OpenStack

Administrator Guide

(August 5, 2016)

docs.openstack.org

OpenStack Administrator Guide

OpenStack Administrator Guide

Abstract

OpenStack offers open source software for OpenStack administrators to manage and troubleshoot an OpenStack cloud.

This guide documents OpenStack Mitaka, and OpenStack Liberty releases.

Contents

Appendix

Glossary

Conventions

The OpenStack documentation uses several typesetting conventions.

Notices

Notices take these forms:

> **Note**
>
> A comment with additional information that explains a part of the text.

> **Important**
>
> Something you must be aware of before proceeding.

> **Tip**
>
> An extra but helpful piece of practical advice.
>
> **Caution**
>
> Helpful information that prevents the user from making mistakes.

> **Warning**
>
> Critical information about the risk of data loss or security issues.

Command prompts

```
$ command
```

Any user, including the `root` user, can run commands that are prefixed with the $ prompt.

```
# command
```

The `root` user must run commands that are prefixed with the # prompt. You can also prefix these commands with the **sudo** command, if available, to run them.

Get started with OpenStack

The OpenStack project is an open source cloud computing platform for all types of clouds, which aims to be simple to implement, massively scalable, and feature rich. Developers and cloud computing technologists from around the world create the OpenStack project.

OpenStack provides an Infrastructure-as-a-Service (*IaaS*) solution through a set of interrelated services. Each service offers an application programming interface (*API*) that facilitates this integration. Depending on your needs, you can install some or all services.

The following table describes the OpenStack services that make up the OpenStack

architecture:

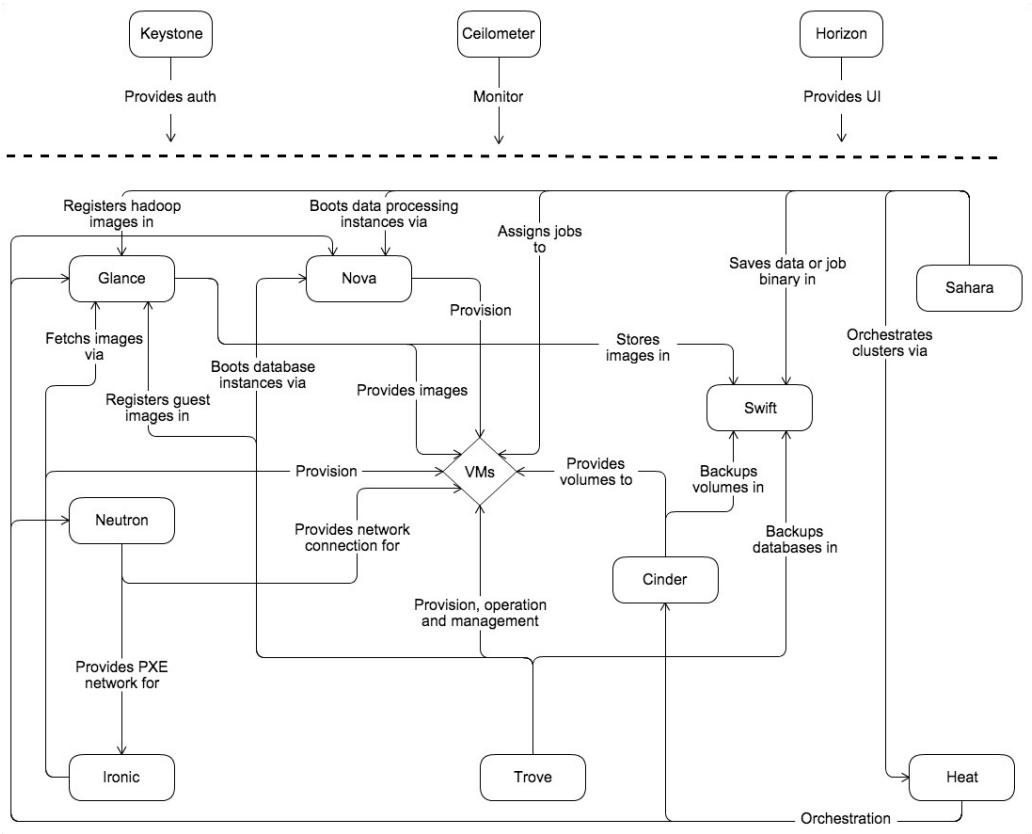
OpenStack Services

Service	Project name	Description
Dashboard	Horizon	Provides a web-based self-service portal to interact with underlying OpenStack services, such as launching an instance, assigning IP addresses and configuring access controls.
Compute	Nova	Manages the lifecycle of compute instances in an OpenStack environment. Responsibilities include spawning, scheduling and decommissioning of virtual machines on demand.
Networking	Neutron	Enables Network-Connectivity-as-a-Service for other OpenStack services, such as OpenStack Compute. Provides an API for users to define networks and the attachments into them. Has a pluggable architecture that supports many popular networking vendors and technologies.
Object Storage	Swift	Stores and retrieves arbitrary unstructured data objects via a RESTful, HTTP based API. It is highly fault tolerant with its data replication and scale-out architecture. Its implementation is not like a file server with mountable directories. In this case, it writes objects and files to multiple drives, ensuring the data is replicated across a server cluster.
Block Storage	Cinder	Provides persistent block storage to running instances. Its pluggable driver architecture facilitates the creation and management of block storage devices.
Identity service	Keystone	Provides an authentication and authorization service for other OpenStack services. Provides a catalog of endpoints for all OpenStack services.

Service	Project name	Description
Image service	Glance	Stores and retrieves virtual machine disk images. OpenStack Compute makes use of this during instance provisioning.
Telemetry	Ceilometer	Monitors and meters the OpenStack cloud for billing, benchmarking, scalability, and statistical purposes.
Orchestration	Heat	Orchestrates multiple composite cloud applications by using either the native HOT template format or the AWS CloudFormation template format, through both an OpenStack-native REST API and a CloudFormation-compatible Query API.
Database service	Trove	Provides scalable and reliable Cloud Database-as-a-Service functionality for both relational and non-relational database engines.
Data processing service	Sahara	Provides capabilities to provision and scale Hadoop clusters in OpenStack by specifying parameters like Hadoop version, cluster topology and nodes hardware details.

Conceptual architecture

The following diagram shows the relationships among the OpenStack services:



Logical architecture

To design, deploy, and configure OpenStack, administrators must understand the logical architecture.

As shown in *Conceptual architecture*, OpenStack consists of several independent parts, named the OpenStack services. All services authenticate through a common Identity service. Individual services interact with each other through public APIs, except where privileged administrator commands are necessary.

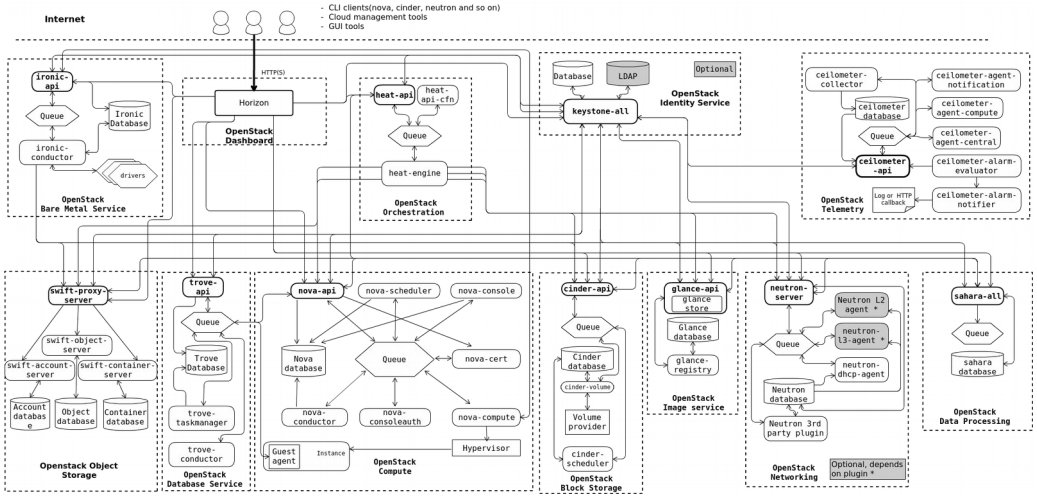
Internally, OpenStack services are composed of several processes. All services have at least one API process, which listens for API requests, preprocesses them and passes them on to other parts of the service. With the exception of the Identity service, the actual work is done by distinct processes.

For communication between the processes of one service, an AMQP message broker is used. The service's state is stored in a database. When deploying and configuring your OpenStack cloud, you can choose among several message broker and database solutions,

such as RabbitMQ, MySQL, MariaDB, and SQLite.

Users can access OpenStack via the web-based user interface implemented by *Dashboard*, via command-line clients and by issuing API requests through tools like browser plug-ins or **curl**. For applications, several SDKs are available. Ultimately, all these access methods issue REST API calls to the various OpenStack services.

The following diagram shows the most common, but not the only possible, architecture for an OpenStack cloud:



OpenStack services

- Compute service overview
- Storage concepts
- Object Storage service overview
- Block Storage service overview
- Shared File Systems service overview
- Networking service overview
- Dashboard overview
- Identity service overview
- Image service overview
- Telemetry service overview
    - Telemetry Data Collection service
    - Telemetry Alarming service
- Orchestration service overview
- Database service overview
- Data Processing service overview

This section describes OpenStack services in detail.

Compute service overview

Use OpenStack Compute to host and manage cloud computing systems. OpenStack Compute is a major part of an Infrastructure-as-a-Service (*IaaS*) system. The main modules are implemented in Python.

OpenStack Compute interacts with OpenStack Identity for authentication; OpenStack Image service for disk and server images; and OpenStack dashboard for the user and administrative interface. Image access is limited by projects, and by users; quotas are limited per project (the number of instances, for example). OpenStack Compute can scale horizontally on standard hardware, and download images to launch instances.

OpenStack Compute consists of the following areas and their components:

`nova-api` service
: Accepts and responds to end user compute API calls. The service supports the OpenStack Compute API, the Amazon EC2 API, and a special Admin API for privileged users to perform administrative actions. It enforces some policies and initiates most orchestration activities, such as running an instance.

`nova-api-metadata` service
: Accepts metadata requests from instances. The `nova-api-metadata` service is generally used when you run in multi-host mode with `nova-network` installations. For details, see Metadata service in the OpenStack Administrator Guide.

`nova-compute` service
: A worker daemon that creates and terminates virtual machine instances through hypervisor APIs. For example:

  - XenAPI for XenServer/XCP
  - libvirt for KVM or QEMU
  - VMwareAPI for VMware

  Processing is fairly complex. Basically, the daemon accepts actions from the queue and performs a series of system commands such as launching a KVM instance and updating its state in the database.

`nova-scheduler` service
: Takes a virtual machine instance request from the queue and determines on which compute server host it runs.

`nova-conductor` module
: Mediates interactions between the `nova-compute` service and the database. It eliminates direct accesses to the cloud database made by the `nova-compute` service. The `nova-conductor` module scales horizontally. However, do not deploy it on nodes where the `nova-compute` service runs. For more information, see Configuration

Reference Guide.

`nova-cert` module

A server daemon that serves the Nova Cert service for X509 certificates. Used to generate certificates for `euca-bundle-image`. Only needed for the EC2 API.

`nova-network worker` daemon

Similar to the `nova-compute` service, accepts networking tasks from the queue and manipulates the network. Performs tasks such as setting up bridging interfaces or changing IPtables rules.

`nova-consoleauth` daemon

Authorizes tokens for users that console proxies provide. See `nova-novncproxy` and `nova-xvpvncproxy`. This service must be running for console proxies to work. You can run proxies of either type against a single nova-consoleauth service in a cluster configuration. For information, see About nova-consoleauth.

`nova-novncproxy` daemon

Provides a proxy for accessing running instances through a VNC connection. Supports browser-based novnc clients.

`nova-spicehtml5proxy` daemon

Provides a proxy for accessing running instances through a SPICE connection. Supports browser-based HTML5 client.

`nova-xvpvncproxy` daemon

Provides a proxy for accessing running instances through a VNC connection. Supports an OpenStack-specific Java client.

`nova-cert` daemon

x509 certificates.

`nova` client

Enables users to submit commands as a tenant administrator or end user.

The queue

A central hub for passing messages between daemons. Usually implemented with RabbitMQ, also can be implemented with another AMQP message queue, such as ZeroMQ.

SQL database

Stores most build-time and run-time states for a cloud infrastructure, including:

- Available instance types
- Instances in use
- Available networks
- Projects

Theoretically, OpenStack Compute can support any database that SQL-Alchemy supports. Common databases are SQLite3 for test and development work, MySQL, MariaDB, and PostgreSQL.

Storage concepts

The OpenStack stack uses the following storage types:

Storage types

On-instance / ephemeral	Block storage (cinder)	Object Storage (swift)	File Storage (manila)
Runs operating systems and provides scratch space	Used for adding additional persistent storage to a virtual machine (VM)	Used for storing virtual machine images and data	Used for providing file shares to a virtual machine
Persists until VM is terminated	Persists until deleted	Persists until deleted	Persists until deleted
Access associated with a VM	Access associated with a VM	Available from anywhere	Access can be provided to a VM
Implemented as a filesystem underlying OpenStack Compute	Mounted via OpenStack Block Storage controlled protocol (for example, iSCSI)	REST API	Provides Shared File System service via nfs, cifs, glusterfs, or hdfs protocol
Encryption is available	Encryption is available	Work in progress - expected for the Mitaka release	Encryption is not available yet
Administrator configures size setting, based on flavors	Sizings based on need	Easily scalable for future growth	Sizing based on need
Example: 10 GB first disk, 30 GB/core second disk	Example: 1 TB “extra hard drive”	Example: 10s of TBs of data set storage	Example: 1 TB of file share

Note

- *You cannot use OpenStack Object Storage like a traditional hard drive.* The Object Storage relaxes some of the constraints of a POSIX-style file system to get other gains. You can access the objects through an API which uses HTTP. Subsequently you don't have to provide atomic operations (that is, relying on eventual consistency), you can scale a storage system easily and avoid a central point of failure.

- *The OpenStack Image service is used to manage the virtual machine images in an OpenStack cluster, not store them.* It provides an abstraction to different methods for storage - a bridge to the storage, not the storage itself.

- *The OpenStack Object Storage can function on its own.* The Object Storage (swift) product can be used independently of the Compute (nova) product.

Object Storage service overview

The OpenStack Object Storage is a multi-tenant object storage system. It is highly scalable and can manage large amounts of unstructured data at low cost through a RESTful HTTP API.

It includes the following components:

Proxy servers (swift-proxy-server)
: Accepts OpenStack Object Storage API and raw HTTP requests to upload files, modify metadata, and create containers. It also serves file or container listings to web browsers. To improve performance, the proxy server can use an optional cache that is usually deployed with memcache.

Account servers (swift-account-server)
: Manages accounts defined with Object Storage.

Container servers (swift-container-server)
: Manages the mapping of containers or folders, within Object Storage.

Object servers (swift-object-server)
: Manages actual objects, such as files, on the storage nodes.

Various periodic processes
: Performs housekeeping tasks on the large data store. The replication services ensure consistency and availability through the cluster. Other periodic processes include auditors, updaters, and reapers.

WSGI middleware
: Handles authentication and is usually OpenStack Identity.

swift client
Enables users to submit commands to the REST API through a command-line client authorized as either a admin user, reseller user, or swift user.

swift-init
Script that initializes the building of the ring file, takes daemon names as parameter and offers commands. Documented in http://docs.openstack.org/developer/swift/admin_guide.html#managing-services.

swift-recon
A cli tool used to retrieve various metrics and telemetry information about a cluster that has been collected by the swift-recon middleware.

swift-ring-builder
Storage ring build and rebalance utility. Documented in http://docs.openstack.org/developer/swift/admin_guide.html#managing-the-rings.

Block Storage service overview

The OpenStack Block Storage service (cinder) adds persistent storage to a virtual machine. Block Storage provides an infrastructure for managing volumes, and interacts with OpenStack Compute to provide volumes for instances. The service also enables management of volume snapshots, and volume types.

The Block Storage service consists of the following components:

cinder-api
Accepts API requests, and routes them to the `cinder-volume` for action.

cinder-volume
Interacts directly with the Block Storage service, and processes such as the `cinder-scheduler`. It also interacts with these processes through a message queue. The `cinder-volume` service responds to read and write requests sent to the Block Storage service to maintain state. It can interact with a variety of storage providers through a driver architecture.

cinder-scheduler daemon
Selects the optimal storage provider node on which to create the volume. A similar component to the `nova-scheduler`.

cinder-backup daemon
The `cinder-backup` service provides backing up volumes of any type to a backup storage provider. Like the `cinder-volume` service, it can interact with a variety of storage providers through a driver architecture.

Messaging queue
Routes information between the Block Storage processes.

Shared File Systems service overview

The OpenStack Shared File Systems service (manila) provides file storage to a virtual machine. The Shared File Systems service provides an infrastructure for managing and provisioning of file shares. The service also enables management of share types as well as share snapshots if a driver supports them.

The Shared File Systems service consists of the following components:

manila-api
: A WSGI app that authenticates and routes requests throughout the Shared File Systems service. It supports the OpenStack APIs.

manila-data
: A standalone service whose purpose is to receive requests, process data operations such as copying, share migration or backup, and send back a response after an operation has been completed.

manila-scheduler
: Schedules and routes requests to the appropriate share service. The scheduler uses configurable filters and weighers to route requests. The Filter Scheduler is the default and enables filters on things like Capacity, Availability Zone, Share Types, and Capabilities as well as custom filters.

manila-share
: Manages back-end devices that provide shared file systems. A manila-share process can run in one of two modes, with or without handling of share servers. Share servers export file shares via share networks. When share servers are not used, the networking requirements are handled outside of Manila.

Messaging queue
: Routes information between the Shared File Systems processes.

For more information, see Configuration Reference Guide.

Networking service overview

OpenStack Networking (neutron) allows you to create and attach interface devices managed by other OpenStack services to networks. Plug-ins can be implemented to accommodate different networking equipment and software, providing flexibility to OpenStack architecture and deployment.

It includes the following components:

neutron-server
: Accepts and routes API requests to the appropriate OpenStack Networking plug-in for action.

OpenStack Networking plug-ins and agents

Plugs and unplugs ports, creates networks or subnets, and provides IP addressing. These plug-ins and agents differ depending on the vendor and technologies used in the particular cloud. OpenStack Networking ships with plug-ins and agents for Cisco virtual and physical switches, NEC OpenFlow products, Open vSwitch, Linux bridging, and the VMware NSX product.

The common agents are L3 (layer 3), DHCP (dynamic host IP addressing), and a plug-in agent.

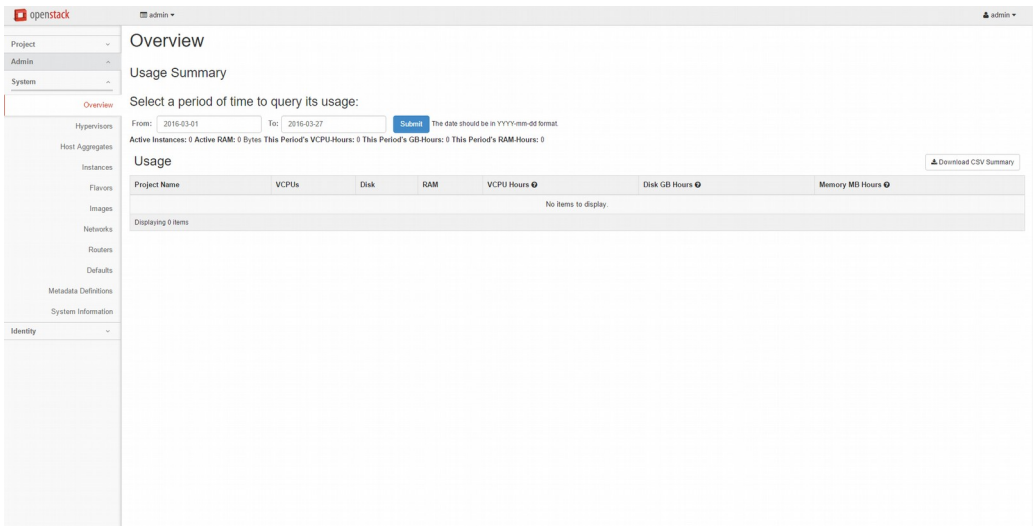
Messaging queue

Used by most OpenStack Networking installations to route information between the neutron-server and various agents. Also acts as a database to store networking state for particular plug-ins.

OpenStack Networking mainly interacts with OpenStack Compute to provide networks and connectivity for its instances.

Dashboard overview

The OpenStack dashboard is a modular Django web application that provides a graphical interface to OpenStack services.



The dashboard is usually deployed through mod_wsgi in Apache. You can modify the dashboard code to make it suitable for different sites.

From a network architecture point of view, this service must be accessible to customers and the public API for each OpenStack service. To use the administrator functionality for other services, it must also connect to Admin API endpoints, which should not be accessible by customers.

Identity service overview

The OpenStack *Identity service* provides a single point of integration for managing authentication, authorization, and service catalog services. Other OpenStack services use the Identity service as a common unified API. Additionally, services that provide information about users but that are not included in OpenStack (such as LDAP services) can be integrated into a pre-existing infrastructure.

In order to benefit from the Identity service, other OpenStack services need to collaborate with it. When an OpenStack service receives a request from a user, it checks with the Identity service whether the user is authorized to make the request.

The Identity service contains these components:

Server
: A centralized server provides authentication and authorization services using a RESTful interface.

Drivers
: Drivers or a service back end are integrated to the centralized server. They are used for accessing identity information in repositories external to OpenStack, and may already exist in the infrastructure where OpenStack is deployed (for example, SQL databases or LDAP servers).

Modules
: Middleware modules run in the address space of the OpenStack component that is using the Identity service. These modules intercept service requests, extract user credentials, and send them to the centralized server for authorization. The integration between the middleware modules and OpenStack components uses the Python Web Server Gateway Interface.

When installing OpenStack Identity service, you must register each service in your OpenStack installation. Identity service can then track which OpenStack services are installed, and where they are located on the network.

Image service overview

The OpenStack Image service is central to Infrastructure-as-a-Service (IaaS) as shown in *Conceptual architecture*. It accepts API requests for disk or server images, and metadata definitions from end users or OpenStack Compute components. It also supports the storage of disk or server images on various repository types, including OpenStack Object Storage.

A number of periodic processes run on the OpenStack Image service to support caching. Replication services ensure consistency and availability through the cluster. Other periodic processes include auditors, updaters, and reapers.

The OpenStack Image service includes the following components:

glance-api
Accepts Image API calls for image discovery, retrieval, and storage.

glance-registry

Stores, processes, and retrieves metadata about images. Metadata includes items such as size and type.

> **Warning**
>
> The registry is a private internal service meant for use by OpenStack Image service. Do not expose this service to users.

Database
Stores image metadata and you can choose your database depending on your preference. Most deployments use MySQL or SQLite.

Storage repository for image files
Various repository types are supported including normal file systems, Object Storage, RADOS block devices, HTTP, and Amazon S3. Note that some repositories will only support read-only usage.

Metadata definition service
A common API for vendors, admins, services, and users to meaningfully define their own custom metadata. This metadata can be used on different types of resources like images, artifacts, volumes, flavors, and aggregates. A definition includes the new property's key, description, constraints, and the resource types which it can be associated with.

Telemetry service overview

Telemetry Data Collection service

The Telemetry Data Collection services provide the following functions:

- Efficiently polls metering data related to OpenStack services.
- Collects event and metering data by monitoring notifications sent from services.
- Publishes collected data to various targets including data stores and message queues.

The Telemetry service consists of the following components:

A compute agent (`ceilometer-agent-compute`)
Runs on each compute node and polls for resource utilization statistics. There may be other types of agents in the future, but for now our focus is creating the compute agent.

A central agent (`ceilometer-agent-central`)
Runs on a central management server to poll for resource utilization statistics for resources not tied to instances or compute nodes. Multiple agents can be started to scale service horizontally.

A notification agent (`ceilometer-agent-notification`)
Runs on a central management server(s) and consumes messages from the message queue(s) to build event and metering data.

A collector (`ceilometer-collector`)
Runs on central management server(s) and dispatches collected telemetry data to a data store or external consumer without modification.

