var/log/swift/storage1.log
local2.notice           /var/log/swift/storage1.error
local2.*                ~

local3.*;local3.!notice /var/log/swift/storage2.log
local3.notice           /var/log/swift/storage2.error
local3.*                ~

local4.*;local4.!notice /var/log/swift/storage3.log
local4.notice           /var/log/swift/storage3.error
local4.*                ~

local5.*;local5.!notice /var/log/swift/storage4.log
local5.notice           /var/log/swift/storage4.error
local5.*
```

Append/add to /etc/rsyslog.conf.

```
$PrivDropToGroup adm
```

Configure the file permissions.

```
$ sudo mkdir -p /var/log/swift/hourly
$ sudo chown -R syslog.adm /var/log/swift
$ sudo chmod -R g+w /var/log/swift
$ sudo service rsyslog restart
```

Configure the components.

Edit /etc/swift/swift.conf.

```
[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
```

Edit /etc/swift/proxy-server.conf.

```
[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

[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 = openstack
delay_auth_decision = 0
signing_dir = /tmp/keystone-signing-swift

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

Create the account file /etc/swift/account-server/1.conf.

```
[DEFAULT]
devices = /srv/node1
mount_check = false
disable_fallocate = true
bind_port = 6012
user = swift
log_facility = LOG_LOCAL2
recon_cache_path = /var/cache/swift

[pipeline:main]
pipeline = recon account-server

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

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

[account-replicator]
vm_test_mode = yes

[account-auditor]

[account-reaper]
```

Duplicate it.

```
$ 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 the container file (/etc/swift/container-server/1.conf).

```
[DEFAULT]
devices = /srv/node1
mount_check = false
disable_fallocate = true
bind_port = 6011
```

```
user = swift
log_facility = LOG_LOCAL2
recon_cache_path = /var/cache/swift

[pipeline:main]
pipeline = recon container-server

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

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

[container-replicator]
vm_test_mode = yes

[container-updater]

[container-auditor]

[container-sync]
```

Duplicate it.

```
$ sudo cp /etc/swift/container-server/1.conf /etc/swift/container-server/2.
conf
$ sudo cp /etc/swift/container-server/1.conf /etc/swift/container-server/3.
conf
$ sudo cp /etc/swift/container-server/1.conf /etc/swift/container-server/4.
conf
$ sudo sed -i 's/6011/6021/g;s/LOCAL2/LOCAL3/g;' /etc/swift/container-server/
2.conf
$ sudo sed -i 's/6011/6031/g;s/LOCAL2/LOCAL4/g;' /etc/swift/container-server/
3.conf
$ sudo sed -i 's/6011/6041/g;s/LOCAL2/LOCAL5/g;' /etc/swift/container-server/
4.conf
```

Create the object server file (/etc/swift/object-server/1.conf).

```
[DEFAULT]
devices = /srv/node1
mount_check = false
disable_fallocate = true
bind_port = 6010
user = swift
log_facility = LOG_LOCAL2
recon_cache_path = /var/cache/swift

[pipeline:main]
pipeline = recon object-server

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

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

[object-replicator]
vm_test_mode = yes

[object-updater]
```

```
[object-auditor]
```

Duplicate it.

```
$ sudo cp /etc/swift/object-server/1.conf /etc/swift/object-server/2.conf
$ sudo cp /etc/swift/object-server/1.conf /etc/swift/object-server/3.conf
$ sudo cp /etc/swift/object-server/1.conf /etc/swift/object-server/4.conf
$ sudo sed -i 's/6010/6020/g;s/LOCAL2/LOCAL3/g;' /etc/swift/object-server/2.
conf
$ sudo sed -i 's/6010/6030/g;s/LOCAL2/LOCAL4/g;' /etc/swift/object-server/3.
conf
$ sudo sed -i 's/6010/6040/g;s/LOCAL2/LOCAL5/g;' /etc/swift/object-server/4.
conf
```

Configure the rings.

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

Create the zones and balance them.

```
$ 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
```

Restart the services. It will take sometime.

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

Check the status

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

Create a container name "test" and with the list option we can see it created.

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

```
test
```