An API server (`ceilometer-api`)
Runs on one or more central management servers to provide data access from the data store.

Telemetry Alarming service

The Telemetry Alarming services trigger alarms when the collected metering or event data break the defined rules.

The Telemetry Alarming service consists of the following components:

An API server (`aodh-api`)
Runs on one or more central management servers to provide access to the alarm information stored in the data store.

An alarm evaluator (`aodh-evaluator`)
Runs on one or more central management servers to determine when alarms fire due to the associated statistic trend crossing a threshold over a sliding time window.

A notification listener (`aodh-listener`)
Runs on a central management server and determines when to fire alarms. The alarms are generated based on defined rules against events, which are captured by the Telemetry Data Collection service's notification agents.

An alarm notifier (`aodh-notifier`)
Runs on one or more central management servers to allow alarms to be set based on the threshold evaluation for a collection of samples.

These services communicate by using the OpenStack messaging bus. Only the collector and API server have access to the data store.

Orchestration service overview

The Orchestration service provides a template-based orchestration for describing a cloud application by running OpenStack API calls to generate running cloud applications. The software integrates other core components of OpenStack into a one-file template system.

The templates allow you to create most OpenStack resource types such as instances, floating IPs, volumes, security groups, and users. It also provides advanced functionality such as instance high availability, instance auto-scaling, and nested stacks. This enables OpenStack core projects to receive a larger user base.

The service enables deployers to integrate with the Orchestration service directly or through custom plug-ins.

The Orchestration service consists of the following components:

`heat` command-line client
: A CLI that communicates with the `heat-api` to run *AWS* CloudFormation APIs. End developers can directly use the Orchestration REST API.

`heat-api` component
: An OpenStack-native REST API that processes API requests by sending them to the `heat-engine` over *Remote Procedure Call (RPC)*.

`heat-api-cfn` component
: An AWS Query API that is compatible with AWS CloudFormation. It processes API requests by sending them to the `heat-engine` over RPC.

`heat-engine`
: Orchestrates the launching of templates and provides events back to the API consumer.

Database service overview

The Database service provides scalable and reliable cloud provisioning functionality for both relational and non-relational database engines. Users can quickly and easily use database features without the burden of handling complex administrative tasks. Cloud users and database administrators can provision and manage multiple database instances as needed.

The Database service provides resource isolation at high performance levels and automates complex administrative tasks such as deployment, configuration, patching, backups, restores, and monitoring.

Process flow example

This example is a high-level process flow for using Database services:

1. The OpenStack Administrator configures the basic infrastructure using the following steps:
    1. Install the Database service.
    2. Create an image for each type of database. For example, one for MySQL and one for MongoDB.
    3. Use the **trove-manage** command to import images and offer them to tenants.

2. The OpenStack end user deploys the Database service using the following steps:

    1. Create a Database service instance using the **trove create** command.
    2. Use the **trove list** command to get the ID of the instance, followed by the **trove show** command to get the IP address of it.
    3. Access the Database service instance using typical database access commands. For example, with MySQL:

        ```
        $ mysql -u myuser -p -h TROVE_IP_ADDRESS mydb
        ```

Components

The Database service includes the following components:

`python-troveclient` command-line client
: A CLI that communicates with the `trove-api` component.

`trove-api` component
: Provides an OpenStack-native RESTful API that supports JSON to provision and manage Trove instances.

`trove-conductor` service
: Runs on the host, and receives messages from guest instances that want to update information on the host.

`trove-taskmanager` service
: Instruments the complex system flows that support provisioning instances, managing the lifecycle of instances, and performing operations on instances.

`trove-guestagent` service
: Runs within the guest instance. Manages and performs operations on the database itself.

Data Processing service overview

The Data processing service for OpenStack (sahara) aims to provide users with a simple means to provision data processing (Hadoop, Spark) clusters by specifying several parameters like Hadoop version, cluster topology, node hardware details and a few more. After a user fills in all the parameters, the Data processing service deploys the cluster in a few minutes. Sahara also provides a means to scale already provisioned clusters by adding or removing worker nodes on demand.

The solution addresses the following use cases:

- Fast provisioning of Hadoop clusters on OpenStack for development and QA.
- Utilization of unused compute power from general purpose OpenStack IaaS cloud.
- Analytics-as-a-Service for ad-hoc or bursty analytic workloads.

Key features are:

- Designed as an OpenStack component.
- Managed through REST API with UI available as part of OpenStack dashboard.
- Support for different Hadoop distributions:
    - Pluggable system of Hadoop installation engines.
    - Integration with vendor specific management tools, such as Apache Ambari or Cloudera Management Console.
- Predefined templates of Hadoop configurations with the ability to modify parameters.
- User-friendly UI for ad-hoc analytics queries based on Hive or Pig.

Feedback

To provide feedback on documentation, join and use the openstack-docs@lists.openstack.org mailing list at OpenStack Documentation Mailing List, or report a bug.

Identity management

OpenStack Identity, code-named keystone, is the default Identity management system for OpenStack. After you install Identity, you configure it through the `/etc/keystone/keystone.conf` configuration file and, possibly, a separate logging configuration file. You initialize data into Identity by using the `keystone` command-line client.

- Identity concepts
- Certificates for PKI
- Configure the Identity service with SSL
- Domain-specific configuration
- External authentication with Identity
- Integrate Identity with LDAP
- Keystone tokens
- Configure Identity service for token binding
- Fernet - Frequently Asked Questions
- Use trusts
- Caching layer
- Start the Identity service
- Example usage and Identity features

- Authentication middleware with user name and password
- Identity API protection with role-based access control (RBAC)
- Troubleshoot the Identity service

Identity concepts

Authentication
The process of confirming the identity of a user. To confirm an incoming request, OpenStack Identity validates a set of credentials users supply. Initially, these credentials are a user name and password, or a user name and API key. When OpenStack Identity validates user credentials, it issues an authentication token. Users provide the token in subsequent requests.

Credentials
Data that confirms the identity of the user. For example, user name and password, user name and API key, or an authentication token that the Identity service provides.

Domain
An Identity service API v3 entity. Domains are a collection of projects and users that define administrative boundaries for managing Identity entities. Domains can represent an individual, company, or operator-owned space. They expose administrative activities directly to system users. Users can be granted the administrator role for a domain. A domain administrator can create projects, users, and groups in a domain and assign roles to users and groups in a domain.

Endpoint
A network-accessible address, usually a URL, through which you can access a service. If you are using an extension for templates, you can create an endpoint template that represents the templates of all consumable services that are available across the regions.

Group
An Identity service API v3 entity. Groups are a collection of users owned by a domain. A group role, granted to a domain or project, applies to all users in the group. Adding or removing users to or from a group grants or revokes their role and authentication to the associated domain or project.

OpenStackClient
A command-line interface for several OpenStack services including the Identity API. For example, a user can run the **openstack service create** and **openstack endpoint create** commands to register services in their OpenStack installation.

Project
A container that groups or isolates resources or identity objects. Depending on the service operator, a project might map to a customer, account, organization, or tenant.

Region
An Identity service API v3 entity. Represents a general division in an OpenStack deployment. You can associate zero or more sub-regions with a region to make a tree-like structured hierarchy. Although a region does not have a geographical connotation, a deployment can use a geographical name for a region, such as `us-east`.

Role
A personality with a defined set of user rights and privileges to perform a specific set of operations. The Identity service issues a token to a user that includes a list of roles. When a user calls a service, that service interprets the user role set, and determines to which operations or resources each role grants access.

Service
An OpenStack service, such as Compute (nova), Object Storage (swift), or Image service (glance), that provides one or more endpoints through which users can access resources and perform operations.

Token
An alpha-numeric text string that enables access to OpenStack APIs and resources. A token may be revoked at any time and is valid for a finite duration. While OpenStack Identity supports token-based authentication in this release, it intends to support additional protocols in the future. OpenStack Identity is an integration service that does not aspire to be a full-fledged identity store and management solution.

User
A digital representation of a person, system, or service that uses OpenStack cloud services. The Identity service validates that incoming requests are made by the user who claims to be making the call. Users have a login and can access resources by using assigned tokens. Users can be directly assigned to a particular project and behave as if they are contained in that project.

User management

Identity user management examples:

- Create a user named `alice`:

  ```
  $ openstack user create --password-prompt --email alice@example.com alice
  ```

- Create a project named `acme`:

  ```
  $ openstack project create acme
  ```

- Create a domain named `emea`:

  ```
  $ openstack --os-identity-api-version=3 domain create emea
  ```

- Create a role named `compute-user`:

  ```
  $ openstack role create compute-user
  ```

> **Note**
>
> Individual services assign meaning to roles, typically through limiting or granting access to users with the role to the operations that the service supports. Role access is typically configured in the service's `policy.json` file. For example, to limit Compute access to the `compute-user` role, edit the Compute service's `policy.json` file to require this role for Compute operations.

The Identity service assigns a tenant and a role to a user. You might assign the `compute-user` role to the `alice` user in the `acme` tenant:

```
$ openstack role add --project acme --user alice compute-user
```

A user can have different roles in different tenants. For example, Alice might also have the `admin` role in the `Cyberdyne` tenant. A user can also have multiple roles in the same tenant.

The `/etc/[SERVICE_CODENAME]/policy.json` file controls the tasks that users can perform for a given service. For example, the `/etc/nova/policy.json` file specifies the access policy for the Compute service, the `/etc/glance/policy.json` file specifies the access policy for the Image service, and the `/etc/keystone/policy.json` file specifies the access policy for the Identity service.

The default `policy.json` files in the Compute, Identity, and Image services recognize only the `admin` role. Any user with any role in a tenant can access all operations that do not require the `admin` role.

To restrict users from performing operations in, for example, the Compute service, you must create a role in the Identity service and then modify the `/etc/nova/policy.json` file so that this role is required for Compute operations.

For example, the following line in the `/etc/cinder/policy.json` file does not restrict which users can create volumes:

```
"volume:create": "",
```

If the user has any role in a tenant, he can create volumes in that tenant.

To restrict the creation of volumes to users who have the `compute-user` role in a particular tenant, you add `"role:compute-user"`:

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

To restrict all Compute service requests to require this role, the resulting file looks like:

```
{
    "admin_or_owner": "role:admin or 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",
    "compute:unlock_override": "rule:admin_api",
    "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_or_owner",
    "compute_extension:admin_actions:unlock": "rule:admin_or_owner",
    "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:evacuate": "rule:admin_api",
    "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:volume_attachments:index": "role:compute-user",
    "compute_extension:volume_attachments:show": "role:compute-user",
    "compute_extension:volume_attachments:create": "role:compute-user",
    "compute_extension:volume_attachments:delete": "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 Identity service provides identity, token, catalog, and policy services. It consists of:

- **keystone Web Server Gateway Interface (WSGI) service**

  Can be run in a WSGI-capable web server such as Apache httpd to provide the Identity service. The service and administrative APIs are run as separate instances of the WSGI service.

- **Identity service functions**

  Each has a pluggable back end that allow different ways to use the particular service. Most support standard back ends like LDAP or SQL.

- **keystone-all**

  Starts both the service and administrative APIs in a single process. Using federation with keystone-all is not supported. keystone-all is deprecated in favor of the WSGI service. Also, this will be removed in Newton.

The Identity service also maintains a user that corresponds to each service, such as, a user named `nova` for the Compute service, and a special service tenant called `service`.

For information about how to create services and endpoints, see the OpenStack Administrator Guide.

Groups

A group is a collection of users in a domain. Administrators can create groups and add users to them. A role can then be assigned to the group, rather than individual users. Groups were introduced with the Identity API v3.

Identity API V3 provides the following group-related operations:

- Create a group
- Delete a group
- Update a group (change its name or description)
- Add a user to a group
- Remove a user from a group
- List group members
- List groups for a user
- Assign a role on a tenant to a group
- Assign a role on a domain to a group
- Query role assignments to groups

> **Note**
>
> The Identity service server might not allow all operations. For example, if you use the Identity server with the LDAP Identity back end and group updates are disabled, a request to create, delete, or update a group fails.

Here are a couple of examples:

- Group A is granted Role A on Tenant A. If User A is a member of Group A, when User A gets a token scoped to Tenant A, the token also includes Role A.
- Group B is granted Role B on Domain B. If User B is a member of Group B, when User B gets a token scoped to Domain B, the token also includes Role B.

Certificates for PKI

PKI stands for Public Key Infrastructure. Tokens are documents, cryptographically signed using the X509 standard. In order to work correctly token generation requires a public/private key pair. The public key must be signed in an X509 certificate, and the certificate used to sign it must be available as a *Certificate Authority (CA)* certificate. These files can be generated either using the **keystone-manage** utility, or externally generated. The files need to be in the locations specified by the top level Identity service configuration file `keystone.conf` as specified in the above section. Additionally, the private key should only be readable by the system user that will run the Identity service.

> **Warning**
>
> The certificates can be world readable, but the private key cannot be. The private key should only be readable by the account that is going to sign tokens. When generating files with the **keystone-manage pki_setup** command, your best option is to run as the pki user. If you run **keystone-manage** as root, you can append *--keystone-user* and *--keystone-group* parameters to set the user name and group keystone is going to run under.

The values that specify where to read the certificates are under the `[signing]` section of the configuration file. The configuration values are:

- **`certfile`**

  Location of certificate used to verify tokens. Default is `/etc/keystone/ssl/certs/signing_cert.pem`.

- **`keyfile`**

  Location of private key used to sign tokens. Default is `/etc/keystone/ssl/private/signing_key.pem`.

- **`ca_certs`**

  Location of certificate for the authority that issued the above certificate. Default is `/etc/keystone/ssl/certs/ca.pem`.

- **`ca_key`**

  Location of the private key used by the CA. Default is `/etc/keystone/ssl/private/cakey.pem`.

- **`key_size`**

  Default is `2048`.

- **`valid_days`**

  Default is `3650`.

- **`cert_subject`**

  Certificate subject (auto generated certificate) for token signing. Default is `/C=US/ST=Unset/L=Unset/O=Unset/CN=www.example.com`.

When generating certificates with the **keystone-manage pki_setup** command, the

`ca_key`, `key_size`, and `valid_days` configuration options are used.

If the **keystone-manage pki_setup** command is not used to generate certificates, or you are providing your own certificates, these values do not need to be set.

If `provider=keystone.token.providers.uuid.Provider` in the `[token]` section of the keystone configuration, a typical token looks like `53f7f6ef0cc344b5be706bcc8b1479e1`. If `provider=keystone.token.providers.pki.Provider`, a typical token is a much longer string, such as:

MIIKtgYJKoZIhvcNAQcCoIIKpzCCCqMCAQExCTAHBgUrDgMCGjCCCY8GCSqGSIb3DQEHAaCCCYAEggl8
eyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMjAxMy0wNS0z
MFQxNTo1MjowNi43MzMxOTgiLCAiZXhwaXJlcyI6ICIyMDEzLTA1LTMxVDE1OjUyOjA2WiIsICJpZCI6
ICJwbGFjZWhvbGRlciIsICJ0ZW5hbnQiOiB7ImRlc2NyaXB0aW9uIjogbnVs
bCwgImVuYWJsZWQiOiB0cnVlLCAiaWQiOiAiYzJjNTliNGQzZDI4NGQ4ZmEwOWYxNjljYjE4MDBlMDYi
LCAibmFtZSI6ICJkZW1vIn19LCAic2VydmljZUNhdGFsb2ciOiBbeyJlbmRw
b2ludHMiOiBbeyJhZG1pblVSTCI6ICJodHRwOi8vMTkyLjE2OC4yNy4xMDA6ODc3NC92Mi9jMmM1OWI0
ZDNkMjg0ZDhmYTA5ZjE2OWNiMTgwMGUwNiIsICJyZWdpb24iOiAiUmVnaW9u
T25lIiwgImludGVybmFsVVJMIjogImh0dHA6Ly8xOTIuMTY4LjI3LjEwMDo4Nzc0L3YyL2MyYzU5YjRk
M2QyODRkOGZhMDlmMTY5Y2IxODAwZTA2IiwgImlkIjogIjFmYjMzYmM5M2Y5
ODRhNGNhZTk3MmViNzcwOTgzZTJlIiwgInB1YmxpY1VSTCI6ICJodHRwOi8vMTkyLjE2OC4yNy4xMDA6
ODc3NC92Mi9jMmM1OWI0ZDNkMjg0ZDhmYTA5ZjE2OWNiMTgwMGUwNiJ9XSwg
ImVuZHBvaW50c19saW5rcyI6IFtdLCAidHlwZSI6ICJjb21wdXRlIiwgIm5hbWUiOiAibm92YSJ9LCB7
ImVuZHBvaW50cyI6IFt7ImFkbWluVVJMIjogImh0dHA6Ly8xOTIuMTY4LjI3
LjEwMDozMzMzIiwgInJlZ2lvbiI6ICJSZWdpb25PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDovLzE5
Mi4xNjguMjcuMTAwOjMzMzMiLCAiaWQiOiAiN2JjMThjYzk1NWFiNDNkYjhm
MGU2YWNlNDU4NjZmMzAiLCAicHVibGljVVJMIjogImh0dHA6Ly8xOTIuMTY4LjI3LjEwMDozMzMzIn1d
LCAiZW5kcG9pbnRzX2xpbmtzIjogW10sICJ0eXBlIjogInMzIiwgIm5hbWUi
OiAiczMifSwgeyJlbmRwb2ludHMiOiBbeyJhZG1pblVSTCI6ICJodHRwOi8vMTkyLjE2OC4yNy4xMDA6
OTI5MiIsICJyZWdpb24iOiAiUmVnaW9uT25lIiwgImludGVybmFsVVJMIjog
Imh0dHA6Ly8xOTIuMTY4LjI3LjEwMDo5MjkyIiwgImlkIjogIjczODQzNTJhNTQ0MjQ1NzVhM2NkOTVk
N2E0YzNjZGY1IiwgInB1YmxpY1VSTCI6ICJodHRwOi8vMTkyLjE2OC4yNy4x
MDA6OTI5MiJ9XSwgImVuZHBvaW50c19saW5rcyI6IFtdLCAidHlwZSI6ICJpbWFnZSIsICJuYW1lIjog
ImdsYW5jZSJ9LCB7ImVuZHBvaW50cyI6IFt7ImFkbWluVVJMIjogImh0dHA6
Ly8xOTIuMTY4LjI3LjEwMDo4Nzc2L3YxL2MyYzU5YjRkM2QyODRkOGZhMDlmMTY5Y2IxODAwZTA2Iiwg
InJlZ2lvbiI6ICJSZWdpb25PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDov
LzE5Mi4xNjguMjcuMTAwOjg3NzYvdjEvYzJjNTliNGQzZDI4NGQ4ZmEwOWYxNjljYjE4MDBlMDYiLCAi
aWQiOiAiMzQ3ZWQ2ZThjMjkxNGU1MGFlMmJiNjA2YWQxNDdjNTQiLCAicHVi
bGljVVJMIjogImh0dHA6Ly8xOTIuMTY4LjI3LjEwMDo4Nzc2L3YxL2MyYzU5YjRkM2QyODRkOGZhMDlm
MTY5Y2IxODAwZTA2In1dLCAiZW5kcG9pbnRzX2xpbmtzIjogW10sICJ0eXBl
IjogInZvbHVtZSIsICJuYW1lIjogImNpbmRlciJ9LCB7ImVuZHBvaW50cyI6IFt7ImFkbWluVVJMIjog
Imh0dHA6Ly8xOTIuMTY4LjI3LjEwMDo4NzczL3NlcnZpY2VzL0FkbWluIiwg
InJlZ2lvbiI6ICJSZWdpb25PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDov
LzE5Mi4xNjguMjcuMTAwOjg3NzMvc2VydmljZXMvQ2xvdWQiLCAiaWQiOiAiMmIwZGMyYjNlY2U4NGJj
YWE1NDAzMDMzNzI5YzY3MjIiLCAicHVibGljVVJMIjogImh0dHA6Ly8xOTIuMTY4LjI3LjEwMDo4Nzcz

L3NlcnZpY2VzL0Nsb3VkIn1dLCAiZW5kcG9pbnRzX2xpbmtzIjogW10sICJ0
eXBlIjogImVjMiIsICJuYW1lIjogImVjMiJ9LCB7ImVuZHBvaW50cyI6IFt7ImFkbWluVVJMIjogImh0
dHA6Ly8xOTIuMTY4LjI3LjEwMDozNTM1Ny92Mi4wIiwgInJlZ2lvbiI6ICJS
ZWdpb25PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDovLzE5Mi4xNjguMjcuMTAwOjUwMDAvdjIuMCIs
ICJpZCI6ICJiNTY2Y2JlZjA2NjQ0ZmY2OWMyOTMxNzY2Yjc5MTIyOSIsICJw
dWJsaWNVUkwiOiAiaHR0cDovLzE5Mi4xNjguMjcuMTAwOjUwMDAvdjIuMCJ9XSwgImVuZHBvaW50c19s
aW5rcyI6IFtdLCAidHlwZSI6ICJpZGVudGl0eSIsICJuYW1lIjogImtleXN0
b25lIn1dLCAidXNlciI6IHsidXNlcm5hbWUiOiAiZGVtbyIsICJyb2xlc19saW5rcyI6IFtdLCAiaWQi
OiAiZTVhMTM3NGE4YTRmNDI4NWIzYWQ3MzQ1MWU2MDY4YjEiLCAicm9sZXMi
OiBbeyJuYW1lIjogImFub3RoZXJyb2xlIn0sIHsibmFtZSI6ICJNZW1iZXIifV0sICJuYW1lIjogImRl
bW8ifSwgIm1ldGFkYXRhIjogeyJpc19hZG1pbiI6IDAsICJyb2xlcyI6IFsi
YWRiODM3NDVkYzQzNGJhMzk5ODllNjBjOTIzYWZhMjgiLCAiMzM2ZTFiNjE1N2Y3NGFmZGJhNWUwYTYw
MWUwNjM5MmYiXX19fTGB-zCB-AIBATBcMFcxCzAJBgNVBAYTAlVTMQ4wDAYD
VQQIEwVVbnNldDEOMAwGA1UEBxMFVW5zZXQxDjAMBgNVBAoTBVVuc2V0MRgwFgYDVQQDEw93d3cuZXhh
bXBsZS5jb20CAQEwBwYFKw4DAhowDQYJKoZIhvcNAQEBBQAEgYCAHLpsEs2R
nouriuiCgFayIqCssK3SVdhOMINiuJtqv0sE-wBDFiEj-Prcudqlz-n+6q7VgV4mwMPszz39-
rwp+P5l4AjrJasUm7FrO-4l02tPLaaZXU1gBQ1jUG5e5aL5jPDP08HbCWuX6wr-QQQB
SrWY8lF3HrTcJT23sZIleg==

Sign certificate issued by external CA

You can use a signing certificate issued by an external CA instead of generated by **keystone-manage**. However, a certificate issued by an external CA must satisfy the following conditions:

- All certificate and key files must be in Privacy Enhanced Mail (PEM) format
- Private key files must not be protected by a password

When using a signing certificate issued by an external CA, you do not need to specify `key_size`, `valid_days`, and `ca_password` as they will be ignored.

The basic workflow for using a signing certificate issued by an external CA involves:

1. Request Signing Certificate from External CA
2. Convert certificate and private key to PEM if needed
3. Install External Signing Certificate

Request a signing certificate from an external CA

One way to request a signing certificate from an external CA is to first generate a PKCS #10 Certificate Request Syntax (CRS) using OpenSSL CLI.

Create a certificate request configuration file. For example, create the `cert_req.conf` file, as follows:

```
[ req ]
default_bits            = 4096
default_keyfile         = keystonekey.pem
default_md              = sha256

prompt                  = no
distinguished_name      = distinguished_name

[ distinguished_name ]
countryName             = US
stateOrProvinceName     = CA
localityName            = Sunnyvale
organizationName        = OpenStack
organizationalUnitName  = Keystone
commonName              = Keystone Signing
emailAddress            = keystone@openstack.org
```

Then generate a CRS with OpenSSL CLI. **Do not encrypt the generated private key. You must use the -nodes option.**

For example:

```
$ openssl req -newkey rsa:1024 -keyout signing_key.pem -keyform PEM \
  -out signing_cert_req.pem -outform PEM -config cert_req.conf -nodes
```

If everything is successful, you should end up with `signing_cert_req.pem` and `signing_key.pem`. Send `signing_cert_req.pem` to your CA to request a token signing certificate and make sure to ask the certificate to be in PEM format. Also, make sure your trusted CA certificate chain is also in PEM format.

Install an external signing certificate

Assuming you have the following already:

- **`signing_cert.pem`**

  (Keystone token) signing certificate in PEM format

- **`signing_key.pem`**

  Corresponding (non-encrypted) private key in PEM format

- **`cacert.pem`**

  Trust CA certificate chain in PEM format

Copy the above to your certificate directory. For example:

```
# mkdir -p /etc/keystone/ssl/certs
# cp signing_cert.pem /etc/keystone/ssl/certs/
# cp signing_key.pem /etc/keystone/ssl/certs/
# cp cacert.pem /etc/keystone/ssl/certs/
# chmod -R 700 /etc/keystone/ssl/certs
```

> **Note**
>
> Make sure the certificate directory is only accessible by root.
>
> **Note**
>
> The procedure of copying the key and cert files may be improved if done after first running **keystone-manage pki_setup** since this command also creates other needed files, such as the `index.txt` and `serial` files.
>
> Also, when copying the necessary files to a different server for replicating the functionality, the entire directory of files is needed, not just the key and cert files.

If your certificate directory path is different from the default `/etc/keystone/ssl/certs`, make sure it is reflected in the `[signing]` section of the configuration file.

Switching out expired signing certificates

The following procedure details how to switch out expired signing certificates with no cloud outages.

1. Generate a new signing key.
2. Generate a new certificate request.
3. Sign the new certificate with the existing CA to generate a new `signing_cert`.
4. Append the new `signing_cert` to the old `signing_cert`. Ensure the old certificate is in the file first.
5. Remove all signing certificates from all your hosts to force OpenStack Compute to download the new `signing_cert`.
6. Replace the old signing key with the new signing key. Move the new signing certificate above the old certificate in the `signing_cert` file.
7. After the old certificate reads as expired, you can safely remove the old signing certificate from the file.

Configure the Identity service with SSL

You can configure the Identity service to support two-way SSL.

You must obtain the x509 certificates externally and configure them.

The Identity service provides a set of sample certificates in the `examples/pki/certs` and `examples/pki/private` directories:

cacert.pem
: Certificate Authority chain to validate against.

ssl_cert.pem
: Public certificate for Identity service server.

middleware.pem
: Public and private certificate for Identity service middleware/client.

cakey.pem
: Private key for the CA.

ssl_key.pem
: Private key for the Identity service server.

> **Note**
>
> You can choose names for these certificates. You can also combine public/private keys in the same file, if you wish. These certificates are provided as an example.

Client authentication with keystone-all

> **Note**
>
> When running the Identity service as a WSGI service in a web server such as Apache HTTP Server, this configuration is done in the web server instead. In this case, the options in the `[eventlet_server_ssl]` section are ignored.
>
> The eventlet support will be removed in Newton.

When running `keystone-all`, the server can be configured to enable SSL with client authentication using the following instructions. Modify the `[eventlet_server_ssl]` section in the `/etc/keystone/keystone.conf` file. The following SSL configuration example uses the included sample certificates:

```
[eventlet_server_ssl]
enable = True
certfile = <path to keystone.pem>
keyfile = <path to keystonekey.pem>
ca_certs = <path to ca.pem>
cert_required = True
```

Options

- **`enable`**

  `True` enables SSL. Default is `False`.

- **`certfile`**

  Path to the Identity service public certificate file.

- **`keyfile`**

  Path to the Identity service private certificate file. If you include the private key in the certfile, you can omit the keyfile.

- **`ca_certs`**

  Path to the CA trust chain.

- **`cert_required`**

  Requires client certificate. Default is `False`.

Domain-specific configuration

The Identity service supports domain-specific Identity drivers. The drivers allow a domain to have its own LDAP or SQL back end. By default, domain-specific drivers are disabled.

Domain-specific Identity configuration options can be stored in domain-specific configuration files, or in the Identity SQL database using API REST calls.

> **Note**
>
> Storing and managing configuration options in an SQL database is experimental in Kilo, and added to the Identity service in the Liberty release.

Enable drivers for domain-specific configuration files

To enable domain-specific drivers, set these options in the `/etc/keystone/keystone.conf` file:

```
[identity]
domain_specific_drivers_enabled = True
domain_config_dir = /etc/keystone/domains
```

When you enable domain-specific drivers, Identity looks in the `domain_config_dir` directory for configuration files that are named as `keystone.DOMAIN_NAME.conf`. A domain without a domain-specific configuration file uses options in the primary configuration file.

Enable drivers for storing configuration options in SQL database

To enable domain-specific drivers, set these options in the `/etc/keystone/keystone.conf` file:

```
[identity]
domain_specific_drivers_enabled = True
domain_configurations_from_database = True
```

Any domain-specific configuration options specified through the Identity v3 API will override domain-specific configuration files in the `/etc/keystone/domains` directory.

Migrate domain-specific configuration files to the SQL database

You can use the `keystone-manage` command to migrate configuration options in domain-specific configuration files to the SQL database:

```
# keystone-manage domain_config_upload --all
```

To upload options from a specific domain-configuration file, specify the domain name:

```
# keystone-manage domain_config_upload --domain-name DOMAIN_NAME
```

External authentication with Identity

When Identity runs in `apache-httpd`, you can use external authentication methods that differ from the authentication provided by the identity store back end. For example, you can use an SQL identity back end together with X.509 authentication and Kerberos, instead of using the user name and password combination.

Use HTTPD authentication

Web servers, like Apache HTTP, support many methods of authentication. Identity can allow the web server to perform the authentication. The web server then passes the authenticated user to Identity by using the `REMOTE_USER` environment variable. This user must already exist in the Identity back end to get a token from the controller. To use this method, Identity should run on `apache-httpd`.

Use X.509

The following Apache configuration snippet authenticates the user based on a valid X.509 certificate from a known CA:

```
<VirtualHost _default_:5000>
    SSLEngine on
    SSLCertificateFile    /etc/ssl/certs/ssl.cert
    SSLCertificateKeyFile /etc/ssl/private/ssl.key

    SSLCACertificatePath /etc/ssl/allowed_cas
    SSLCARevocationPath  /etc/ssl/allowed_cas
    SSLUserName          SSL_CLIENT_S_DN_CN
    SSLVerifyClient      require
    SSLVerifyDepth       10
```

```
    (...)
</VirtualHost>
```

Integrate Identity with LDAP

- Integrate Identity back end with LDAP
- Integrate assignment back end with LDAP
- Secure the OpenStack Identity service connection to an LDAP back end

The OpenStack Identity service supports integration with existing LDAP directories for authentication and authorization services. LDAP back ends require initialization before configuring the OpenStack Identity service to work with it. For more information, see Setting up LDAP for use with Keystone.

When the OpenStack Identity service is configured to use LDAP back ends, you can split authentication (using the *identity* feature) and authorization (using the *assignment* feature).

The *identity* feature enables administrators to manage users and groups by each domain or the OpenStack Identity service entirely.

The *assignment* feature enables administrators to manage project role authorization using the OpenStack Identity service SQL database, while providing user authentication through the LDAP directory.

> **Important**
>
> For the OpenStack Identity service to access LDAP servers, you must enable the `authlogin_nsswitch_use_ldap` boolean value for SELinux on the server running the OpenStack Identity service. To enable and make the option persistent across reboots, set the following boolean value as the root user:

```
# setsebool -P authlogin_nsswitch_use_ldap on
```

The Identity configuration is split into two separate back ends; identity (back end for users and groups), and assignments (back end for domains, projects, roles, role assignments). To configure Identity, set options in the `/etc/keystone/keystone.conf` file. See *Integrate Identity back end with LDAP* for Identity back end configuration examples and *Integrate assignment back end with LDAP* for assignment back end configuration examples. Modify these examples as needed.

> **Note**

> Multiple back ends are supported. You can integrate the OpenStack Identity service with a single LDAP server (configure both identity and assignments to LDAP, or set identity and assignments back end with SQL or LDAP), or multiple back ends using domain-specific configuration files.

To define the destination LDAP server

1. Define the destination LDAP server in the `keystone.conf` file:

```
[ldap]
url = ldap://localhost
user = dc=Manager,dc=example,dc=org
password = samplepassword
suffix = dc=example,dc=org
use_dumb_member = False
allow_subtree_delete = False
```

2. Configure `dumb_member` to true if your environment requires the `use_dumb_member` variable.

```
[ldap]
dumb_member = cn=dumb,dc=nonexistent
```

Additional LDAP integration settings

Set these options in the `/etc/keystone/keystone.conf` file for a single LDAP server, or `/etc/keystone/domains/keystone.DOMAIN_NAME.conf` files for multiple back ends. Example configurations appear below each setting summary:

Query option

- Use `query_scope` to control the scope level of data presented (search only the first level or search an entire sub-tree) through LDAP.
- Use `page_size` to control the maximum results per page. A value of zero disables paging.
- Use `alias_dereferencing` to control the LDAP dereferencing option for queries.
- Use `chase_referrals` to override the system's default referral chasing behavior for queries.

```
[ldap]
query_scope = sub
page_size = 0
alias_dereferencing = default
chase_referrals =
```

Debug

Use `debug_level` to set the LDAP debugging level for LDAP calls. A value of zero means that debugging is not enabled.

```
[ldap]
debug_level = 0
```

> **Warning**
>
> This value is a bitmask, consult your LDAP documentation for possible values.

Connection pooling

Use `use_pool` to enable LDAP connection pooling. Configure the connection pool size, maximum retry, reconnect trials, timeout (-1 indicates indefinite wait) and lifetime in seconds.

```
[ldap]
use_pool = true
pool_size = 10
pool_retry_max = 3
pool_retry_delay = 0.1
pool_connection_timeout = -1
pool_connection_lifetime = 600
```

Connection pooling for end user authentication

Use `use_auth_pool` to enable LDAP connection pooling for end user authentication. Configure the connection pool size and lifetime in seconds.

```
[ldap]
use_auth_pool = false
auth_pool_size = 100
auth_pool_connection_lifetime = 60
```

When you have finished the configuration, restart the OpenStack Identity service.

> **Warning**
>
> During the service restart, authentication and authorization are unavailable.

Integrate Identity back end with LDAP

The Identity back end contains information for users, groups, and group member lists. Integrating the Identity back end with LDAP allows administrators to use users and groups in LDAP.

> **Important**
>
> For OpenStack Identity service to access LDAP servers, you must define the destination LDAP server in the `keystone.conf` file. For more information, see *Integrate Identity with LDAP*.

To integrate one Identity back end with LDAP

1. Enable the LDAP Identity driver in the `keystone.conf` file. This allows LDAP as an identity back end:

   ```
   [identity]
   #driver = keystone.identity.backends.sql.Identity
   driver = keystone.identity.backends.ldap.Identity
   ```

2. Create the organizational units (OU) in the LDAP directory, and define the corresponding location in the `keystone.conf` file:

   ```
   [ldap]
   user_tree_dn = ou=Users,dc=example,dc=org
   user_objectclass = inetOrgPerson

   group_tree_dn = ou=Groups,dc=example,dc=org
   group_objectclass = groupOfNames
   ```

   > **Note**
   >
   > These schema attributes are extensible for compatibility with various schemas. For example, this entry maps to the person attribute in Active Directory:
   >
   > ```
   > user_objectclass = person
   > ```

3. A read-only implementation is recommended for LDAP integration. These permissions are applied to object types in the `keystone.conf`:

   ```
   [ldap]
   user_allow_create = False
   user_allow_update = False
   ```

```
user_allow_delete = False

group_allow_create = False
group_allow_update = False
group_allow_delete = False
```

Restart the OpenStack Identity service.

> **Warning**
>
> During service restart, authentication and authorization are unavailable.

To integrate multiple Identity back ends with LDAP

1. Set the following options in the `/etc/keystone/keystone.conf` file:
    1. Enable the LDAP driver:

    ```
    [identity]
    #driver = keystone.identity.backends.sql.Identity
    driver = keystone.identity.backends.ldap.Identity
    ```

    2. Enable domain-specific drivers:

    ```
    [identity]
    domain_specific_drivers_enabled = True
    domain_config_dir = /etc/keystone/domains
    ```

2. Restart the OpenStack Identity service.

> **Warning**
>
> During service restart, authentication and authorization are unavailable.

3. List the domains using the dashboard, or the OpenStackClient CLI. Refer to the Command List for a list of OpenStackClient commands.
4. Create domains using OpenStack dashboard, or the OpenStackClient CLI.
5. For each domain, create a domain-specific configuration file in the `/etc/keystone/domains` directory. Use the file naming convention `keystone.DOMAIN_NAME.conf`, where DOMAIN_NAME is the domain name assigned in the previous step.

> **Note**
>
> The options set in the `/etc/keystone/domains/keystone.DOMAIN_NAME.conf` file will override options in the `/etc/keystone/keystone.conf` file.

6. Define the destination LDAP server in the `/etc/keystone/domains/keystone.DOMAIN_NAME.conf` file. For example:

```
[ldap]
url = ldap://localhost
user = dc=Manager,dc=example,dc=org
password = samplepassword
suffix = dc=example,dc=org
use_dumb_member = False
allow_subtree_delete = False
```

7. Create the organizational units (OU) in the LDAP directories, and define their corresponding locations in the `/etc/keystone/domains/keystone.DOMAIN_NAME.conf` file. For example:

```
[ldap]
user_tree_dn = ou=Users,dc=example,dc=org
user_objectclass = inetOrgPerson

group_tree_dn = ou=Groups,dc=example,dc=org
group_objectclass = groupOfNames
```

> **Note**
>
> These schema attributes are extensible for compatibility with various schemas. For example, this entry maps to the person attribute in Active Directory:
>
> ```
> user_objectclass = person
> ```

8. A read-only implementation is recommended for LDAP integration. These permissions are applied to object types in the `/etc/keystone/domains/keystone.DOMAIN_NAME.conf` file:

```
[ldap]
user_allow_create = False
user_allow_update = False
user_allow_delete = False
```

```
group_allow_create = False
group_allow_update = False
group_allow_delete = False
```

9. Restart the OpenStack Identity service.

> **Warning**
>
> During service restart, authentication and authorization are unavailable.

Additional LDAP integration settings

Set these options in the `/etc/keystone/keystone.conf` file for a single LDAP server, or `/etc/keystone/domains/keystone.DOMAIN_NAME.conf` files for multiple back ends. Example configurations appear below each setting summary:

Filters

Use filters to control the scope of data presented through LDAP.

```
[ldap]
user_filter = (memberof=cn=openstack-users,ou=workgroups,dc=example,dc=org)
group_filter =
```

Identity attribute mapping

Mask account status values (include any additional attribute mappings) for compatibility with various directory services. Superfluous accounts are filtered with `user_filter`.

Setting attribute ignore to list of attributes stripped off on update.

For example, you can mask Active Directory account status attributes in the `keystone.conf` file:

```
[ldap]
user_id_attribute       = cn
user_name_attribute     = sn
user_mail_attribute     = mail
user_pass_attribute     = userPassword
user_enabled_attribute = userAccountControl
user_enabled_mask       = 2
user_enabled_invert     = false
user_enabled_default    = 51
user_default_project_id_attribute =
user_attribute_ignore = default_project_id,tenants
user_additional_attribute_mapping =
```

```
group_id_attribute     = cn
group_name_attribute   = ou
group_member_attribute = member
group_desc_attribute   = description
group_attribute_ignore =
group_additional_attribute_mapping =
```

Enabled emulation

An alternative method to determine if a user is enabled or not is by checking if that user is a member of the emulation group.

Use DN of the group entry to hold enabled user when using enabled emulation.

```
[ldap]
user_enabled_emulation = false
user_enabled_emulation_dn = false
```

When you have finished configuration, restart the OpenStack Identity service.

> **Warning**
>
> During service restart, authentication and authorization are unavailable.

Integrate assignment back end with LDAP

When you configure the OpenStack Identity service to use LDAP servers, you can split authentication and authorization using the *assignment* feature. Integrating the *assignment* back end with LDAP allows administrators to use projects (tenant), roles, domains, and role assignments in LDAP.

> **Note**
>
> Be aware of domain-specific back end limitations when configuring OpenStack Identity. The OpenStack Identity service does not support domain-specific assignment back ends. Using LDAP as an assignment back end is not recommended.

> **Important**
>
> For OpenStack Identity assignments to access LDAP servers, you must define the destination LDAP server in the `keystone.conf` file. For more information, see *Integrate Identity with LDAP*.

To integrate assignment back ends with LDAP

1. Enable the assignment driver. In the `[assignment]` section, set the `driver` configuration key to `keystone.assignment.backends.sql.Assignment`:

```
[assignment]
#driver = keystone.assignment.backends.sql.Assignment
driver = keystone.assignment.backends.ldap.Assignment
```

2. Create the organizational units (OU) in the LDAP directory, and define their corresponding location in the `keystone.conf` file:

```
[ldap]
role_tree_dn =
role_objectclass = inetOrgPerson

project_tree_dn = ou=Groups,dc=example,dc=org
project_objectclass = groupOfNames
```

> **Note**
>
> These schema attributes are extensible for compatibility with various schemas. For example, this entry maps to the groupOfNames attribute in Active Directory:
>
> ```
> project_objectclass = groupOfNames
> ```

3. A read-only implementation is recommended for LDAP integration. These permissions are applied to object types in the `keystone.conf` file:

```
[ldap]
role_allow_create = False
role_allow_update = False
role_allow_delete = False

project_allow_create = False
project_allow_update = False
project_allow_delete = False
```

4. Restart the OpenStack Identity service.

> **Warning**
>
> During service restart, authentication and authorization are unavailable.

Additional LDAP integration settings.

Set these options in the `/etc/keystone/keystone.conf` file for a single LDAP server, or `/etc/keystone/domains/keystone.DOMAIN_NAME.conf` files for multiple back ends.

Filters

Use filters to control the scope of data presented through LDAP.

```
[ldap]
project_filter = (member=cn=openstack-user,ou=workgroups,
dc=example,dc=org)
role_filter =
```

Warning

Filtering method

Assignment attribute mapping

Mask account status values (include any additional attribute mappings) for compatibility with various directory services. Superfluous accounts are filtered with user_filter.

Setting attribute ignore to list of attributes stripped off on update.

```
[ldap]
role_id_attribute = cn
role_name_attribute = ou
role_member_attribute = roleOccupant
role_additional_attribute_mapping =
role_attribute_ignore =

project_id_attribute = cn
project_name_attribute = ou
project_member_attribute = member
project_desc_attribute = description
project_enabled_attribute = enabled
project_domain_id_attribute = businessCategory
project_additional_attribute_mapping =
project_attribute_ignore =
```

Enabled emulation

An alternative method to determine if a project is enabled or not is to check if that project is a member of the emulation group.

Use DN of the group entry to hold enabled projects when using enabled emulation.

```
[ldap]
project_enabled_emulation = false
project_enabled_emulation_dn = false
```

Secure the OpenStack Identity service connection to an LDAP back end

The Identity service supports the use of TLS to encrypt LDAP traffic. Before configuring this, you must first verify where your certificate authority file is located. For more information, see the OpenStack Security Guide SSL introduction.

Once you verify the location of your certificate authority file:

To configure TLS encryption on LDAP traffic

1. Open the `/etc/keystone/keystone.conf` configuration file.
2. Find the `[ldap]` section.
3. In the `[ldap]` section, set the `use_tls` configuration key to `True`. Doing so will enable TLS.
4. Configure the Identity service to use your certificate authorities file. To do so, set the `tls_cacertfile` configuration key in the `ldap` section to the certificate authorities file's path.

> **Note**
>
> You can also set the `tls_cacertdir` (also in the `ldap` section) to the directory where all certificate authorities files are kept. If both `tls_cacertfile` and `tls_cacertdir` are set, then the latter will be ignored.

5. Specify what client certificate checks to perform on incoming TLS sessions from the LDAP server. To do so, set the `tls_req_cert` configuration key in the `[ldap]` section to `demand`, `allow`, or `never`:
   - `demand` - The LDAP server always receives certificate requests. The session terminates if no certificate is provided, or if the certificate provided cannot be verified against the existing certificate authorities file.
   - `allow` - The LDAP server always receives certificate requests. The session will proceed as normal even if a certificate is not provided. If a certificate is provided but it cannot be verified against the existing certificate authorities file, the certificate will be ignored and the session will proceed as normal.
   - `never` - A certificate will never be requested.

On distributions that include openstack-config, you can configure TLS encryption on LDAP traffic by running the following commands instead.

```
# openstack-config --set /etc/keystone/keystone.conf \
  ldap use_tls True
# openstack-config --set /etc/keystone/keystone.conf \
  ldap tls_cacertfile ``CA_FILE``
# openstack-config --set /etc/keystone/keystone.conf \
  ldap tls_req_cert ``CERT_BEHAVIOR``
```

Where:

- `CA_FILE` is the absolute path to the certificate authorities file that should be used to encrypt LDAP traffic.
- `CERT_BEHAVIOR` specifies what client certificate checks to perform on an incoming TLS session from the LDAP server (`demand`, `allow`, or `never`).

Keystone tokens

Tokens are used to authenticate and authorize your interactions with the various OpenStack APIs. Tokens come in many flavors, representing various authorization scopes and sources of identity. There are also several different "token providers", each with their own user experience, performance, and deployment characteristics.

Authorization scopes

Tokens can express your authorization in different scopes. You likely have different sets of roles, in different projects, and in different domains. While tokens always express your identity, they may only ever express one set of roles in one authorization scope at a time.

Each level of authorization scope is useful for certain types of operations in certain OpenStack services, and are not interchangeable.

Unscoped tokens

An unscoped token contains neither a service catalog, any roles, a project scope, nor a domain scope. Their primary use case is simply to prove your identity to keystone at a later time (usually to generate scoped tokens), without repeatedly presenting your original credentials.

The following conditions must be met to receive an unscoped token:

- You must not specify an authorization scope in your authentication request (for example, on the command line with arguments such as `--os-project-name` or

`--os-domain-id`),

- Your identity must not have a “default project” associated with it that you also have role assignments, and thus authorization, upon.

Project-scoped tokens

Project-scoped tokens are the bread and butter of OpenStack. They express your authorization to operate in a specific tenancy of the cloud and are useful to authenticate yourself when working with most other services.

They contain a service catalog, a set of roles, and details of the project upon which you have authorization.

Domain-scoped tokens

Domain-scoped tokens also have limited use cases in OpenStack. They express your authorization to operate a domain-level, above that of the user and projects contained therein (typically as a domain-level administrator). Depending on Keystone’s configuration, they are useful for working with a single domain in Keystone.

They contain a limited service catalog (only those services which do not explicitly require per-project endpoints), a set of roles, and details of the project upon which you have authorization.

They can also be used to work with domain-level concerns in other services, such as to configure domain-wide quotas that apply to all users or projects in a specific domain.

Token providers

The token type issued by keystone is configurable through the `etc/keystone.conf` file. Currently, there are four supported token types and they include `UUID`, `fernet`, `PKI`, and `PKIZ`.

UUID tokens

UUID was the first token type supported and is currently the default token provider. UUID tokens are 32 bytes in length and must be persisted in a back end. Clients must pass their UUID token to the Identity service in order to validate it.

Fernet tokens

The fernet token format was introduced in the OpenStack Kilo release. Unlike the other token types mentioned in this document, fernet tokens do not need to be persisted in a back end. `AES256` encryption is used to protect the information stored in the token and integrity is verified with a `SHA256 HMAC` signature. Only the Identity service should have access to the keys used to encrypt and decrypt fernet tokens. Like UUID tokens, fernet

tokens must be passed back to the Identity service in order to validate them. For more information on the fernet token type, see the *Fernet - Frequently Asked Questions*.

PKI and PKIZ tokens

PKI tokens are signed documents that contain the authentication context, as well as the service catalog. Depending on the size of the OpenStack deployment, these tokens can be very long. The Identity service uses public/private key pairs and certificates in order to create and validate PKI tokens.

The same concepts from PKI tokens apply to PKIZ tokens. The only difference between the two is PKIZ tokens are compressed to help mitigate the size issues of PKI. For more information on the certificate setup for PKI and PKIZ tokens, see the *Certificates for PKI*.

Configure Identity service for token binding

Token binding embeds information from an external authentication mechanism, such as a Kerberos server or X.509 certificate, inside a token. By using token binding, a client can enforce the use of a specified external authentication mechanism with the token. This additional security mechanism ensures that if a token is stolen, for example, it is not usable without external authentication.

You configure the authentication types for a token binding in the `keystone.conf` file:

```
[token]
bind = kerberos
```

or

```
[token]
bind = x509
```

Currently `kerberos` and `x509` are supported.

To enforce checking of token binding, set the `enforce_token_bind` option to one of these modes:

- **`disabled`**

  Disables token bind checking.

- **`permissive`**

  Enables bind checking. If a token is bound to an unknown authentication

mechanism, the server ignores it. The default is this mode.

- **strict**

  Enables bind checking. If a token is bound to an unknown authentication mechanism, the server rejects it.

- **required**

  Enables bind checking. Requires use of at least authentication mechanism for tokens.

- **kerberos**

  Enables bind checking. Requires use of kerberos as the authentication mechanism for tokens:

  ```
  [token]
  enforce_token_bind = kerberos
  ```

- **x509**

  Enables bind checking. Requires use of X.509 as the authentication mechanism for tokens:

  ```
  [token]
  enforce_token_bind = x509
  ```

Fernet - Frequently Asked Questions

The following questions have been asked periodically since the initial release of the fernet token format in Kilo.

What are the different types of keys?

A key repository is required by keystone in order to create fernet tokens. These keys are used to encrypt and decrypt the information that makes up the payload of the token. Each key in the repository can have one of three states. The state of the key determines how keystone uses a key with fernet tokens. The different types are as follows:

Primary key:
There is only ever one primary key in a key repository. The primary key is allowed to encrypt and decrypt tokens. This key is always named as the highest index in the repository.

Secondary key:

A secondary key was at one point a primary key, but has been demoted in place of another primary key. It is only allowed to decrypt tokens. Since it was the primary at some point in time, its existence in the key repository is justified. Keystone needs to be able to decrypt tokens that were created with old primary keys.

Staged key:

The staged key is a special key that shares some similarities with secondary keys. There can only ever be one staged key in a repository and it must exist. Just like secondary keys, staged keys have the ability to decrypt tokens. Unlike secondary keys, staged keys have never been a primary key. In fact, they are opposites since the staged key will always be the next primary key. This helps clarify the name because they are the next key staged to be the primary key. This key is always named as `0` in the key repository.

So, how does a staged key help me and why do I care about it?

The fernet keys have a natural lifecycle. Each key starts as a staged key, is promoted to be the primary key, and then demoted to be a secondary key. New tokens can only be encrypted with a primary key. Secondary and staged keys are never used to encrypt token. The staged key is a special key given the order of events and the attributes of each type of key. The staged key is the only key in the repository that has not had a chance to encrypt any tokens yet, but it is still allowed to decrypt tokens. As an operator, this gives you the chance to perform a key rotation on one keystone node, and distribute the new key set over a span of time. This does not require the distribution to take place in an ultra short period of time. Tokens encrypted with a primary key can be decrypted, and validated, on other nodes where that key is still staged.

Where do I put my key repository?

The key repository is specified using the `key_repository` option in the keystone configuration file. The keystone process should be able to read and write to this location but it should be kept secret otherwise. Currently, keystone only supports file-backed key repositories.

```
[fernet_tokens]
key_repository = /etc/keystone/fernet-keys/
```

What is the recommended way to rotate and distribute keys?

The **keystone-manage** command line utility includes a key rotation mechanism. This mechanism will initialize and rotate keys but does not make an effort to distribute keys across keystone nodes. The distribution of keys across a keystone deployment is best handled through configuration management tooling. Use **keystone-manage fernet_rotate** to rotate the key repository.

Do fernet tokens still expire?

Yes, fernet tokens can expire just like any other keystone token formats.

Why should I choose fernet tokens over UUID tokens?

Even though fernet tokens operate very similarly to UUID tokens, they do not require persistence. The keystone token database no longer suffers bloat as a side effect of authentication. Pruning expired tokens from the token database is no longer required when using fernet tokens. Because fernet tokens do not require persistence, they do not have to be replicated. As long as each keystone node shares the same key repository, fernet tokens can be created and validated instantly across nodes.

Why should I choose fernet tokens over PKI or PKIZ tokens?

The arguments for using fernet over PKI and PKIZ remain the same as UUID, in addition to the fact that fernet tokens are much smaller than PKI and PKIZ tokens. PKI and PKIZ tokens still require persistent storage and can sometimes cause issues due to their size. This issue is mitigated when switching to fernet because fernet tokens are kept under a 250 byte limit. PKI and PKIZ tokens typically exceed 1600 bytes in length. The length of a PKI or PKIZ token is dependent on the size of the deployment. Bigger service catalogs will result in longer token lengths. This pattern does not exist with fernet tokens because the contents of the encrypted payload is kept to a minimum.

Should I rotate and distribute keys from the same keystone node every rotation?

No, but the relationship between rotation and distribution should be lock-step. Once you rotate keys on one keystone node, the key repository from that node should be distributed to the rest of the cluster. Once you confirm that each node has the same key repository state, you could rotate and distribute from any other node in the cluster.

If the rotation and distribution are not lock-step, a single keystone node in the deployment will create tokens with a primary key that no other node has as a staged key. This will cause tokens generated from one keystone node to fail validation on other keystone nodes.

How do I add new keystone nodes to a deployment?

The keys used to create fernet tokens should be treated like super secret configuration files, similar to an SSL secret key. Before a node is allowed to join an existing cluster, issuing and validating tokens, it should have the same key repository as the rest of the nodes in the cluster.

How should I approach key distribution?

Remember that key distribution is only required in multi-node keystone deployments. If you only have one keystone node serving requests in your deployment, key distribution is unnecessary.

Key distribution is a problem best approached from the deployment's current configuration management system. Since not all deployments use the same configuration management systems, it makes sense to explore options around what is already available for managing keys, while keeping the secrecy of the keys in mind. Many configuration management tools can leverage something like `rsync` to manage key distribution.

Key rotation is a single operation that promotes the current staged key to primary, creates a new staged key, and prunes old secondary keys. It is easiest to do this on a single node and verify the rotation took place properly before distributing the key repository to the rest of the cluster. The concept behind the staged key breaks the expectation that key rotation and key distribution have to be done in a single step. With the staged key, we have time to inspect the new key repository before syncing state with the rest of the cluster. Key distribution should be an operation that can run in succession until it succeeds. The following might help illustrate the isolation between key rotation and key distribution.

1. Ensure all keystone nodes in the deployment have the same key repository.
2. Pick a keystone node in the cluster to rotate from.

3. Rotate keys.
    1. Was it successful?
        1. If no, investigate issues with the particular keystone node you rotated keys on. Fernet keys are small and the operation for rotation is trivial. There should not be much room for error in key rotation. It is possible that the user does not have the ability to write new keys to the key repository. Log output from `keystone-manage fernet_rotate` should give more information into specific failures.
        2. If yes, you should see a new staged key. The old staged key should be the new primary. Depending on the `max_active_keys` limit you might have secondary keys that were pruned. At this point, the node that you rotated on will be creating fernet tokens with a primary key that all other nodes should have as the staged key. This is why we checked the state of all key repositories in Step one. All other nodes in the cluster should be able to decrypt tokens created with the new primary key. At this point, we are ready to distribute the new key set.
4. Distribute the new key repository.
    1. Was it successful?
        1. If yes, you should be able to confirm that all nodes in the cluster have the same key repository that was introduced in Step 3. All nodes in the cluster will be creating tokens with the primary key that was promoted in Step 3. No further action is required until the next schedule key rotation.
        2. If no, try distributing again. Remember that we already rotated the repository and performing another rotation at this point will result in tokens that cannot be validated across certain hosts. Specifically, the hosts that did not get the latest key set. You should be able to distribe keys until it is successful. If certain nodes have issues syncing, it could be permission or network issues and those should be resolved before subsequent rotations.

How long should I keep my keys around?

The fernet tokens that keystone creates are only secure as the keys creating them. With staged keys the penalty of key rotation is low, allowing you to err on the side of security and rotate weekly, daily, or even hourly. Ultimately, this should be less time than it takes an attacker to break a `AES256` key and a `SHA256 HMAC`.

Is a fernet token still a bearer token?

Yes, and they follow exactly the same validation path as UUID tokens, with the exception of being written to, and read from, a back end. If someone compromises your fernet token,

they have the power to do all the operations you are allowed to do.

What if I need to revoke all my tokens?

To invalidate every token issued from keystone and start fresh, remove the current key repository, create a new key set, and redistribute it to all nodes in the cluster. This will render every token issued from keystone as invalid regardless if the token has actually expired. When a client goes to re-authenticate, the new token will have been created with a new fernet key.

What can an attacker do if they compromise a fernet key in my deployment?

If any key used in the key repository is compromised, an attacker will be able to build their own tokens. If they know the ID of an administrator on a project, they could generate administrator tokens for the project. They will be able to generate their own tokens until the compromised key has been removed from from the repository.

I rotated keys and now tokens are invalidating early, what did I do?

Using fernet tokens requires some awareness around token expiration and the key lifecycle. You do not want to rotate so often that secondary keys are removed that might still be needed to decrypt unexpired tokens. If this happens, you will not be able to decrypt the token because the key the was used to encrypt it is now gone. Only remove keys that you know are not being used to encrypt or decrypt tokens.

For example, your token is valid for 24 hours and we want to rotate keys every six hours. We will need to make sure tokens that were created at 08:00 AM on Monday are still valid at 07:00 AM on Tuesday, assuming they were not prematurely revoked. To accomplish this, we will want to make sure we set `max_active_keys=6` in our keystone configuration file. This will allow us to hold all keys that might still be required to validate a previous token, but keeps the key repository limited to only the keys that are needed.

The number of `max_active_keys` for a deployment can be determined by dividing the token lifetime, in hours, by the frequency of rotation in hours and adding two. Better illustrated as:

```
token_expiration = 24
rotation_frequency = 6
max_active_keys = (token_expiration / rotation_frequency) + 2
```

The reason for adding two additional keys to the count is to include the staged key and a buffer key. This can be shown based on the previous example. We initially setup the key repository at 6:00 AM on Monday, and the initial state looks like:

```
$ ls -la /etc/keystone/fernet-keys/
drwx------ 2 keystone keystone 4096 .
drwxr-xr-x 3 keystone keystone 4096 ..
-rw------- 1 keystone keystone   44 0    (staged key)
-rw------- 1 keystone keystone   44 1    (primary key)
```

All tokens created after 6:00 AM are encrypted with key `1`. At 12:00 PM we will rotate keys again, resulting in,

```
$ ls -la /etc/keystone/fernet-keys/
drwx------ 2 keystone keystone 4096 .
drwxr-xr-x 3 keystone keystone 4096 ..
-rw------- 1 keystone keystone   44 0    (staged key)
-rw------- 1 keystone keystone   44 1    (secondary key)
-rw------- 1 keystone keystone   44 2    (primary key)
```

We are still able to validate tokens created between 6:00 - 11:59 AM because the `1` key still exists as a secondary key. All tokens issued after 12:00 PM will be encrypted with key `2`. At 6:00 PM we do our next rotation, resulting in:

```
$ ls -la /etc/keystone/fernet-keys/
drwx------ 2 keystone keystone 4096 .
drwxr-xr-x 3 keystone keystone 4096 ..
-rw------- 1 keystone keystone   44 0    (staged key)
-rw------- 1 keystone keystone   44 1    (secondary key)
-rw------- 1 keystone keystone   44 2    (secondary key)
-rw------- 1 keystone keystone   44 3    (primary key)
```

It is still possible to validate tokens issued from 6:00 AM - 5:59 PM because keys `1` and `2` exist as secondary keys. Every token issued until 11:59 PM will be encrypted with key `3`, and at 12:00 AM we do our next rotation:

```
$ ls -la /etc/keystone/fernet-keys/
drwx------ 2 keystone keystone 4096 .
drwxr-xr-x 3 keystone keystone 4096 ..
-rw------- 1 keystone keystone   44 0    (staged key)
-rw------- 1 keystone keystone   44 1    (secondary key)
-rw------- 1 keystone keystone   44 2    (secondary key)
-rw------- 1 keystone keystone   44 3    (secondary key)
-rw------- 1 keystone keystone   44 4    (primary key)
```

Just like before, we can still validate tokens issued from 6:00 AM the previous day until

5:59 AM today because keys `1` - `4` are present. At 6:00 AM, tokens issued from the previous day will start to expire and we do our next scheduled rotation:

```
$ ls -la /etc/keystone/fernet-keys/
drwx------ 2 keystone keystone 4096 .
drwxr-xr-x 3 keystone keystone 4096 ..
-rw------- 1 keystone keystone   44 0    (staged key)
-rw------- 1 keystone keystone   44 1    (secondary key)
-rw------- 1 keystone keystone   44 2    (secondary key)
-rw------- 1 keystone keystone   44 3    (secondary key)
-rw------- 1 keystone keystone   44 4    (secondary key)
-rw------- 1 keystone keystone   44 5    (primary key)
```

Tokens will naturally expire after 6:00 AM, but we will not be able to remove key `1` until the next rotation because it encrypted all tokens from 6:00 AM to 12:00 PM the day before. Once we do our next rotation, which is at 12:00 PM, the `1` key will be pruned from the repository:

```
$ ls -la /etc/keystone/fernet-keys/
drwx------ 2 keystone keystone 4096 .
drwxr-xr-x 3 keystone keystone 4096 ..
-rw------- 1 keystone keystone   44 0    (staged key)
-rw------- 1 keystone keystone   44 2    (secondary key)
-rw------- 1 keystone keystone   44 3    (secondary key)
-rw------- 1 keystone keystone   44 4    (secondary key)
-rw------- 1 keystone keystone   44 5    (secondary key)
-rw------- 1 keystone keystone   44 6    (primary key)
```

If keystone were to receive a token that was created between 6:00 AM and 12:00 PM the day before, encrypted with the `1` key, it would not be valid because it was already expired. This makes it possible for us to remove the `1` key from the repository without negative validation side-effects.

Use trusts

OpenStack Identity manages authentication and authorization. A trust is an OpenStack Identity extension that enables delegation and, optionally, impersonation through `keystone`. A trust extension defines a relationship between:

Trustor
The user delegating a limited set of their own rights to another user.

Trustee

The user trust is being delegated to, for a limited time.

The trust can eventually allow the trustee to impersonate the trustor. For security reasons, some safeties are added. For example, if a trustor loses a given role, any trusts the user issued with that role, and the related tokens, are automatically revoked.

The delegation parameters are:

User ID
The user IDs for the trustor and trustee.

Privileges

The delegated privileges are a combination of a tenant ID and a number of roles that must be a subset of the roles assigned to the trustor.

If you omit all privileges, nothing is delegated. You cannot delegate everything.

Delegation depth

Defines whether or not the delegation is recursive. If it is recursive, defines the delegation chain length.

Specify one of the following values:

- `0`. The delegate cannot delegate these permissions further.
- `1`. The delegate can delegate the permissions to any set of delegates but the latter cannot delegate further.
- `inf`. The delegation is infinitely recursive.

Endpoints

A list of endpoints associated with the delegation.

This parameter further restricts the delegation to the specified endpoints only. If you omit the endpoints, the delegation is useless. A special value of `all_endpoints` allows the trust to be used by all endpoints associated with the delegated tenant.

Duration
(Optional) Comprised of the start time and end time for the trust.

Caching layer

OpenStack Identity supports a caching layer that is above the configurable subsystems (for example, token, assignment). OpenStack Identity uses the dogpile.cache library which

allows flexible cache back ends. The majority of the caching configuration options are set in the `[cache]` section of the `keystone.conf` file. However, each section that has the capability to be cached usually has a caching boolean value that toggles caching.

So to enable only the token back end caching, set the values as follows:

```
[cache]
enabled=true

[assignment]
caching=false

[token]
caching=true
```

> **Note**
>
> Since the Juno release, the default setting is enabled for subsystem caching, but the global toggle is disabled. As a result, no caching in available unless the global toggle for `[cache]` is enabled by setting the value to `true`.

Caching for tokens and tokens validation

The token system has a separate `cache_time` configuration option, that can be set to a value above or below the global `expiration_time` default, allowing for different caching behavior from the other systems in OpenStack Identity. This option is set in the `[token]` section of the configuration file. Fernet tokens do not need to be persisted in a back end and therefore must not be cached.

The token revocation list cache time is handled by the configuration option `revocation_cache_time` in the `[token]` section. The revocation list is refreshed whenever a token is revoked. It typically sees significantly more requests than specific token retrievals or token validation calls.

Here is a list of actions that are affected by the cached time: getting a new token, revoking tokens, validating tokens, checking v2 tokens, and checking v3 tokens.

The delete token API calls invalidate the cache for the tokens being acted upon, as well as invalidating the cache for the revoked token list and the validate/check token calls.

Token caching is configurable independently of the `revocation_list` caching. Lifted expiration checks from the token drivers to the token manager. This ensures that cached tokens will still raise a `TokenNotFound` flag when expired.

For cache consistency, all token IDs are transformed into the short token hash at the

provider and token driver level. Some methods have access to the full ID (PKI Tokens), and some methods do not. Cache invalidation is inconsistent without token ID normalization.

Caching around assignment CRUD

The assignment system has a separate `cache_time` configuration option, that can be set to a value above or below the global `expiration_time` default, allowing for different caching behavior from the other systems in Identity service. This option is set in the `[assignment]` section of the configuration file.

Currently `assignment` has caching for `project`, `domain`, and `role` specific requests (primarily around the CRUD actions). Caching is currently not implemented on grants. The `list` methods are not subject to caching.

Here is a list of actions that are affected by the assignment: assign domain API, assign project API, and assign role API.

The create, update, and delete actions for domains, projects and roles will perform proper invalidations of the cached methods listed above.

> **Note**
>
> If a read-only `assignment` back end is in use, the cache will not immediately reflect changes on the back end. Any given change may take up to the `cache_time` (if set in the `[assignment]` section of the configuration file) or the global `expiration_time` (set in the `[cache]` section of the configuration file) before it is reflected. If this type of delay (when using a read-only `assignment` back end) is an issue, it is recommended that caching be disabled on `assignment`. To disable caching specifically on `assignment`, in the `[assignment]` section of the configuration set `caching` to `False`.

For more information about the different back ends (and configuration options), see:

- dogpile.cache.backends.memory
- dogpile.cache.backends.memcached

> **Note**
>
> The memory back end is not suitable for use in a production environment.

- dogpile.cache.backends.redis
- dogpile.cache.backends.file
- `keystone.common.cache.backends.mongo`

Configure the Memcached back end example

The following example shows how to configure the memcached back end:

```
[cache]

enabled = true
backend = dogpile.cache.memcached
backend_argument = url:127.0.0.1:11211
```

You need to specify the URL to reach the `memcached` instance with the `backend_argument` parameter.

Start the Identity service

In Kilo and newer releases, the Identity service should use the Apache HTTP Server with the `mod_wsgi` module instead of the eventlet library. Using the proper WSGI configuration, the Apache HTTP Server binds to ports 5000 and 35357 rather than the keystone process.

> **Important**
>
> Eventlet support will be removed in OpenStack Newton.

For more information, see http://docs.openstack.org/developer/keystone/apache-httpd.html and https://git.openstack.org/cgit/openstack/keystone/tree/httpd.

Example usage and Identity features

The `openstack` CLI is used to interact with the Identity service. It is set up to expect commands in the general form of `openstack command argument`, followed by flag-like keyword arguments to provide additional (often optional) information. For example, the **user list** and **project create** commands can be invoked as follows:

```
# Using token auth env variables
export OS_SERVICE_ENDPOINT=http://127.0.0.1:5000/v2.0/
export OS_SERVICE_TOKEN=secrete_token
openstack user list
openstack project create demo

# Using token auth flags
openstack --os-token secrete --os-endpoint http://127.0.0.1:5000/v2.0/ user list
```

```
openstack --os-token secrete --os-endpoint http://127.0.0.1:5000/v2.0/ project
create demo

# Using user + password + project_name env variables
export OS_USERNAME=admin
export OS_PASSWORD=secrete
export OS_PROJECT_NAME=admin
openstack user list
openstack project create demo

# Using user + password + project-name flags
openstack --os-username admin --os-password secrete --os-project-name admin user
list
openstack --os-username admin --os-password secrete --os-project-name admin
project create demo
```

Logging

You configure logging externally to the rest of Identity. The name of the file specifying the logging configuration is set using the `log_config` option in the `[DEFAULT]` section of the `keystone.conf` file. To route logging through syslog, set `use_syslog=true` in the `[DEFAULT]` section.

A sample logging configuration file is available with the project in `etc/logging.conf.sample`. Like other OpenStack projects, Identity uses the Python logging module, which provides extensive configuration options that let you define the output levels and formats.

User CRUD

Identity provides a user CRUD (Create, Read, Update, and Delete) filter that Administrators can add to the `public_api` pipeline. The user CRUD filter enables users to use a HTTP PATCH to change their own password. To enable this extension you should define a `user_crud_extension` filter, insert it after the `*_body` middleware and before the `public_service` application in the `public_api` WSGI pipeline in `keystone-paste.ini`. For example:

```
[filter:user_crud_extension]
paste.filter_factory = keystone.contrib.user_crud:CrudExtension.factory

[pipeline:public_api]
pipeline = sizelimit url_normalize request_id build_auth_context token_auth
admin_token_auth json_body ec2_extension user_crud_extension public_service
```

Each user can then change their own password with a HTTP PATCH.

```
$ curl -X PATCH http://localhost:5000/v2.0/OS-KSCRUD/users/USERID -H "Content-
type: application/json"  \
  -H "X_Auth_Token: AUTHTOKENID" -d '{"user": {"password": "ABCD",
"original_password": "DCBA"}}'
```

In addition to changing their password, all current tokens for the user are invalidated.

> **Note**
>
> Only use a KVS back end for tokens when testing.

Authentication middleware with user name and password

You can also configure Identity authentication middleware using the `admin_user` and `admin_password` options.

> **Note**
>
> The `admin_token` option is deprecated and no longer used for configuring auth_token middleware.

For services that have a separate paste-deploy `.ini` file, you can configure the authentication middleware in the `[keystone_authtoken]` section of the main configuration file, such as `nova.conf`. In Compute, for example, you can remove the middleware parameters from `api-paste.ini`, as follows:

```
[filter:authtoken]
paste.filter_factory = keystonemiddleware.auth_token:filter_factory
```

And set the following values in `nova.conf` as follows:

```
[DEFAULT]
...
auth_strategy=keystone

[keystone_authtoken]
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
```

```
admin_user = admin
admin_password = SuperSekretPassword
admin_tenant_name = service
```

> **Note**
>
> The middleware parameters in the paste config take priority. You must remove them to use the values in the `[keystone_authtoken]` section.

> **Note**
>
> Comment out any `auth_host`, `auth_port`, and `auth_protocol` options because the `identity_uri` option replaces them.

This sample paste config filter makes use of the `admin_user` and `admin_password` options:

```
[filter:authtoken]
paste.filter_factory = keystonemiddleware.auth_token:filter_factory
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
auth_token = 012345SECRET99TOKEN012345
admin_user = admin
admin_password = keystone123
```

> **Note**
>
> Using this option requires an admin tenant/role relationship. The admin user is granted access to the admin role on the admin tenant.

> **Note**
>
> Comment out any `auth_host`, `auth_port`, and `auth_protocol` options because the `identity_uri` option replaces them.

Identity API protection with role-based access control (RBAC)

Like most OpenStack projects, Identity supports the protection of its APIs by defining policy rules based on an RBAC approach. Identity stores a reference to a policy JSON file in the main Identity configuration file, `keystone.conf`. Typically this file is named `policy.json`, and contains the rules for which roles have access to certain actions in defined services.

Each Identity API v3 call has a line in the policy file that dictates which level of governance of access applies.

```
API_NAME: RULE_STATEMENT or MATCH_STATEMENT
```

Where:

`RULE_STATEMENT` can contain `RULE_STATEMENT` or `MATCH_STATEMENT`.

`MATCH_STATEMENT` is a set of identifiers that must match between the token provided by the caller of the API and the parameters or target entities of the API call in question. For example:

```
"identity:create_user": "role:admin and domain_id:%(user.domain_id)s"
```

Indicates that to create a user, you must have the admin role in your token. The `domain_id` in your token must match the `domain_id` in the user object that you are trying to create, which implies this must be a domain-scoped token. In other words, you must have the admin role on the domain in which you are creating the user, and the token that you use must be scoped to that domain.

Each component of a match statement uses this format:

```
ATTRIB_FROM_TOKEN:CONSTANT or ATTRIB_RELATED_TO_API_CALL
```

The Identity service expects these attributes:

Attributes from token:

- `user_id`
- `domain_id`
- `project_id`

The `project_id` attribute requirement depends on the scope, and the list of roles you have within that scope.

Attributes related to API call:

- `user.domain_id`
- Any parameters passed into the API call
- Any filters specified in the query string

You reference attributes of objects passed with an object.attribute syntax (such as, `user.domain_id`). The target objects of an API are also available using a target.object.attribute syntax. For instance:

```
"identity:delete_user": "role:admin and domain_id:%(target.user.domain_id)s"
```

would ensure that Identity only deletes the user object in the same domain as the provided token.

Every target object has an `id` and a `name` available as `target.OBJECT.id` and `target.OBJECT.name`. Identity retrieves other attributes from the database, and the attributes vary between object types. The Identity service filters out some database fields, such as user passwords.

List of object attributes:

```
role:
     target.role.id
     target.role.name

user:
     target.user.default_project_id
     target.user.description
     target.user.domain_id
     target.user.enabled
     target.user.id
     target.user.name

group:
     target.group.description
     target.group.domain_id
     target.group.id
     target.group.name

domain:
     target.domain.enabled
     target.domain.id
     target.domain.name

project:
     target.project.description
     target.project.domain_id
```

```
target.project.enabled
target.project.id
target.project.name
```

The default `policy.json` file supplied provides a somewhat basic example of API protection, and does not assume any particular use of domains. Refer to `policy.v3cloudsample.json` as an example of multi-domain configuration installations where a cloud provider wants to delegate administration of the contents of a domain to a particular `admin domain`. This example policy file also shows the use of an `admin_domain` to allow a cloud provider to enable administrators to have wider access across the APIs.

A clean installation could start with the standard policy file, to allow creation of the `admin_domain` with the first users within it. You could then obtain the `domain_id` of the admin domain, paste the ID into a modified version of `policy.v3cloudsample.json`, and then enable it as the main `policy file`.

Troubleshoot the Identity service

To troubleshoot the Identity service, review the logs in the `/var/log/keystone/keystone.log` file.

Use the `/etc/keystone/logging.conf` file to configure the location of log files.

> **Note**
>
> The `insecure_debug` flag is unique to the Identity service. If you enable `insecure_debug`, error messages from the API change to return security-sensitive information. For example, the error message on failed authentication includes information on why your authentication failed.

The logs show the components that have come in to the WSGI request, and ideally show an error that explains why an authorization request failed. If you do not see the request in the logs, run keystone with the *--debug* parameter. Pass the *--debug* parameter before the command parameters.

Debug PKI middleware

Problem

If you receive an `Invalid OpenStack Identity Credentials` message when you accessing and reaching an OpenStack service, it might be caused by the changeover from UUID tokens to PKI tokens in the Grizzly release.

The PKI-based token validation scheme relies on certificates from Identity that are fetched through HTTP and stored in a local directory. The location for this directory is specified by the `signing_dir` configuration option.

Solution

In your services configuration file, look for a section like this:

```
[keystone_authtoken]
signing_dir = /var/cache/glance/api
auth_uri = http://controller:5000/v2.0
identity_uri = http://controller:35357
admin_tenant_name = service
admin_user = glance
```

The first thing to check is that the `signing_dir` does, in fact, exist. If it does, check for certificate files:

```
$ ls -la /var/cache/glance/api/

total 24
drwx------. 2 ayoung root 4096 Jul 22 10:58 .
drwxr-xr-x. 4 root root 4096 Nov 7 2012 ..
-rw-r-----. 1 ayoung ayoung 1424 Jul 22 10:58 cacert.pem
-rw-r-----. 1 ayoung ayoung 15 Jul 22 10:58 revoked.pem
-rw-r-----. 1 ayoung ayoung 4518 Jul 22 10:58 signing_cert.pem
```

This directory contains two certificates and the token revocation list. If these files are not present, your service cannot fetch them from Identity. To troubleshoot, try to talk to Identity to make sure it correctly serves files, as follows:

```
$ curl http://localhost:35357/v2.0/certificates/signing
```

This command fetches the signing certificate:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, ST=Unset, L=Unset, O=Unset, CN=www.example.com
        Validity
            Not Before: Jul 22 14:57:31 2013 GMT
            Not After : Jul 20 14:57:31 2023 GMT
        Subject: C=US, ST=Unset, O=Unset, CN=www.example.com
```

Note the expiration dates of the certificate:

```
Not Before: Jul 22 14:57:31 2013 GMT
Not After : Jul 20 14:57:31 2023 GMT
```

The token revocation list is updated once a minute, but the certificates are not. One possible problem is that the certificates are the wrong files or garbage. You can remove these files and run another command against your server; they are fetched on demand.

The Identity service log should show the access of the certificate files. You might have to turn up your logging levels. Set `debug = True` in your Identity configuration file and restart the Identity server.

```
(keystone.common.wsgi): 2013-07-24 12:18:11,461 DEBUG wsgi __call__
arg_dict: {}
(access): 2013-07-24 12:18:11,462 INFO core __call__ 127.0.0.1 - -
[24/Jul/2013:16:18:11 +0000]
"GET http://localhost:35357/v2.0/certificates/signing HTTP/1.0" 200 4518
```

If the files do not appear in your directory after this, it is likely one of the following issues:

- Your service is configured incorrectly and cannot talk to Identity. Check the `auth_port` and `auth_host` values and make sure that you can talk to that service through cURL, as shown previously.
- Your signing directory is not writable. Use the `chmod` command to change its permissions so that the service (POSIX) user can write to it. Verify the change through `su` and `touch` commands.
- The SELinux policy is denying access to the directory.

SELinux troubles often occur when you use Fedora or RHEL-based packages and you choose configuration options that do not match the standard policy. Run the `setenforce permissive` command. If that makes a difference, you should relabel the directory. If you are using a sub-directory of the `/var/cache/` directory, run the following command:

```
# restorecon /var/cache/
```

If you are not using a `/var/cache` sub-directory, you should. Modify the `signing_dir` configuration option for your service and restart.

Set back to `setenforce enforcing` to confirm that your changes solve the problem.

If your certificates are fetched on demand, the PKI validation is working properly. Most likely, the token from Identity is not valid for the operation you are attempting to perform, and your user needs a different role for the operation.

Debug signing key file errors

Problem

If an error occurs when the signing key file opens, it is possible that the person who ran the **keystone-manage pki_setup** command to generate certificates and keys did not use the correct user.

Solution

When you run the **keystone-manage pki_setup** command, Identity generates a set of certificates and keys in `/etc/keystone/ssl*`, which is owned by `root:root`. This can present a problem when you run the Identity daemon under the keystone user account (nologin) when you try to run PKI. Unless you run the **chown** command against the files `keystone:keystone`, or run the **keystone-manage pki_setup** command with the *--keystone-user* and *--keystone-group* parameters, you will get an error. For example:

```
2012-07-31 11:10:53 ERROR [keystone.common.cms] Error opening signing key file
/etc/keystone/ssl/private/signing_key.pem
140380567730016:error:0200100D:system library:fopen:Permission
denied:bss_file.c:398:fopen('/etc/keystone/ssl/private/signing_key.pem','r')
140380567730016:error:20074002:BIO routines:FILE_CTRL:system lib:bss_file.c:400:
unable to load signing key file
```

Flush expired tokens from the token database table

Problem

As you generate tokens, the token database table on the Identity server grows.

Solution

To clear the token table, an administrative user must run the **keystone-manage token_flush** command to flush the tokens. When you flush tokens, expired tokens are deleted and traceability is eliminated.

Use `cron` to schedule this command to run frequently based on your workload. For large workloads, running it every minute is recommended.

Dashboard

The OpenStack Dashboard is a web-based interface that allows you to manage OpenStack

resources and services. The Dashboard allows you to interact with the OpenStack Compute cloud controller using the OpenStack APIs. For more information about installing and configuring the Dashboard, see the OpenStack Installation Guide for your operating system.

- Customize the dashboard
    - Logo and site colors
    - HTML title
    - Logo link
    - Help URL
- Set up session storage for the Dashboard
    - Local memory cache
    - Cached database
    - Cookies
- Create and manage images
    - Create images
    - Update images
    - Delete images
- Create and manage roles
    - Create a role
    - Edit a role
    - Delete a role
- Manage instances
    - Create instance snapshots
    - Control the state of an instance
    - Track usage
- Manage flavors
    - Create flavors
    - Update flavors
    - Update Metadata
    - Delete flavors
- Manage volumes and volume types
    - Create a volume type
    - Create an encrypted volume type
    - Delete volume types
    - Delete volumes
- Manage shares and share types
    - Create a share type
    - Update share type
    - Delete share types

    - Delete shares
    - Delete share server
    - Delete share networks
- View and manage quotas
    - View default project quotas
    - Update project quotas
- View cloud resources
    - View services information
    - View cloud usage statistics
- Create and manage host aggregates
    - To create a host aggregate
    - To manage host aggregates
- Launch and manage stacks using the Dashboard

- To deploy the dashboard, see the OpenStack dashboard documentation.
- To launch instances with the dashboard as an end user, see the Launch and manage instances. in the OpenStack End User Guide.
- To create and manage ports, see the Create and manage networks section of the OpenStack End User Guide.

Customize the dashboard

Once you have the dashboard installed you can customize the way it looks and feels to suit your own needs.

> **Note**
>
> The OpenStack dashboard by default on Ubuntu installs the `openstack-dashboard-ubuntu-theme` package.
>
> If you do not want to use this theme you can remove it and its dependencies using the following command:
>
> ```
> # apt-get remove --auto-remove openstack-dashboard-ubuntu-theme
> ```
>
> **Note**
>
> This guide focuses on the `local_settings.py` file.

The following can easily be customized:

- Logo
- Site colors
- HTML title
- Logo link
- Help URL

Logo and site colors

1. Create two PNG logo files with transparent backgrounds using the following sizes:
   - Login screen: 365 x 50
   - Logged in banner: 216 x 35
2. Upload your new images to `/usr/share/openstack-dashboard/openstack_dashboard/static/dashboard/img/`.
3. Create a CSS style sheet in `/usr/share/openstack-dashboard/openstack_dashboard/static/dashboard/scss/`.
4. Change the colors and image file names as appropriate, though the relative directory paths should be the same. The following example file shows you how to customize your CSS file:

```
/*
* 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;
}
```

5. Open the following HTML template in an editor of your choice:

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

6. Add a line to include your newly created style sheet. For example, `custom.css` file:

   ```
   <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' />
   ```

7. Restart the Apache service.

8. To view your changes reload your dashboard. If necessary go back and modify your CSS file as appropriate.

HTML title

1. Set the HTML title, which appears at the top of the browser window, by adding the following line to `local_settings.py`:

   ```
   SITE_BRANDING = "Example, Inc. Cloud"
   ```

2. Restart Apache for this change to take effect.

Logo link

1. The logo also acts as a hyperlink. The default behavior is to redirect to `horizon:user_home`. To change this, add the following attribute to

`local_settings.py`:

```
SITE_BRANDING_LINK = "http://example.com"
```

2. Restart Apache for this change to take effect.

Help URL

1. By default the help URL points to http://docs.openstack.org. Change this by editing the following attribute to the URL of your choice in `local_settings.py`:

```
HORIZON_CONFIG["help_url"] = "http://openstack.mycompany.org"
```

2. Restart Apache for this change to take effect.

Set up session storage for the Dashboard

The Dashboard uses Django sessions framework to handle user session data. However, you can use any available session back end. You customize the session back end through the `SESSION_ENGINE` setting in your `local_settings.py` file.

After architecting and implementing the core OpenStack services and other required services, combined with the Dashboard service steps below, users and administrators can use the OpenStack dashboard. Refer to the OpenStack Dashboard chapter of the OpenStack End User Guide for further instructions on logging in to the Dashboard.

The following sections describe the pros and cons of each option as it pertains to deploying the Dashboard.

Local memory cache

Local memory storage is the quickest and easiest session back end to set up, as it has no external dependencies whatsoever. It has the following significant drawbacks:

- No shared storage across processes or workers.
- No persistence after a process terminates.

The local memory back end 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.

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

```
  }
}
```

You can use applications such as `Memcached` or `Redis` for external caching. These applications offer persistence and shared storage and are useful for small-scale deployments and development.

Memcached

Memcached is a high-performance and distributed memory object caching system providing in-memory key-value store for small chunks of arbitrary data.

Requirements:

- Memcached service running and accessible.
- Python module `python-memcached` installed.

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

Redis

Redis is an open source, BSD licensed, advanced key-value store. It is often referred to as a data structure server.

Requirements:

- Redis service running and accessible.
- Python modules `redis` and `django-redis` installed.

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
    "default": {
        "BACKEND": "redis_cache.cache.RedisCache",
        "LOCATION": "127.0.0.1:6379:1",
        "OPTIONS": {
            "CLIENT_CLASS": "redis_cache.client.DefaultClient",
        }
    }
}
```

Initialize and configure the database

Database-backed sessions are scalable, persistent, and can be made high-concurrency and highly-available.

However, 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.

1. Start the MySQL command-line client.

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

2. Enter the MySQL root user's password when prompted.

3. To configure the MySQL database, create the dash database.

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

4. Create a MySQL user for the newly created dash database that has full control of the database. Replace DASH_DBPASS with a password for the new user.

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

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

6. In the `local_settings.py` file, change these options:

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

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

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

8. The following output is returned:

```
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.
```

9. To avoid a warning when you restart Apache on Ubuntu, create a `blackhole` directory in the Dashboard directory, as follows.

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

10. Restart the Apache service.

11. On Ubuntu, restart the `nova-api` service to ensure that the API server can connect to the Dashboard without error.

```
# service nova-api restart
```

Cached database

To mitigate the performance issues of database queries, you can use the Django `cached_db` session back end, which utilizes both your database and caching infrastructure to perform write-through caching and efficient retrieval.

Enable this hybrid setting by configuring both your database and cache, as discussed previously. Then, set the following value:

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

Cookies

If you use Django 1.4 or later, the `signed_cookies` back end avoids server load and scaling problems.

This back end stores session data in a cookie, which is stored by the user's browser. The back end 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 engine are that it requires no 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 that can be stored.

See the Django cookie-based sessions documentation.

Create and manage images

As an administrative user, you can create and manage images for the projects to which you belong. You can also create and manage images for users in all projects to which you have access.

To create and manage images in specified projects as an end user, see the upload and manage images with Dashboard in OpenStack End User Guide and manage images with CLI in OpenStack End User Guide .

To create and manage images as an administrator for other users, use the following procedures.

Create images

For details about image creation, see the Virtual Machine Image Guide.

1. Log in to the Dashboard and select the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Images* category. The images that you can administer for cloud users appear on this page.
3. Click *Create Image*, which opens the *Create An Image* window.

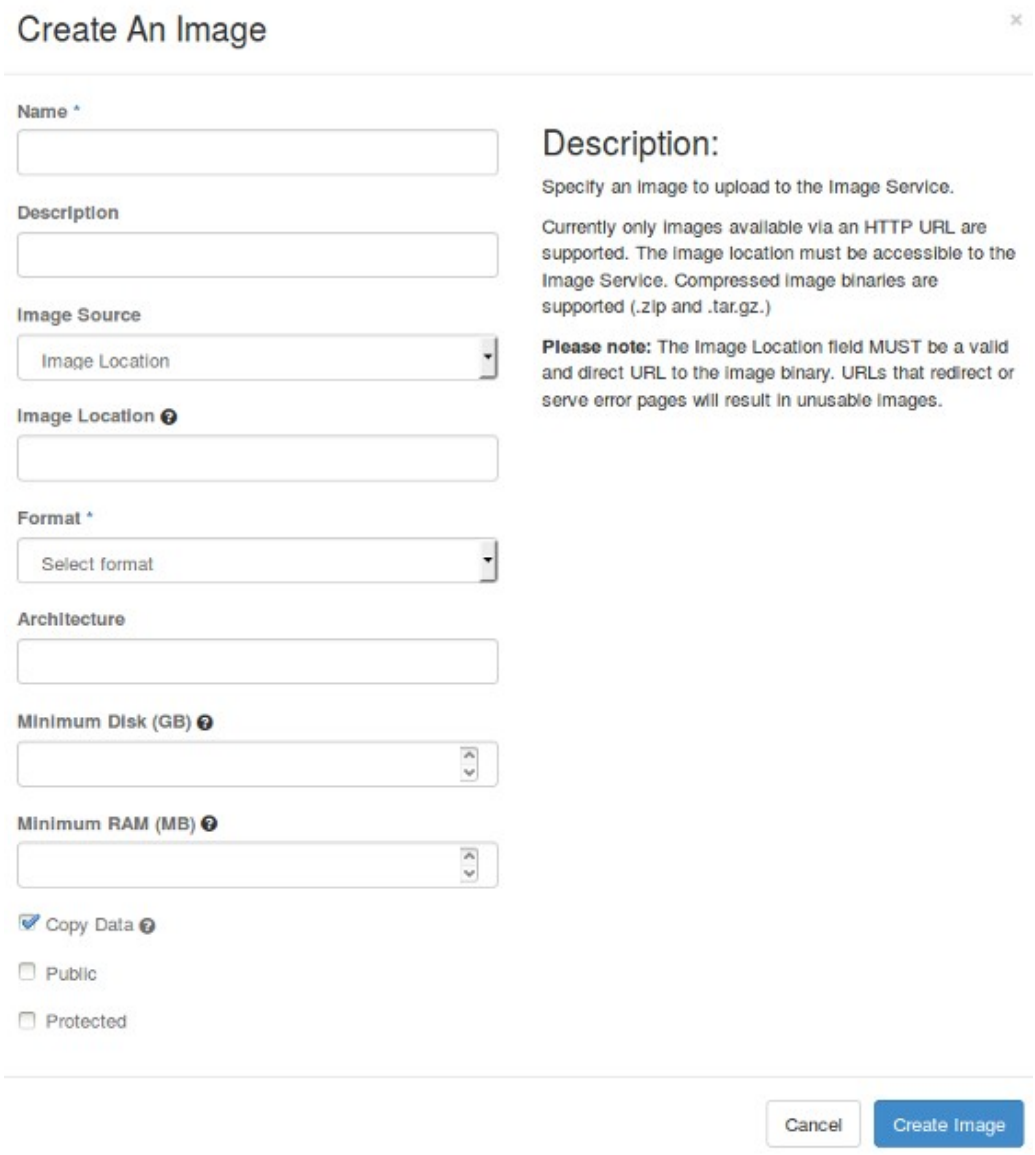


Figure Dashboard — Create Image

4. In the *Create An Image* window, enter or select the following values:

Name	Enter a name for the image.
Description	Enter a brief description of the image.
Image Source	Choose the image source from the dropdown list. Your choices are *Image Location* and *Image File*.
Image File or *Image Location*	Based on your selection, there is an *Image File* or *Image Location* field. You can include the location URL or browse for the image file on your file system and add it.
Kernel	Select the kernel to boot an AMI-

	style image.
Ramdisk	Select the ramdisk to boot an AMI-style image.
Format	Select the image format.
Architecture	Specify the architecture. For example, `i386` for a 32-bit architecture or `x86_64` for a 64-bit architecture.
Minimum Disk (GB)	Leave this field empty.
Minimum RAM (MB)	Leave this field empty.
Copy Data	Specify this option to copy image data to the Image service.
Public	Select this option to make the image public to all users.
Protected	Select this option to ensure that only users with permissions can delete it.

5. Click *Create Image*.

   The image is queued to be uploaded. It might take several minutes before the status changes from `Queued` to `Active`.

Update images

1. Log in to the Dashboard and select the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Images* category.
3. Select the images that you want to edit. Click *Edit Image*.
4. In the *Edit Image* window, you can change the image name.

   Select the *Public* check box to make the image public. Clear this check box to make the image private. You cannot change the *Kernel ID*, *Ramdisk ID*, or *Architecture* attributes for an image.

5. Click *Edit Image*.

Delete images

1. Log in to the Dashboard and select the *admin* project from the drop-down list.
2. On the *Admin tab*, open the *System* tab and click the *Images* category.
3. Select the images that you want to delete.
4. Click *Delete Images*.
5. In the *Confirm Delete Images* window, click *Delete Images* to confirm the deletion.

   You cannot undo this action.

Create and manage roles

A role is a personality that a user assumes to perform a specific set of operations. A role includes a set of rights and privileges. A user assumes that role inherits those rights and privileges.

> **Note**
>
> OpenStack Identity service defines a user's role on a project, but it is completely up to the individual service to define what that role means. This is referred to as the service's policy. To get details about what the privileges for each role are, refer to the `policy.json` file available for each service in the `/etc/SERVICE/policy.json` file. For example, the policy defined for OpenStack Identity service is defined in the `/etc/keystone/policy.json` file.

Create a role

1. Log in to the dashboard and select the *admin* project from the drop-down list.
2. On the *Identity* tab, click the *Roles* category.
3. Click the *Create Role* button.

   In the *Create Role* window, enter a name for the role.
4. Click the *Create Role* button to confirm your changes.

Edit a role

1. Log in to the dashboard and select the *Identity* project from the drop-down list.

2. On the *Identity* tab, click the *Roles* category.
3. Click the *Edit* button.

   In the *Update Role* window, enter a new name for the role.
4. Click the *Update Role* button to confirm your changes.

> **Note**
>
> Using the dashboard, you can edit only the name assigned to a role.

Delete a role

1. Log in to the dashboard and select the *Identity* project from the drop-down list.
2. On the *Identity* tab, click the *Roles* category.
3. Select the role you want to delete and click the *Delete Roles* button.
4. In the *Confirm Delete Roles* window, click *Delete Roles* to confirm the deletion.

   You cannot undo this action.

Manage instances

As an administrative user, you can manage instances for users in various projects. You can view, terminate, edit, perform a soft or hard reboot, create a snapshot from, and migrate instances. You can also view the logs for instances or launch a VNC console for an instance.

For information about using the Dashboard to launch instances as an end user, see the OpenStack End User Guide.

Create instance snapshots

1. Log in to the Dashboard and select the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Instances* category.
3. Select an instance to create a snapshot from it. From the *Actions* drop-down list, select *Create Snapshot*.
4. In the *Create Snapshot* window, enter a name for the snapshot.
5. Click *Create Snapshot*. The Dashboard shows the instance snapshot in the *Images* category.
6. To launch an instance from the snapshot, select the snapshot and click *Launch Instance*. For information about launching instances, see the OpenStack End User

Guide.

Control the state of an instance

1. Log in to the Dashboard and select the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Instances* category.
3. Select the instance for which you want to change the state.
4. From the drop-down list in the *Actions* column, select the state.

   Depending on the current state of the instance, you can perform various actions on the instance. For example, pause, un-pause, suspend, resume, soft or hard reboot, or terminate (actions in red are dangerous).

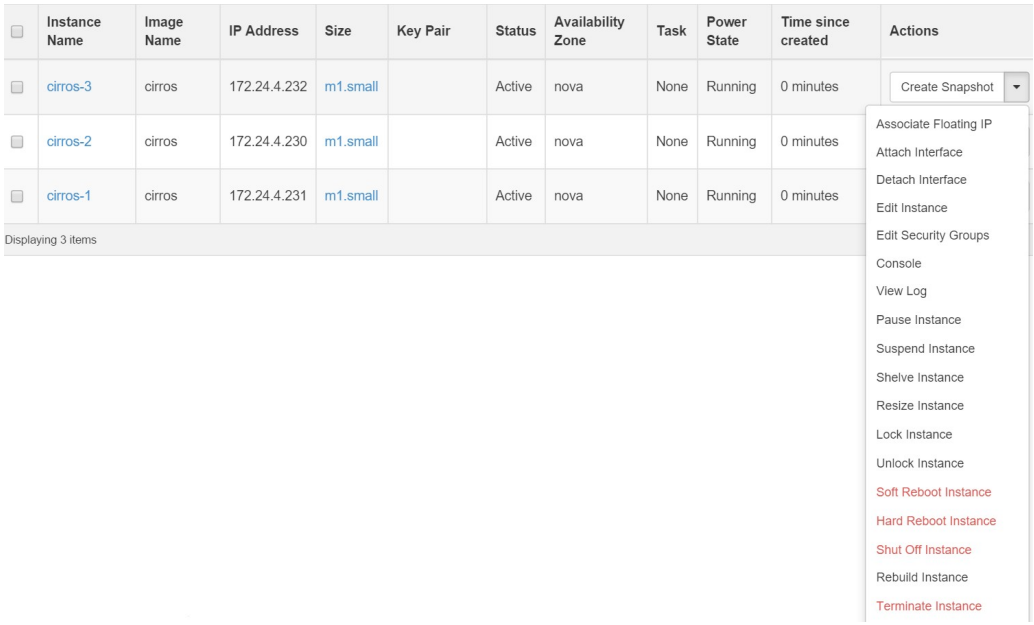


Figure Dashboard — Instance Actions

Track usage

Use the *Overview* category to track usage of instances for each project.

You can track costs per month by showing meters like number of VCPUs, disks, RAM, and uptime of all your instances.

1. Log in to the Dashboard and select the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Overview* category.
3. Select a month and click *Submit* to query the instance usage for that month.
4. Click *Download CSV Summary* to download a CSV summary.

Manage flavors

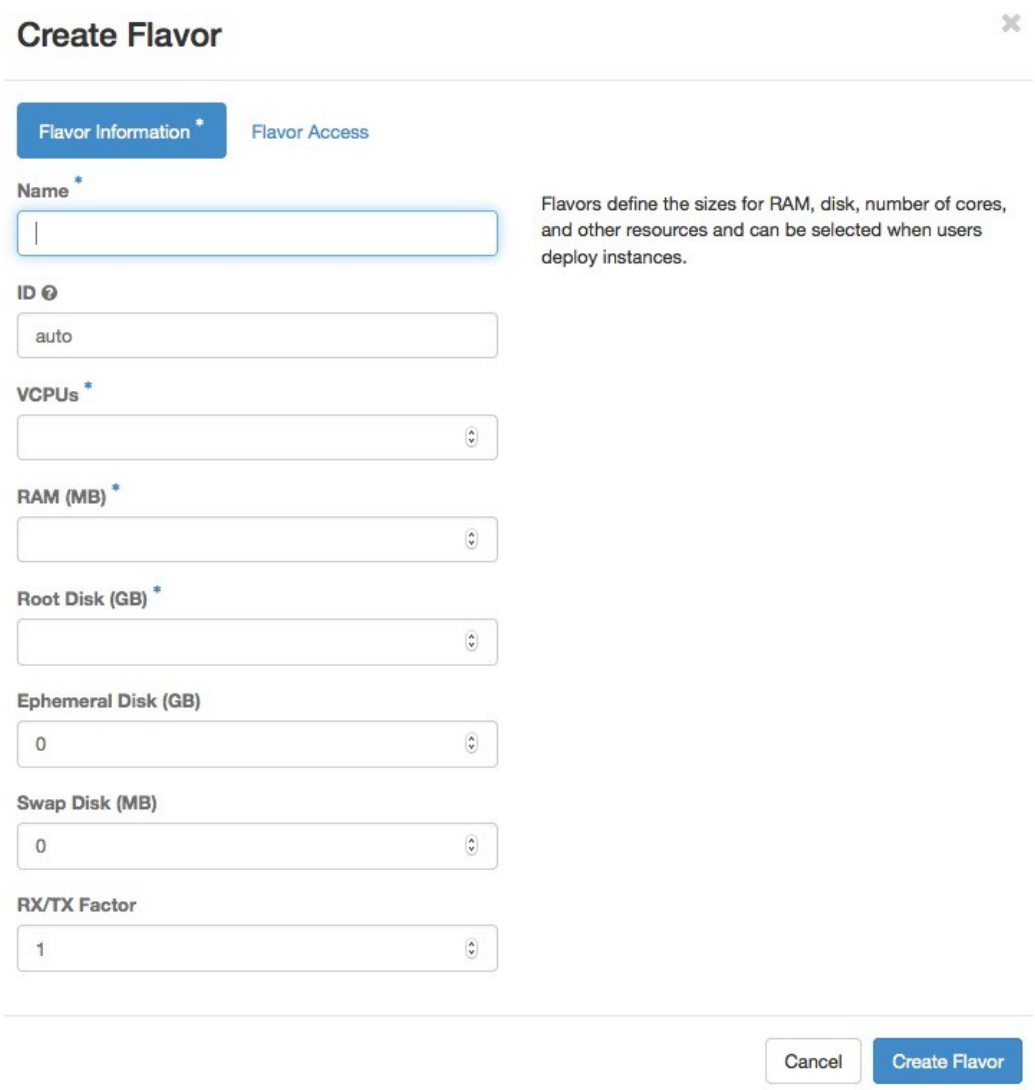
In OpenStack, a flavor defines the compute, memory, and storage capacity of a virtual server, also known as an instance. As an administrative user, you can create, edit, and delete flavors.

The following table lists the default flavors.

Flavor	VCPUs	Disk (in GB)	RAM (in MB)
m1.tiny	1	1	512
m1.small	1	20	2048
m1.medium	2	40	4096
m1.large	4	80	8192
m1.xlarge	8	160	16384

Create flavors

1. Log in to the Dashboard and select the *admin* project from the drop-down list.
2. In the *Admin* tab, open the *System* tab and click the *Flavors* category.
3. Click *Create Flavor*.
4. In the *Create Flavor* window, enter or select the parameters for the flavor in the *Flavor Information* tab.



Dashboard — Create Flavor

Name	Enter the flavor name.
ID	Unique ID (integer or UUID) for the new flavor. If specifying ‘auto’, a UUID will be automatically generated.
VCPUs	Enter the number of virtual CPUs to use.
RAM (MB)	Enter the amount of RAM to use, in megabytes.
Root Disk (GB)	Enter the amount of disk space in gigabytes to use for the root (/) partition.
Ephemeral Disk (GB)	Enter the amount of disk space in gigabytes to use for the ephemeral partition. If unspecified, the value is 0

	by default. Ephemeral disks offer machine local disk storage linked to the lifecycle of a VM instance. When a VM is terminated, all data on the ephemeral disk is lost. Ephemeral disks are not included in any snapshots.
Swap Disk (MB)	Enter the amount of swap space (in megabytes) to use. If unspecified, the default is 0.
RX/TX Factor	Optional property allows servers with a different bandwidth to be created with the RX/TX Factor. The default value is 1. That is, the new bandwidth is the same as that of the attached network.

5. In the *Flavor Access* tab, you can control access to the flavor by moving projects from the *All Projects* column to the *Selected Projects* column.

   Only projects in the *Selected Projects* column can use the flavor. If there are no projects in the right column, all projects can use the flavor.

6. Click *Create Flavor*.

Update flavors

1. Log in to the Dashboard and select the *admin* project from the drop-down list.
2. In the *Admin* tab, open the *System* tab and click the *Flavors* category.
3. Select the flavor that you want to edit. Click *Edit Flavor*.
4. In the *Edit Flavor* window, you can change the flavor name, VCPUs, RAM, root disk, ephemeral disk, and swap disk values.
5. Click *Save*.

Update Metadata

1. Log in to the Dashboard and select the *admin* project from the drop-down list.
2. In the *Admin* tab, open the *System* tab and click the *Flavors* category.
3. Select the flavor that you want to update. In the drop-down list, click *Update Metadata* or click *No* or *Yes* in the *Metadata* column.
4. In the *Update Flavor Metadata* window, you can customize some metadata keys,

then add it to this flavor and set them values.

5. Click *Save*.

Optional metadata keys

CPU limits	quota:cpu_shares
	quota:cpu_period
	quota:cpu_limit
	quota:cpu_reservation
	quota:cpu_quota
Disk tuning	quota:disk_read_bytes_sec
	quota:disk_read_iops_sec
	quota:disk_write_bytes_sec
	quota:disk_write_iops_sec
	quota:disk_total_bytes_sec
	quota:disk_total_iops_sec
Bandwidth I/O	quota:vif_inbound_average
	quota:vif_inbound_burst
	quota:vif_inbound_peak
	quota:vif_outbound_average
	quota:vif_outbound_burst
	quota:vif_outbound_peak
Watchdog behavior	hw:watchdog_action
Random-number generator	hw_rng:allowed
	hw_rng:rate_bytes
	hw_rng:rate_period

For information about supporting metadata keys, see the *Flavors*.

Delete flavors

1. Log in to the Dashboard and select the *admin* project from the drop-down list.
2. In the *Admin* tab, open the *System* tab and click the *Flavors* category.
3. Select the flavors that you want to delete.
4. Click *Delete Flavors*.
5. In the *Confirm Delete Flavors* window, click *Delete Flavors* to confirm the deletion. You cannot undo this action.

Manage volumes and volume types

Volumes are the Block Storage devices that you attach to instances to enable persistent storage. Users can attach a volume to a running instance or detach a volume and attach it to another instance at any time. For information about using the dashboard to create and manage volumes as an end user, see the OpenStack End User Guide.

As an administrative user, you can manage volumes and volume types for users in various projects. You can create and delete volume types, and you can view and delete volumes. Note that a volume can be encrypted by using the steps outlined below.

Create a volume type

1. Log in to the dashboard and select the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Volumes* category.
3. Click the *Volume Types* tab, and click *Create Volume Type* button. In the *Create Volume Type* window, enter a name for the volume type.
4. Click *Create Volume Type* button to confirm your changes.

> **Note**
>
> A message indicates whether the action succeeded.

Create an encrypted volume type

1. Create a volume type using the steps above for *Create a volume type*.
2. Click *Create Encryption* in the Actions column of the newly created volume type.
3. Configure the encrypted volume by setting the parameters below from available options (see table):

   Provider

   Specifies the class responsible for configuring the encryption.

   Control Location

   Specifies whether the encryption is from the front end (nova) or the back end (cinder).

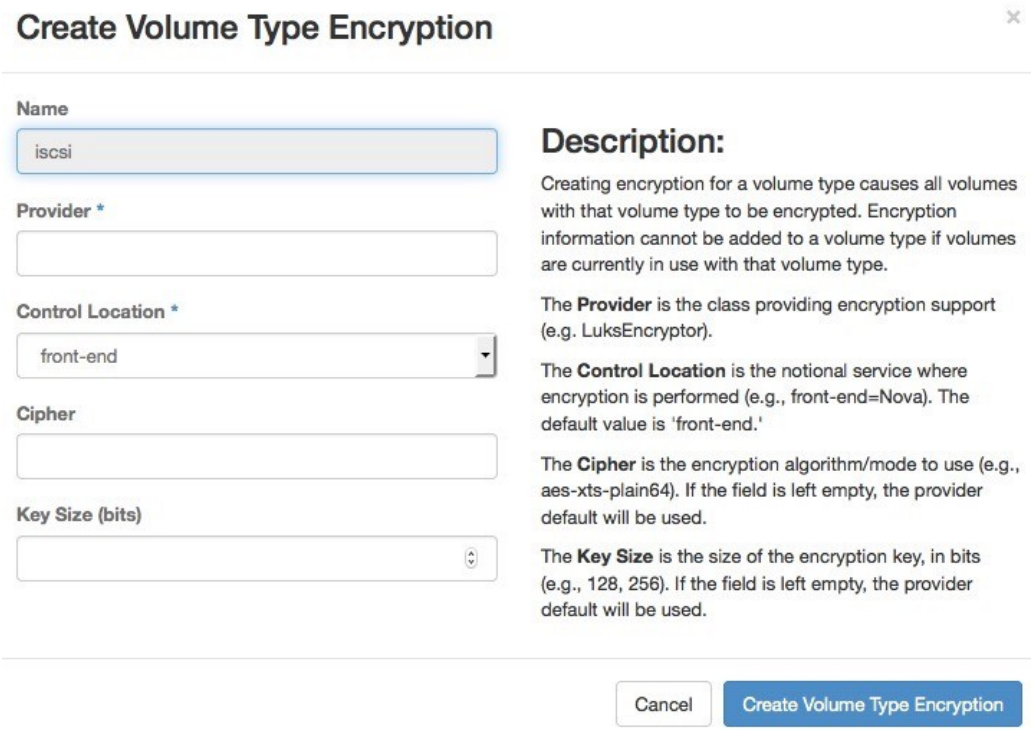
   Cipher

   Specifies the encryption algorithm.

Key Size (bits)

Specifies the encryption key size.

4. Click *Create Volume Type Encryption*.



Encryption Options

The table below provides a few alternatives available for creating encrypted volumes.

Encryption parameters	Parameter options	Comments
Provider	nova.volume.encryptors. luks.LuksEncryptor (Recommended)	Allows easier import and migration of imported encrypted volumes, and allows access key to be changed without re-encrypting the volume
	nova.volume.encryptors. cryptsetup. CryptsetupEncryptor	Less disk overhead than LUKS

Encryption parameters	Parameter options	Comments
Control Location	front-end (Recommended)	The encryption occurs within nova so that the data transmitted over the network is encrypted
	back-end	This could be selected if a cinder plug-in supporting an encrypted back-end block storage device becomes available in the future. TLS or other network encryption would also be needed to protect data as it traverses the network
Cipher	aes-xts-plain64 (Recommended)	See NIST reference below to see advantages*
	aes-cbc-essiv	Note: On the command line, type ‘cryptsetup benchmark’ for additional options
Key Size (bits)	512 (Recommended for aes-xts-plain64. 256 should be used for aes-cbc-essiv)	Using this selection for aes-xts, the underlying key size would only be 256-bits*
	256	Using this selection for aes-xts, the underlying key size would

Encryption parameters	Parameter options	Comments
		only be 128-bits*

* Source NIST SP 800-38E

Delete volume types

When you delete a volume type, volumes of that type are not deleted.

1. Log in to the dashboard and select the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Volumes* category.
3. Click the *Volume Types* tab, select the volume type or types that you want to delete.
4. Click *Delete Volume Types* button.
5. In the *Confirm Delete Volume Types* window, click the *Delete Volume Types* button to confirm the action.

> **Note**
>
> A message indicates whether the action succeeded.

Delete volumes

When you delete an instance, the data of its attached volumes is not destroyed.

1. Log in to the dashboard and select the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Volumes* category.
3. Select the volume or volumes that you want to delete.
4. Click *Delete Volumes* button.
5. In the *Confirm Delete Volumes* window, click the *Delete Volumes* button to confirm the action.

> **Note**
>
> A message indicates whether the action succeeded.

Manage shares and share types

Shares are file storage that instances can access. Users can allow or deny a running instance to have access to a share at any time. For information about using the Dashboard to create and manage shares as an end user, see the OpenStack End User Guide.

As an administrative user, you can manage shares and share types for users in various projects. You can create and delete share types, and view or delete shares.

Create a share type

1. Log in to the Dashboard and choose the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Shares* category.
3. Click the *Share Types* tab, and click *Create Share Type* button. In the *Create Share Type* window, enter or select the following values.

   Name: Enter a name for the share type.

   Driver handles share servers: Choose True or False

   Extra specs: To add extra specs, use key=value.
4. Click *Create Share Type* button to confirm your changes.

> **Note**
>
> A message indicates whether the action succeeded.

Update share type

1. Log in to the Dashboard and choose the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Shares* category.
3. Click the *Share Types* tab, select the share type that you want to update.
4. Select *Update Share Type* from Actions.
5. In the *Update Share Type* window, update extra specs.

   Extra specs: To add extra specs, use key=value. To unset extra specs, use key.
6. Click *Update Share Type* button to confirm your changes.

> **Note**
>
> A message indicates whether the action succeeded.

Delete share types

When you delete a share type, shares of that type are not deleted.

1. Log in to the Dashboard and choose the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Shares* category.
3. Click the *Share Types* tab, select the share type or types that you want to delete.
4. Click *Delete Share Types* button.
5. In the *Confirm Delete Share Types* window, click the *Delete Share Types* button to confirm the action.

> **Note**
>
> A message indicates whether the action succeeded.

Delete shares

1. Log in to the Dashboard and choose the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Shares* category.
3. Select the share or shares that you want to delete.
4. Click *Delete Shares* button.
5. In the *Confirm Delete Shares* window, click the *Delete Shares* button to confirm the action.

> **Note**
>
> A message indicates whether the action succeeded.

Delete share server

1. Log in to the Dashboard and choose the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Share Servers* category.
3. Select the share that you want to delete.
4. Click *Delete Share Server* button.
5. In the *Confirm Delete Share Server* window, click the *Delete Share Server* button to confirm the action.

> **Note**
>
> A message indicates whether the action succeeded.

Delete share networks

Note

Using the command-line interface, you can manage quotas for the OpenStack Compute service, the OpenStack Block Storage service, and the OpenStack Networking service (see OpenStack Administrator Guide). Additionally, you can update Compute service quotas for tenant users.

The following table describes the Compute and Block Storage service quotas:

Quota Descriptions

Quota Name	Defines the number of	Service
Gigabytes	Volume gigabytes allowed for each project.	Block Storage
Instances	Instances allowed for each project.	Compute
Injected Files	Injected files allowed for each project.	Compute
Injected File Content Bytes	Content bytes allowed for each injected file.	Compute
Keypairs	Number of keypairs.	Compute
Metadata Items	Metadata items allowed for each instance.	Compute
RAM (MB)	RAM megabytes allowed for each instance.	Compute
Security Groups	Security groups allowed for each project.	Compute
Security Group Rules	Rules allowed for each security group.	Compute
Snapshots	Volume snapshots allowed for each project.	Block Storage
VCPUs	Instance cores allowed for each project.	Compute
Volumes	Volumes allowed for each project.	Block Storage

1. Log in to the Dashboard and choose the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Share Networks* category.
3. Select the share network or share networks that you want to delete.
4. Click *Delete Share Networks* button.
5. In the *Confirm Delete Share Networks* window, click the *Delete Share Networks* button to confirm the action.

Note

A message indicates whether the action succeeded.

View and manage quotas

To prevent system capacities from being exhausted without notification, you can set up quotas. Quotas are operational limits. For example, the number of gigabytes allowed for each tenant can be controlled so that cloud resources are optimized. Quotas can be enforced at both the tenant (or project) and the tenant-user level.

Typically, you change quotas when a project needs more than ten volumes or 1 TB on a compute node.

Using the Dashboard, you can view default Compute and Block Storage quotas for new tenants, as well as update quotas for existing tenants.

View default project quotas

1. Log in to the dashboard and select the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Defaults* category.

3. The default quota values are displayed.

> **Note**
>
> You can sort the table by clicking on either the *Quota Name* or *Limit* column headers.

Update project quotas

1. Log in to the dashboard and select the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Defaults* category.
3. Click the *Update Defaults* button.
4. In the *Update Default Quotas* window, you can edit the default quota values.
5. Click the *Update Defaults* button.

> **Note**
>
> The dashboard does not show all possible project quotas. To view and update the quotas for a service, use its command-line client. See OpenStack Administrator Guide.

View cloud resources

- View services information
- View cloud usage statistics
    - View resource statistics

View services information

As an administrative user, you can view information for OpenStack services.

1. Log in to the Dashboard and select the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *System Information* category.

   View the following information on these tabs:

   - *Services*: Displays the internal name and the public OpenStack name for each service, the host on which the service runs, and whether or not the service is enabled.
   - *Compute Services*: Displays information specific to the Compute service. Both host and zone are listed for each service, as well as its activation status.
   - *Block Storage Services*: Displays information specific to the Block Storage service. Both host and zone are listed for each service, as well as its

activation status.

- *Network Agents*: Displays the network agents active within the cluster, such as L3 and DHCP agents, and the status of each agent.
- *Orchestration Services*: Displays information specific to the Orchestration service. Name, engine id, host and topic are listed for each service, as well as its activation status.

View cloud usage statistics

The Telemetry service provides user-level usage data for OpenStack-based clouds, which can be used for customer billing, system monitoring, or alerts. Data can be collected by notifications sent by existing OpenStack components (for example, usage events emitted from Compute) or by polling the infrastructure (for example, libvirt).

> **Note**
>
> You can only view metering statistics on the dashboard (available only to administrators). The Telemetry service must be set up and administered through the **ceilometer** command-line interface (CLI).
>
> For basic administration information, refer to the “Measure Cloud Resources” chapter in the OpenStack End User Guide.

View resource statistics

1. Log in to the dashboard and select the *admin* project from the drop-down list.
2. On the *Admin* tab, click the *Resource Usage* category.
3. Click the:
   - *Usage Report* tab to view a usage report per tenant (project) by specifying the time period (or even use a calendar to define a date range).
   - *Stats* tab to view a multi-series line chart with user-defined meters. You group by project, define the value type (min, max, avg, or sum), and specify the time period (or even use a calendar to define a date range).

Create and manage host aggregates

Host aggregates enable administrative users to assign key-value pairs to groups of machines.

Each node can have multiple aggregates and each aggregate can have multiple key-value pairs. You can assign the same key-value pair to multiple aggregates.

The scheduler uses this information to make scheduling decisions. For information, see Scheduling.

To create a host aggregate

1. Log in to the Dashboard and select the *admin* project from the drop-down list.
2. On the *Admin* tab, open the *System* tab and click the *Host Aggregates* category.
3. Click *Create Host Aggregate*.
4. In the *Create Host Aggregate* dialog box, enter or select the following values on the *Host Aggregate Information* tab:
    - *Name*: The host aggregate name.
    - *Availability Zone*: The cloud provider defines the default availability zone, such as `us-west`, `apac-south`, or `nova`. You can target the host aggregate, as follows:
        - When the host aggregate is exposed as an availability zone, select the availability zone when you launch an instance.
        - When the host aggregate is not exposed as an availability zone, select a flavor and its extra specs to target the host aggregate.
5. Assign hosts to the aggregate using the *Manage Hosts within Aggregate* tab in the same dialog box.

    To assign a host to the aggregate, click **+** for the host. The host moves from the *All available hosts* list to the *Selected hosts* list.

You can add one host to one or more aggregates. To add a host to an existing aggregate, edit the aggregate.

To manage host aggregates

1. Select the *admin* project from the drop-down list at the top of the page.
2. On the *Admin* tab, open the *System* tab and click the *Host Aggregates* category.
    - To edit host aggregates, select the host aggregate that you want to edit. Click *Edit Host Aggregate*.

      In the *Edit Host Aggregate* dialog box, you can change the name and availability zone for the aggregate.
    - To manage hosts, locate the host aggregate that you want to edit in the table. Click *More* and select *Manage Hosts*.

      In the *Add/Remove Hosts to Aggregate* dialog box, click **+** to assign a host to an aggregate. Click **-** to remove a host that is assigned to an aggregate.

- To delete host aggregates, locate the host aggregate that you want to edit in the table. Click *More* and select *Delete Host Aggregate*.

Launch and manage stacks using the Dashboard

The Orchestration service provides a template-based orchestration engine for the OpenStack cloud. Orchestration services create and manage cloud infrastructure resources such as storage, networking, instances, and applications as a repeatable running environment.

Administrators use templates to create stacks, which are collections of resources. For example, a stack might include instances, floating IPs, volumes, security groups, or users. The Orchestration service offers access to all OpenStack core services via a single modular template, with additional orchestration capabilities such as auto-scaling and basic high availability.

For information about:

- administrative tasks on the command-line, see the OpenStack Administrator Guide.

> **Note**
>
> There are no administration-specific tasks that can be done through the Dashboard.

- the basic creation and deletion of Orchestration stacks, refer to the OpenStack End User Guide.

Compute

The OpenStack Compute service allows you to control an *Infrastructure-as-a-Service (IaaS)* cloud computing platform. It gives you control over instances and networks, and allows you to manage access to the cloud through users and projects.

Compute does not include virtualization software. Instead, it defines drivers that interact with underlying virtualization mechanisms that run on your host operating system, and exposes functionality over a web-based API.

- System architecture

System architecture

OpenStack Compute contains several main components.

- The *cloud controller* represents the global state and interacts with the other components. The `API server` acts as the web services front end for the cloud controller. The `compute controller` provides compute server resources and usually also contains the Compute service.
- The `object store` is an optional component that provides storage services; you can also use OpenStack Object Storage instead.
- An `auth manager` provides authentication and authorization services when used with the Compute system; you can also use OpenStack Identity as a separate authentication service instead.
- A `volume controller` provides fast and permanent block-level storage for the compute servers.
- The `network controller` provides virtual networks to enable compute servers to interact with each other and with the public network. You can also use OpenStack Networking instead.
- The `scheduler` is used to select the most suitable compute controller to host an instance.

Compute uses a messaging-based, `shared nothing` architecture. All major components exist on multiple servers, including the compute, volume, and network controllers, and the Object Storage or Image service. The state of the entire system is stored in a database. The cloud controller communicates with the internal object store using HTTP, but it communicates with the scheduler, network controller, and volume controller using Advanced Message Queuing Protocol (AMQP). To avoid blocking a component while waiting for a response, Compute uses asynchronous calls, with a callback that is triggered when a response is received.

Hypervisors

Compute controls hypervisors through an API server. Selecting the best hypervisor to use can be difficult, and you must take budget, resource constraints, supported features, and required technical specifications into account. However, the majority of OpenStack development is done on systems using KVM and Xen-based hypervisors. For a detailed list of features and support across different hypervisors, see http://wiki.openstack.org/HypervisorSupportMatrix.

You can also orchestrate clouds using multiple hypervisors in different availability zones. Compute supports the following hypervisors:

- Baremetal
- Docker
- Hyper-V
- Kernel-based Virtual Machine (KVM)
- Linux Containers (LXC)
- Quick Emulator (QEMU)
- User Mode Linux (UML)
- VMware vSphere
- Xen

For more information about hypervisors, see the Hypervisors section in the OpenStack Configuration Reference.

Tenants, users, and roles

The Compute system is designed to be used by different consumers in the form of tenants on a shared system, and role-based access assignments. Roles control the actions that a user is allowed to perform.

Tenants are isolated resource containers that form the principal organizational structure within the Compute service. They consist of an individual VLAN, and volumes, instances, images, keys, and users. A user can specify the tenant by appending `project_id` to their access key. If no tenant is specified in the API request, Compute attempts to use a tenant with the same ID as the user.

For tenants, you can use quota controls to limit the:

- Number of volumes that can be launched.
- Number of processor cores and the amount of RAM that can be allocated.
- Floating IP addresses assigned to any instance when it launches. This allows instances to have the same publicly accessible IP addresses.
- Fixed IP addresses assigned to the same instance when it launches. This allows

instances to have the same publicly or privately accessible IP addresses.

Roles control the actions a user is allowed to perform. By default, most actions do not require a particular role, but you can configure them by editing the `policy.json` file for user roles. For example, a rule can be defined so that a user must have the `admin` role in order to be able to allocate a public IP address.

A tenant limits users' access to particular images. Each user is assigned a user name and password. Keypairs granting access to an instance are enabled for each user, but quotas are set, so that each tenant can control resource consumption across available hardware resources.

> **Note**
>
> Earlier versions of OpenStack used the term `project` instead of `tenant`. Because of this legacy terminology, some command-line tools use *--project_id* where you would normally expect to enter a tenant ID.

Block storage

OpenStack provides two classes of block storage: ephemeral storage and persistent volume.

Ephemeral storage

Ephemeral storage includes a root ephemeral volume and an additional ephemeral volume.

The root disk is associated with an instance, and exists only for the life of this very instance. Generally, it is used to store an instance's root file system, persists across the guest operating system reboots, and is removed on an instance deletion. The amount of the root ephemeral volume is defined by the flavor of an instance.

In addition to the ephemeral root volume, all default types of flavors, except `m1.tiny`, which is the smallest one, provide an additional ephemeral block device sized between 20 and 160 GB (a configurable value to suit an environment). It is represented as a raw block device with no partition table or file system. A cloud-aware operating system can discover, format, and mount such a storage device. OpenStack Compute defines the default file system for different operating systems as Ext4 for Linux distributions, VFAT for non-Linux and non-Windows operating systems, and NTFS for Windows. However, it is possible to specify any other filesystem type by using `virt_mkfs` or `default_ephemeral_format` configuration options.

> **Note**

> For example, the `cloud-init` package included into an Ubuntu's stock cloud image, by default, formats this space as an Ext4 file system and mounts it on `/mnt`. This is a cloud-init feature, and is not an OpenStack mechanism. OpenStack only provisions the raw storage.

Persistent volume

A persistent volume is represented by a persistent virtualized block device independent of any particular instance, and provided by OpenStack Block Storage.

Only a single configured instance can access a persistent volume. Multiple instances cannot access a persistent volume. This type of configuration requires a traditional network file system to allow multiple instances accessing the persistent volume. It also requires a traditional network file system like NFS, CIFS, or a cluster file system such as GlusterFS. These systems can be built within an OpenStack cluster, or provisioned outside of it, but OpenStack software does not provide these features.

You can configure a persistent volume as bootable and use it to provide a persistent virtual instance similar to the traditional non-cloud-based virtualization system. It is still possible for the resulting instance to keep ephemeral storage, depending on the flavor selected. In this case, the root file system can be on the persistent volume, and its state is maintained, even if the instance is shut down. For more information about this type of configuration, see the OpenStack Configuration Reference.

> **Note**
>
> A persistent volume does not provide concurrent access from multiple instances. That type of configuration requires a traditional network file system like NFS, or CIFS, or a cluster file system such as GlusterFS. These systems can be built within an OpenStack cluster, or provisioned outside of it, but OpenStack software does not provide these features.

EC2 compatibility API

In addition to the native compute API, OpenStack provides an EC2-compatible API. This API allows EC2 legacy workflows built for EC2 to work with OpenStack.

> **Warning**
>
> Nova in tree EC2-compatible API is deprecated. The ec2-api project is working to implement the EC2 API.

You can use numerous third-party tools and language-specific SDKs to interact with OpenStack clouds. You can use both native and compatibility APIs. Some of the more

popular third-party tools are:

Euca2ools
A popular open source command-line tool for interacting with the EC2 API. This is convenient for multi-cloud environments where EC2 is the common API, or for transitioning from EC2-based clouds to OpenStack. For more information, see the Eucalyptus Documentation.

Hybridfox
A Firefox browser add-on that provides a graphical interface to many popular public and private cloud technologies, including OpenStack. For more information, see the hybridfox site.

boto
Python library for interacting with Amazon Web Services. You can use this library to access OpenStack through the EC2 compatibility API. For more information, see the boto project page on GitHub.

fog
A Ruby cloud services library. It provides methods to interact with a large number of cloud and virtualization platforms, including OpenStack. For more information, see the fog site.

php-opencloud
A PHP SDK designed to work with most OpenStack-based cloud deployments, as well as Rackspace public cloud. For more information, see the php-opencloud site.

Building blocks

In OpenStack the base operating system is usually copied from an image stored in the OpenStack Image service. This is the most common case and results in an ephemeral instance that starts from a known template state and loses all accumulated states on virtual machine deletion. It is also possible to put an operating system on a persistent volume in the OpenStack Block Storage volume system. This gives a more traditional persistent system that accumulates states which are preserved on the OpenStack Block Storage volume across the deletion and re-creation of the virtual machine. To get a list of available images on your system, run:

```
$ nova image-list
+--------------------------------------+-----------------------------+--------+---------+
| ID                                   | Name                        | Status | Server  |
+--------------------------------------+-----------------------------+--------+---------+
| aee1d242-730f-431f-88c1-87630c0f07ba | Ubuntu 14.04 cloudimg amd64 | ACTIVE |         |
| 0b27baa1-0ca6-49a7-b3f4-48388e440245 | Ubuntu 14.10 cloudimg amd64 | ACTIVE |         |
| df8d56fc-9cea-4dfd-a8d3-28764de3cb08 | jenkins                     | ACTIVE |         |
+--------------------------------------+-----------------------------+--------+---------+
```

The displayed image attributes are:

ID

Automatically generated UUID of the image

Name

Free form, human-readable name for image

Status

The status of the image. Images marked `ACTIVE` are available for use.

Server

For images that are created as snapshots of running instances, this is the UUID of the instance the snapshot derives from. For uploaded images, this field is blank.

Virtual hardware templates are called `flavors`. The default installation provides five flavors. By default, these are configurable by admin users, however that behavior can be changed by redefining the access controls for `compute_extension:flavormanage` in `/etc/nova/policy.json` on the `compute-api` server.

For a list of flavors that are available on your system:

```
$ nova flavor-list
+-----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| ID  | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+-----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| 1   | m1.tiny   | 512       | 1    | 0         |      | 1     | 1.0         | True      |
| 2   | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True      |
| 3   | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 4   | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 5   | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
+-----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
```

Compute service architecture

These basic categories describe the service architecture and information about the cloud controller.

API server

At the heart of the cloud framework is an API server, which makes command and control of the hypervisor, storage, and networking programmatically available to users.

The API endpoints are basic HTTP web services which handle authentication, authorization, and basic command and control functions using various API interfaces under the Amazon, Rackspace, and related models. This enables API compatibility with multiple existing tool sets created for interaction with offerings from other vendors. This broad compatibility prevents vendor lock-in.

Message queue

A messaging queue brokers the interaction between compute nodes (processing), the networking controllers (software which controls network infrastructure), API endpoints, the scheduler (determines which physical hardware to allocate to a virtual resource), and

similar components. Communication to and from the cloud controller is handled by HTTP requests through multiple API endpoints.

A typical message passing event begins with the API server receiving a request from a user. The API server authenticates the user and ensures that they are permitted to issue the subject command. The availability of objects implicated in the request is evaluated and, if available, the request is routed to the queuing engine for the relevant workers. Workers continually listen to the queue based on their role, and occasionally their type host name. When an applicable work request arrives on the queue, the worker takes assignment of the task and begins executing it. Upon completion, a response is dispatched to the queue which is received by the API server and relayed to the originating user. Database entries are queried, added, or removed as necessary during the process.

Compute worker

Compute workers manage computing instances on host machines. The API dispatches commands to compute workers to complete these tasks:

- Run instances
- Delete instances (Terminate instances)
- Reboot instances
- Attach volumes
- Detach volumes
- Get console output

Network Controller

The Network Controller manages the networking resources on host machines. The API server dispatches commands through the message queue, which are subsequently processed by Network Controllers. Specific operations include:

- Allocating fixed IP addresses
- Configuring VLANs for projects
- Configuring networks for compute nodes

Images and instances

Virtual machine images contain a virtual disk that holds a bootable operating system on it. Disk images provide templates for virtual machine file systems. The Image service controls image storage and management.

Instances are the individual virtual machines that run on physical compute nodes inside the cloud. Users can launch any number of instances from the same image. Each launched instance runs from a copy of the base image. Any changes made to the instance do not affect the base image. Snapshots capture the state of an instances running disk. Users can

create a snapshot, and build a new image based on these snapshots. The Compute service controls instance, image, and snapshot storage and management.

When you launch an instance, you must choose a `flavor`, which represents a set of virtual resources. Flavors define virtual CPU number, RAM amount available, and ephemeral disks size. Users must select from the set of available flavors defined on their cloud. OpenStack provides a number of predefined flavors that you can edit or add to.

> **Note**
>
> - For more information about creating and troubleshooting images, see the OpenStack Virtual Machine Image Guide.
> - For more information about image configuration options, see the Image services section of the OpenStack Configuration Reference.
> - For more information about flavors, see *Flavors*.

You can add and remove additional resources from running instances, such as persistent volume storage, or public IP addresses. The example used in this chapter is of a typical virtual system within an OpenStack cloud. It uses the `cinder-volume` service, which provides persistent block storage, instead of the ephemeral storage provided by the selected instance flavor.

This diagram shows the system state prior to launching an instance. The image store has a number of predefined images, supported by the Image service. Inside the cloud, a compute node contains the available vCPU, memory, and local disk resources. Additionally, the `cinder-volume` service stores predefined volumes.

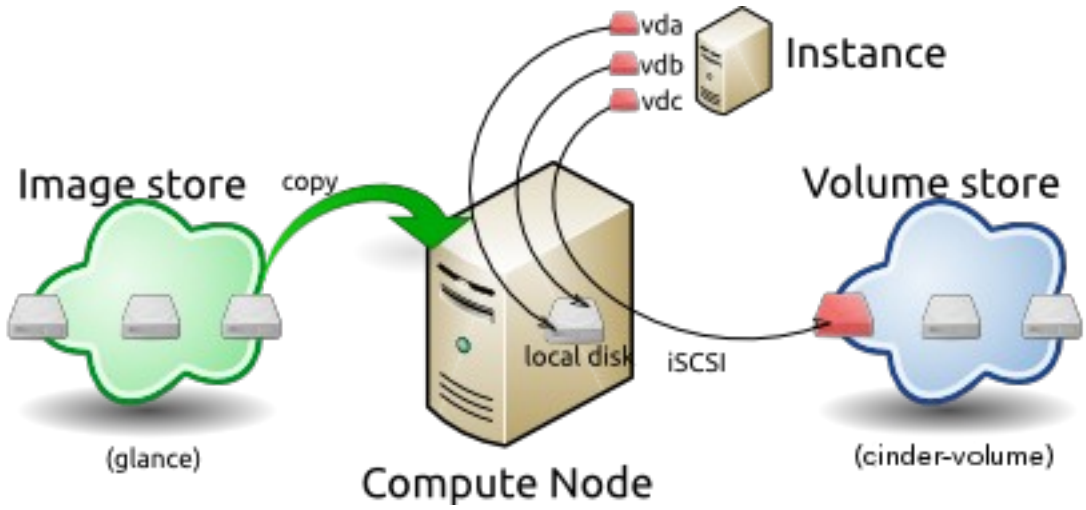
The base image state with no running instances


Image store
(glance)
local disk
Compute Node
Volume store
(cinder-volume)

Instance Launch

To launch an instance, select an image, flavor, and any optional attributes. The selected flavor provides a root volume, labeled `vda` in this diagram, and additional ephemeral storage, labeled `vdb`. In this example, the `cinder-volume` store is mapped to the third virtual disk on this instance, `vdc`.

Instance creation from an image



The Image service copies the base image from the image store to the local disk. The local disk is the first disk that the instance accesses, which is the root volume labeled `vda`. Smaller instances start faster. Less data needs to be copied across the network.

The new empty ephemeral disk is also created, labeled `vdb`. This disk is deleted when you delete the instance.

The compute node connects to the attached `cinder-volume` using iSCSI. The `cinder-volume` is mapped to the third disk, labeled `vdc` in this diagram. After the compute node provisions the vCPU and memory resources, the instance boots up from root volume `vda`. The instance runs and changes data on the disks (highlighted in red on the diagram). If the volume store is located on a separate network, the `my_block_storage_ip` option specified in the storage node configuration file directs image traffic to the compute node.

> **Note**
>
> Some details in this example scenario might be different in your environment. For example, you might use a different type of back-end storage, or different network protocols. One common variant is that the ephemeral storage used for volumes `vda` and `vdb` could be backed by network storage rather than a local disk.

When you delete an instance, the state is reclaimed with the exception of the persistent volume. The ephemeral storage, whether encrypted or not, is purged. Memory and vCPU resources are released. The image remains unchanged throughout this process.

The end state of an image and volume after the instance exits

Image properties and property protection

An image property is a key and value pair that the administrator or the image owner attaches to an OpenStack Image service image, as follows:

- The administrator defines core properties, such as the image name.
- The administrator and the image owner can define additional properties, such as licensing and billing information.

The administrator can configure any property as protected, which limits which policies or user roles can perform CRUD operations on that property. Protected properties are generally additional properties to which only administrators have access.

For unprotected image properties, the administrator can manage core properties and the image owner can manage additional properties.

To configure property protection

To configure property protection, edit the `policy.json` file. This file can also be used to set policies for Image service actions.

1. Define roles or policies in the `policy.json` file:

```
{
    "context_is_admin":  "role:admin",
    "default": "",

    "add_image": "",
    "delete_image": "",
```

```
    "get_image": "",
    "get_images": "",
    "modify_image": "",
    "publicize_image": "role:admin",
    "copy_from": "",

    "download_image": "",
    "upload_image": "",

    "delete_image_location": "",
    "get_image_location": "",
    "set_image_location": "",

    "add_member": "",
    "delete_member": "",
    "get_member": "",
    "get_members": "",
    "modify_member": "",

    "manage_image_cache": "role:admin",

    "get_task": "",
    "get_tasks": "",
    "add_task": "",
    "modify_task": "",

    "deactivate": "",
    "reactivate": "",

    "get_metadef_namespace": "",
    "get_metadef_namespaces":"",
    "modify_metadef_namespace":"",
    "add_metadef_namespace":"",
    "delete_metadef_namespace":"",

    "get_metadef_object":"",
    "get_metadef_objects":"",
    "modify_metadef_object":"",
    "add_metadef_object":"",

    "list_metadef_resource_types":"",
    "get_metadef_resource_type":"",
    "add_metadef_resource_type_association":"",
```

```
    "get_metadef_property":"",
    "get_metadef_properties":"",
    "modify_metadef_property":"",
    "add_metadef_property":"",

    "get_metadef_tag":"",
    "get_metadef_tags":"",
    "modify_metadef_tag":"",
    "add_metadef_tag":"",
    "add_metadef_tags":""
}
```

For each parameter, use `"rule:restricted"` to restrict access to all users or `"role:admin"` to limit access to administrator roles. For example:

```
"download_image":
"upload_image":
```

2. Define which roles or policies can manage which properties in a property protections configuration file. For example:

```
[x_none_read]
create = context_is_admin
read = !
update = !
delete = !

[x_none_update]
create = context_is_admin
read = context_is_admin
update = !
delete = context_is_admin

[x_none_delete]
create = context_is_admin
read = context_is_admin
update = context_is_admin
delete = !
```

   - A value of `@` allows the corresponding operation for a property.
   - A value of `!` disallows the corresponding operation for a property.

3. In the `glance-api.conf` file, define the location of a property protections configuration file.

```
property_protection_file = {file_name}
```

This file contains the rules for property protections and the roles and policies associated with it.

By default, property protections are not enforced.

If you specify a file name value and the file is not found, the `glance-api` service does not start.

To view a sample configuration file, see glance-api.conf.

4. Optionally, in the `glance-api.conf` file, specify whether roles or policies are used in the property protections configuration file

```
property_protection_rule_format = roles
```

The default is `roles`.

To view a sample configuration file, see glance-api.conf.

Image download: how it works

Prior to starting a virtual machine, transfer the virtual machine image to the compute node from the Image service. How this works can change depending on the settings chosen for the compute node and the Image service.

Typically, the Compute service will use the image identifier passed to it by the scheduler service and request the image from the Image API. Though images are not stored in glance—rather in a back end, which could be Object Storage, a filesystem or any other supported method—the connection is made from the compute node to the Image service and the image is transferred over this connection. The Image service streams the image from the back end to the compute node.

It is possible to set up the Object Storage node on a separate network, and still allow image traffic to flow between the Compute and Object Storage nodes. Configure the `my_block_storage_ip` option in the storage node configuration file to allow block storage traffic to reach the Compute node.

Certain back ends support a more direct method, where on request the Image service will return a URL that links directly to the back-end store. You can download the image using this approach. Currently, the only store to support the direct download approach is the filesystem store. Configured the approach using the `filesystems` option in the `image_file_url` section of the `nova.conf` file on compute nodes.

Compute nodes also implement caching of images, meaning that if an image has been used before it won't necessarily be downloaded every time. Information on the configuration options for caching on compute nodes can be found in the Configuration Reference.

Instance building blocks

In OpenStack, the base operating system is usually copied from an image stored in the OpenStack Image service. This results in an ephemeral instance that starts from a known template state and loses all accumulated states on shutdown.

You can also put an operating system on a persistent volume in Compute or the Block Storage volume system. This gives a more traditional, persistent system that accumulates states that are preserved across restarts. To get a list of available images on your system, run:

```
$ nova image-list
+--------------------------+------------------+--------+----------------+
| ID                       | Name             | Status | Server         |
+--------------------------+------------------+--------+----------------+
| aee1d242-730f-431f-88c1- |                  |        |                |
| 87630c0f07ba             | Ubuntu 14.04     |        |                |
|                          | cloudimg amd64   | ACTIVE |                |
| 0b27baa1-0ca6-49a7-b3f4- |                  |        |                |
| 48388e440245             | Ubuntu 14.10     |        |                |
|                          | cloudimg amd64   | ACTIVE |                |
| df8d56fc-9cea-4dfd-a8d3- |                  |        |                |
| 28764de3cb08             | jenkins          | ACTIVE |                |
+--------------------------+------------------+--------+----------------+
```

The displayed image attributes are:

`ID`
: Automatically generated UUID of the image.

`Name`
: Free form, human-readable name for the image.

`Status`
: The status of the image. Images marked `ACTIVE` are available for use.

`Server`
: For images that are created as snapshots of running instances, this is the UUID of the instance the snapshot derives from. For uploaded images, this field is blank.

Virtual hardware templates are called `flavors`. The default installation provides five predefined flavors.

For a list of flavors that are available on your system, run:

```
$ nova flavor-list
+----+----------+----------+-----+----------+-----+------+------------+----------+
| ID | Name     | Memory_MB| Disk| Ephemeral| Swap| VCPUs| RXTX_Factor| Is_Public|
+----+----------+----------+-----+----------+-----+------+------------+----------+
| 1  | m1.tiny  | 512      | 1   | 0        |     | 1    | 1.0        | True     |
| 2  | m1.small | 2048     | 20  | 0        |     | 1    | 1.0        | True     |
| 3  | m1.medium| 4096     | 40  | 0        |     | 2    | 1.0        | True     |
| 4  | m1.large | 8192     | 80  | 0        |     | 4    | 1.0        | True     |
| 5  | m1.xlarge| 16384    | 160 | 0        |     | 8    | 1.0        | True     |
+----+----------+----------+-----+----------+-----+------+------------+----------+
```

By default, administrative users can configure the flavors. You can change this behavior by redefining the access controls for `compute_extension:flavormanage` in `/etc/nova/policy.json` on the `compute-api` server.

Instance management tools

OpenStack provides command-line, web interface, and API-based instance management tools. Third-party management tools are also available, using either the native API or the provided EC2-compatible API.

The OpenStack python-novaclient package provides a basic command-line utility, which uses the **nova** command. This is available as a native package for most Linux distributions, or you can install the latest version using the pip python package installer:

```
# pip install python-novaclient
```

For more information about python-novaclient and other command-line tools, see the OpenStack End User Guide.

Control where instances run

The OpenStack Configuration Reference provides detailed information on controlling where your instances run, including ensuring a set of instances run on different compute nodes for service resiliency or on the same node for high performance inter-instance communications.

Administrative users can specify which compute node their instances run on. To do this, specify the `--availability-zone AVAILABILITY_ZONE:COMPUTE_HOST` parameter.

Launch instances with UEFI

Unified Extensible Firmware Interface (UEFI) is a standard firmware designed to replace legacy BIOS. There is a slow but steady trend for operating systems to move to the UEFI format and, in some cases, make it their only format.

To configure UEFI environment

To successfully launch an instance from an UEFI image in QEMU/KVM environment, the administrator has to install the following packages on compute node:

- OVMF, a port of Intel's tianocore firmware to QEMU virtual machine.
- libvirt, which has been supporting UEFI boot since version 1.2.9.

Because default UEFI loader path is `/usr/share/OVMF/OVMF_CODE.fd`, the administrator must create one link to this location after UEFI package is installed.

To upload UEFI images

To launch instances from a UEFI image, the administrator first has to upload one UEFI image. To do so, `hw_firmware_type` property must be set to `uefi` when the image is created. For example:

```
$ glance image-create --container-format bare --disk-format qcow2 \
  --property hw_firmware_type=uefi --file /tmp/cloud-uefi.qcow --name uefi
```

After that, you can launch instances from this UEFI image.

Networking with nova-network

Understanding the networking configuration options helps you design the best configuration for your Compute instances.

You can choose to either install and configure `nova-network` or use the OpenStack Networking service (neutron). This section contains a brief overview of `nova-network`. For more information about OpenStack Networking, see *Networking*.

Networking concepts

Compute assigns a private IP address to each VM instance. Compute makes a distinction between fixed IPs and floating IP. Fixed IPs are IP addresses that are assigned to an instance on creation and stay the same until the instance is explicitly terminated. Floating IPs are addresses that can be dynamically associated with an instance. A floating IP address can be disassociated and associated with another instance at any time. A user can reserve a floating IP for their project.

> **Note**
>
> Currently, Compute with `nova-network` only supports Linux bridge networking that allows virtual interfaces to connect to the outside network through the physical interface.

The network controller with `nova-network` provides virtual networks to enable compute servers to interact with each other and with the public network. Compute with `nova-network` supports the following network modes, which are implemented as Network Manager types:

Flat Network Manager

In this mode, a network administrator specifies a subnet. IP addresses for VM instances are assigned from the subnet, and then injected into the image on launch. Each instance receives a fixed IP address from the pool of available addresses. A system administrator must create the Linux networking bridge (typically named `br100`, although this is configurable) on the systems running the `nova-network` service. All instances of the system are attached to the same bridge, which is configured manually by the network administrator.

> **Note**
>
> Configuration injection currently only works on Linux-style systems that keep networking configuration in `/etc/network/interfaces`.

Flat DHCP Network Manager

In this mode, OpenStack starts a DHCP server (dnsmasq) to allocate IP addresses to VM instances from the specified subnet, in addition to manually configuring the networking bridge. IP addresses for VM instances are assigned from a subnet specified by the network administrator.

Like flat mode, all instances are attached to a single bridge on the compute node. Additionally, a DHCP server configures instances depending on single-/multi-host mode, alongside each `nova-network`. In this mode, Compute does a bit more configuration. It attempts to bridge into an Ethernet device (`flat_interface`, eth0 by default). For every instance, Compute allocates a fixed IP address and configures dnsmasq with the MAC ID and IP address for the VM. Dnsmasq does not take part in the IP address allocation process, it only hands out IPs according to the mapping done by Compute. Instances receive their fixed IPs with the **dhcpdiscover** command. These IPs are not assigned to any of the host's network interfaces, only to the guest-side interface for the VM.

In any setup with flat networking, the hosts providing the `nova-network` service are responsible for forwarding traffic from the private network. They also run and configure dnsmasq as a DHCP server listening on this bridge, usually on IP address 10.0.0.1 (see *DHCP server: dnsmasq*). Compute can determine the NAT entries for each network, although sometimes NAT is not used, such as when the network has been configured with all public IPs, or if a hardware router is used (which is a high availability option). In

this case, hosts need to have `br100` configured and physically connected to any other nodes that are hosting VMs. You must set the `flat_network_bridge` option or create networks with the bridge parameter in order to avoid raising an error. Compute nodes have iptables or ebtables entries created for each project and instance to protect against MAC ID or IP address spoofing and ARP poisoning.

> **Note**
>
> In single-host Flat DHCP mode you will be able to ping VMs through their fixed IP from the `nova-network` node, but you cannot ping them from the compute nodes. This is expected behavior.

VLAN Network Manager

This is the default mode for OpenStack Compute. In this mode, Compute creates a VLAN and bridge for each tenant. For multiple-machine installations, the VLAN Network Mode requires a switch that supports VLAN tagging (IEEE 802.1Q). The tenant gets a range of private IPs that are only accessible from inside the VLAN. In order for a user to access the instances in their tenant, a special VPN instance (code named `cloudpipe`) needs to be created. Compute generates a certificate and key for the user to access the VPN and starts the VPN automatically. It provides a private network segment for each tenant's instances that can be accessed through a dedicated VPN connection from the internet. In this mode, each tenant gets its own VLAN, Linux networking bridge, and subnet.

The subnets are specified by the network administrator, and are assigned dynamically to a tenant when required. A DHCP server is started for each VLAN to pass out IP addresses to VM instances from the subnet assigned to the tenant. All instances belonging to one tenant are bridged into the same VLAN for that tenant. OpenStack Compute creates the Linux networking bridges and VLANs when required.

These network managers can co-exist in a cloud system. However, because you cannot select the type of network for a given tenant, you cannot configure multiple network types in a single Compute installation.

All network managers configure the network using network drivers. For example, the Linux L3 driver (`l3.py` and `linux_net.py`), which makes use of `iptables`, `route` and other network management facilities, and the libvirt network filtering facilities. The driver is not tied to any particular network manager; all network managers use the same driver. The driver usually initializes only when the first VM lands on this host node.

All network managers operate in either single-host or multi-host mode. This choice greatly influences the network configuration. In single-host mode, a single `nova-network` service provides a default gateway for VMs and hosts a single DHCP server (dnsmasq). In multi-host mode, each compute node runs its own `nova-network` service. In both cases, all

traffic between VMs and the internet flows through `nova-network`. Each mode has benefits and drawbacks. For more on this, see the Network Topology section in the OpenStack Operations Guide.

All networking options require network connectivity to be already set up between OpenStack physical nodes. OpenStack does not configure any physical network interfaces. All network managers automatically create VM virtual interfaces. Some network managers can also create network bridges such as `br100`.

The internal network interface is used for communication with VMs. The interface should not have an IP address attached to it before OpenStack installation, it serves only as a fabric where the actual endpoints are VMs and dnsmasq. Additionally, the internal network interface must be in `promiscuous` mode, so that it can receive packets whose target MAC address is the guest VM, not the host.

All machines must have a public and internal network interface (controlled by these options: `public_interface` for the public interface, and `flat_interface` and `vlan_interface` for the internal interface with flat or VLAN managers). This guide refers to the public network as the external network and the private network as the internal or tenant network.

For flat and flat DHCP modes, use the **nova network-create** command to create a network:

```
$ nova network-create vmnet \
  --fixed-range-v4 10.0.0.0/16 --fixed-cidr 10.0.20.0/24 --bridge br100
```

This example uses the following parameters:

`--fixed-range-v4` specifies the network subnet.

`--fixed-cidr` specifies a range of fixed IP addresses to allocate, and can be a subset of the `--fixed-range-v4` argument.

`--bridge` specifies the bridge device to which this network is connected on every compute node.

DHCP server: dnsmasq

The Compute service uses dnsmasq as the DHCP server when using either Flat DHCP Network Manager or VLAN Network Manager. For Compute to operate in IPv4/IPv6 dual-stack mode, use at least dnsmasq v2.63. The `nova-network` service is responsible for starting dnsmasq processes.

The behavior of dnsmasq can be customized by creating a dnsmasq configuration file. Specify the configuration file using the `dnsmasq_config_file` configuration option:

```
dnsmasq_config_file=/etc/dnsmasq-nova.conf
```

For more information about creating a dnsmasq configuration file, see the OpenStack Configuration Reference, and the dnsmasq documentation.

Dnsmasq also acts as a caching DNS server for instances. You can specify the DNS server that dnsmasq uses by setting the `dns_server` configuration option in `/etc/nova/nova.conf`. This example configures dnsmasq to use Google's public DNS server:

```
dns_server=8.8.8.8
```

Dnsmasq logs to syslog (typically `/var/log/syslog` or `/var/log/messages`, depending on Linux distribution). Logs can be useful for troubleshooting, especially in a situation where VM instances boot successfully but are not reachable over the network.

Administrators can specify the starting point IP address to reserve with the DHCP server (in the format n.n.n.n) with this command:

```
$ nova-manage fixed reserve --address IP_ADDRESS
```

This reservation only affects which IP address the VMs start at, not the fixed IP addresses that `nova-network` places on the bridges.

Configure Compute to use IPv6 addresses

If you are using OpenStack Compute with `nova-network`, you can put Compute into dual-stack mode, so that it uses both IPv4 and IPv6 addresses for communication. In dual-stack mode, instances can acquire their IPv6 global unicast addresses by using a stateless address auto-configuration mechanism [RFC 4862/2462]. IPv4/IPv6 dual-stack mode works with both `VlanManager` and `FlatDHCPManager` networking modes.

In `VlanManager` networking mode, each project uses a different 64-bit global routing prefix. In `FlatDHCPManager` mode, all instances use one 64-bit global routing prefix.

This configuration was tested with virtual machine images that have an IPv6 stateless address auto-configuration capability. This capability is required for any VM to run with an IPv6 address. You must use an EUI-64 address for stateless address auto-configuration. Each node that executes a `nova-*` service must have `python-netaddr` and `radvd` installed.

Switch into IPv4/IPv6 dual-stack mode

1. For every node running a `nova-*` service, install python-netaddr:

   ```
   # apt-get install python-netaddr
   ```

2. For every node running `nova-network`, install `radvd` and configure IPv6 networking:

```
# apt-get install radvd
# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
# echo 0 > /proc/sys/net/ipv6/conf/all/accept_ra
```

3. On all nodes, edit the `nova.conf` file and specify `use_ipv6 = True`.
4. Restart all `nova-*` services.

IPv6 configuration options

You can use the following options with the **nova network-create** command:

- Add a fixed range for IPv6 addresses to the **nova network-create** command. Specify `public` or `private` after the `network-create` parameter.

  ```
  $ nova network-create public --fixed-range-v4 FIXED_RANGE_V4 \
    --vlan VLAN_ID --vpn VPN_START --fixed-range-v6 FIXED_RANGE_V6
  ```

- Set the IPv6 global routing prefix by using the `--fixed_range_v6` parameter. The default value for the parameter is `fd00::/48`.

  When you use `FlatDHCPManager`, the command uses the original `--fixed_range_v6` value. For example:

  ```
  $ nova network-create public  --fixed-range-v4 10.0.2.0/24 \
    --fixed-range-v6 fd00:1::/48
  ```

- When you use `VlanManager`, the command increments the subnet ID to create subnet prefixes. Guest VMs use this prefix to generate their IPv6 global unicast addresses. For example:

  ```
  $ nova network-create public --fixed-range-v4 10.0.1.0/24 --vlan 100 \
    --vpn 1000 --fixed-range-v6 fd00:1::/48
  ```

Description of IPv6 configuration options

Configuration option = Default value	Description
[DEFAULT]	
fixed_range_v6 = fd00::/48	(StrOpt) Fixed IPv6 address block
gateway_v6 = None	(StrOpt) Default IPv6 gateway
ipv6_backend = rfc2462	(StrOpt) Backend to use for IPv6 generation
use_ipv6 = False	(BoolOpt) Use IPv6

Metadata service

Compute uses a metadata service for virtual machine instances to retrieve instance-specific data. Instances access the metadata service at `http://169.254.169.254`. The metadata service supports two sets of APIs: an OpenStack metadata API and an EC2-compatible API. Both APIs are versioned by date.

To retrieve a list of supported versions for the OpenStack metadata API, make a GET request to `http://169.254.169.254/openstack`:

```
$ curl http://169.254.169.254/openstack
2012-08-10
2013-04-04
2013-10-17
latest
```

To list supported versions for the EC2-compatible metadata API, make a GET request to `http://169.254.169.254`:

```
$ curl http://169.254.169.254
1.0
2007-01-19
2007-03-01
2007-08-29
2007-10-10
2007-12-15
2008-02-01
2008-09-01
2009-04-04
latest
```

If you write a consumer for one of these APIs, always attempt to access the most recent API version supported by your consumer first, then fall back to an earlier version if the most recent one is not available.

Metadata from the OpenStack API is distributed in JSON format. To retrieve the metadata, make a GET request to `http://169.254.169.254/openstack/2012-08-10/meta_data.json`:

```
$ curl http://169.254.169.254/openstack/2012-08-10/meta_data.json
```


```
{
   "uuid": "d8e02d56-2648-49a3-bf97-6be8f1204f38",
   "availability_zone": "nova",
   "hostname": "test.novalocal",
   "launch_index": 0,
```

```
    "meta": {
        "priority": "low",
        "role": "webserver"
    },
    "project_id": "f7ac731cc11f40efbc03a9f9e1d1d21f",
    "public_keys": {
        "mykey": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQDYVEprvtYJXVOBN0XNKV\
                  VRNCRX6BlnNbI+USLGais1sUWPwtSg7z9K9vhbYAPUZcq8c/s5S9dg5vTH\
                  bsiyPCIDOKyeHba4MUJq8Oh5b2i71/3BISpyxTBH/uZDHdslW2a+SrPDCe\
                  uMMoss9NFhBdKtDkdG9zyi0ibmCP6yMdEX8Q== Generated by Nova\n"
    },
    "name": "test"
}
```

Instances also retrieve user data (passed as the `user_data` parameter in the API call or by the *--user_data* flag in the **nova boot** command) through the metadata service, by making a GET request to `http://169.254.169.254/openstack/2012-08-10/user_data`:

```
$ curl http://169.254.169.254/openstack/2012-08-10/user_data
#!/bin/bash
echo 'Extra user data here'
```

The metadata service has an API that is compatible with version 2009-04-04 of the Amazon EC2 metadata service. This means that virtual machine images designed for EC2 will work properly with OpenStack.

The EC2 API exposes a separate URL for each metadata element. Retrieve a listing of these elements by making a GET query to `http://169.254.169.254/2009-04-04/meta-data/`:

```
$ curl http://169.254.169.254/2009-04-04/meta-data/
ami-id
ami-launch-index
ami-manifest-path
block-device-mapping/
hostname
instance-action
instance-id
instance-type
kernel-id
local-hostname
local-ipv4
placement/
public-hostname
public-ipv4
public-keys/
```

```
ramdisk-id
reservation-id
security-groups

$ curl http://169.254.169.254/2009-04-04/meta-data/block-device-mapping/
ami

$ curl http://169.254.169.254/2009-04-04/meta-data/placement/
availability-zone

$ curl http://169.254.169.254/2009-04-04/meta-data/public-keys/
0=mykey
```

Instances can retrieve the public SSH key (identified by keypair name when a user requests a new instance) by making a GET request to `http://169.254.169.254/2009-04-04/meta-data/public-keys/0/openssh-key`:

```
$ curl http://169.254.169.254/2009-04-04/meta-data/public-keys/0/openssh-key
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQDYVEprvtYJXVOBN0XNKVVRNCRX6BlnNbI+US\
LGais1sUWPwtSg7z9K9vhbYAPUZcq8c/s5S9dg5vTHbsiyPCIDOKyeHba4MUJq8Oh5b2i71/3B\
ISpyxTBH/uZDHdslW2a+SrPDCeuMMoss9NFhBdKtDkdG9zyi0ibmCP6yMdEX8Q== Generated\
by Nova
```

Instances can retrieve user data by making a GET request to `http://169.254.169.254/2009-04-04/user-data`:

```
$ curl http://169.254.169.254/2009-04-04/user-data
#!/bin/bash
echo 'Extra user data here'
```

The metadata service is implemented by either the `nova-api` service or the `nova-api-metadata` service. Note that the `nova-api-metadata` service is generally only used when running in multi-host mode, as it retrieves instance-specific metadata. If you are running the `nova-api` service, you must have `metadata` as one of the elements listed in the `enabled_apis` configuration option in `/etc/nova/nova.conf`. The default `enabled_apis` configuration setting includes the metadata service, so you do not need to modify it.

Hosts access the service at `169.254.169.254:80`, and this is translated to `metadata_host:metadata_port` by an iptables rule established by the `nova-network` service. In multi-host mode, you can set `metadata_host` to `127.0.0.1`.

For instances to reach the metadata service, the `nova-network` service must configure iptables to NAT port `80` of the `169.254.169.254` address to the IP address specified in `metadata_host` (this defaults to `$my_ip`, which is the IP address of the `nova-network` service) and port specified in `metadata_port` (which defaults to `8775`) in

`/etc/nova/nova.conf`.

> **Note**
>
> The `metadata_host` configuration option must be an IP address, not a host name.

The default Compute service settings assume that `nova-network` and `nova-api` are running on the same host. If this is not the case, in the `/etc/nova/nova.conf` file on the host running `nova-network`, set the `metadata_host` configuration option to the IP address of the host where `nova-api` is running.

Description of metadata configuration options

Configuration option = Default value	Description
[DEFAULT]	
metadata_cache_expiration = 15	(IntOpt) Time in seconds to cache metadata; 0 to disable metadata caching entirely (not recommended). Increasing this should improve response times of the metadata API when under heavy load. Higher values may increase memory usage and result in longer times for host metadata changes to take effect.
metadata_host = $my_ip	(StrOpt) The IP address for the metadata API server
metadata_listen = 0.0.0.0	(StrOpt) The IP address on which the metadata API will listen.

Configuration option = Default value	Description
[DEFAULT]	
metadata_listen_port = 8775	(IntOpt) The port on which the metadata API will listen.
metadata_manager = nova.api.manager.MetadataManager	(StrOpt) OpenStack metadata service manager
metadata_port = 8775	(IntOpt) The port for the metadata API port
metadata_workers = None	(IntOpt) Number of workers for metadata service. The default will be the number of CPUs available.
vendordata_driver = nova.api.metadata.vendordata_json.JsonFileVendorData	(StrOpt) Driver to use for vendor data
vendordata_jsonfile_path = None	(StrOpt) File to load JSON formatted vendor data from

Enable ping and SSH on VMs

You need to enable `ping` and `ssh` on your VMs for network access. This can be done with either the **nova** or **euca2ools** commands.

> **Note**
>
> Run these commands as root only if the credentials used to interact with `nova-api` are in `/root/.bashrc`. If the EC2 credentials in the `.bashrc` file are for an unprivileged user, you must run these commands as that user instead.

Enable ping and SSH with **nova** commands:

```
$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
```

Enable ping and SSH with `euca2ools`:

```
$ euca-authorize -P icmp -t -1:-1 -s 0.0.0.0/0 default
$ euca-authorize -P tcp -p 22 -s 0.0.0.0/0 default
```

If you have run these commands and still cannot ping or SSH your instances, check the number of running `dnsmasq` processes, there should be two. If not, kill the processes and restart the service with these commands:

```
# killall dnsmasq
# service nova-network restart
```

Configure public (floating) IP addresses

This section describes how to configure floating IP addresses with `nova-network`. For information about doing this with OpenStack Networking, see *L3 routing and NAT*.

Private and public IP addresses

In this section, the term floating IP address is used to refer to an IP address, usually public, that you can dynamically add to a running virtual instance.

Every virtual instance is automatically assigned a private IP address. You can choose to assign a public (or floating) IP address instead. OpenStack Compute uses network address translation (NAT) to assign floating IPs to virtual instances.

To be able to assign a floating IP address, edit the `/etc/nova/nova.conf` file to specify which interface the `nova-network` service should bind public IP addresses to:

```
public_interface=VLAN100
```

If you make changes to the `/etc/nova/nova.conf` file while the `nova-network` service is running, you will need to restart the service to pick up the changes.

> **Note**
>
> Floating IPs are implemented by using a source NAT (SNAT rule in iptables), so security groups can sometimes display inconsistent behavior if VMs use their floating IP to communicate with other VMs, particularly on the same physical host. Traffic from VM to VM across the fixed network does not have this issue, and so this is the recommended setup. To ensure that traffic does not get SNATed to the floating range, explicitly set:
>
> ```
> dmz_cidr=x.x.x.x/y
> ```

The `x.x.x.x/y` value specifies the range of floating IPs for each pool of floating IPs that you define. This configuration is also required if the VMs in the source group have floating IPs.

Enable IP forwarding

IP forwarding is disabled by default on most Linux distributions. You will need to enable it in order to use floating IPs.

> **Note**
>
> IP forwarding only needs to be enabled on the nodes that run `nova-network`. However, you will need to enable it on all compute nodes if you use `multi_host` mode.

To check if IP forwarding is enabled, run:

```
$ cat /proc/sys/net/ipv4/ip_forward
0
```

Alternatively, run:

```
$ sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
```

In these examples, IP forwarding is disabled.

To enable IP forwarding dynamically, run:

```
# sysctl -w net.ipv4.ip_forward=1
```

Alternatively, run:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

To make the changes permanent, edit the `/etc/sysctl.conf` file and update the IP forwarding setting:

```
net.ipv4.ip_forward = 1
```

Save the file and run this command to apply the changes:

```
# sysctl -p
```

You can also apply the changes by restarting the network service:

- on Ubuntu, Debian:

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

- on RHEL, Fedora, CentOS, openSUSE and SLES:

```
# service network restart
```

Create a list of available floating IP addresses

Compute maintains a list of floating IP addresses that are available for assigning to instances. Use the **nova-manage floating** commands to perform floating IP operations:

- Add entries to the list:

```
# nova-manage floating create --pool nova --ip_range 68.99.26.170/31
```

- List the floating IP addresses in the pool:

```
# nova-manage floating list
```

- Create specific floating IPs for either a single address or a subnet:

```
# nova-manage floating create --pool POOL_NAME --ip_range CIDR
```

- Remove floating IP addresses using the same parameters as the create command:

```
# nova-manage floating delete CIDR
```

For more information about how administrators can associate floating IPs with instances, see Manage IP addresses in the OpenStack Administrator Guide.

Automatically add floating IPs

You can configure `nova-network` to automatically allocate and assign a floating IP address to virtual instances when they are launched. Add this line to the `/etc/nova/nova.conf` file:

```
auto_assign_floating_ip=True
```

Save the file, and restart `nova-network`

> **Note**
>
> If this option is enabled, but all floating IP addresses have already been allocated, the **nova boot** command will fail.

Remove a network from a project

You cannot delete a network that has been associated to a project. This section describes the procedure for dissociating it so that it can be deleted.

In order to disassociate the network, you will need the ID of the project it has been associated to. To get the project ID, you will need to be an administrator.

Disassociate the network from the project using the **scrub** command, with the project ID as the final parameter:

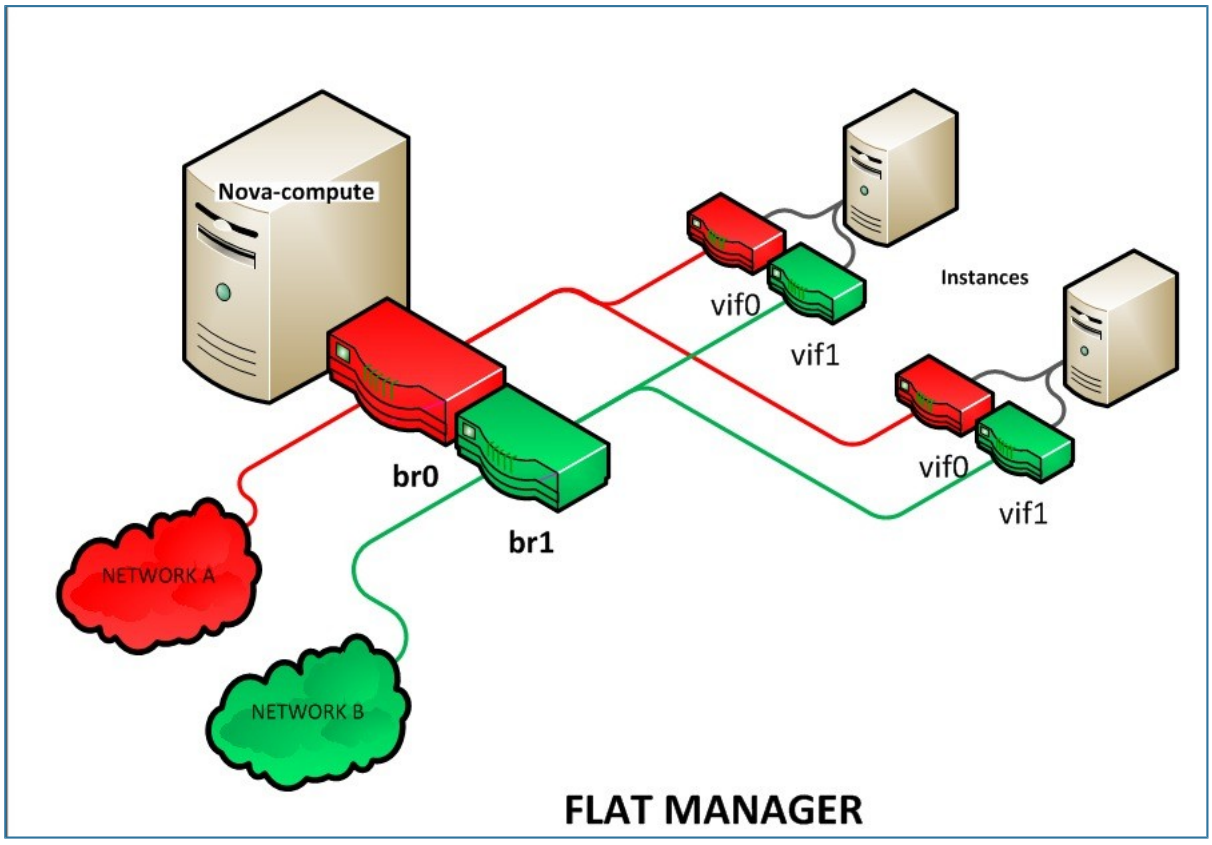
```
# nova-manage project scrub --project ID
```

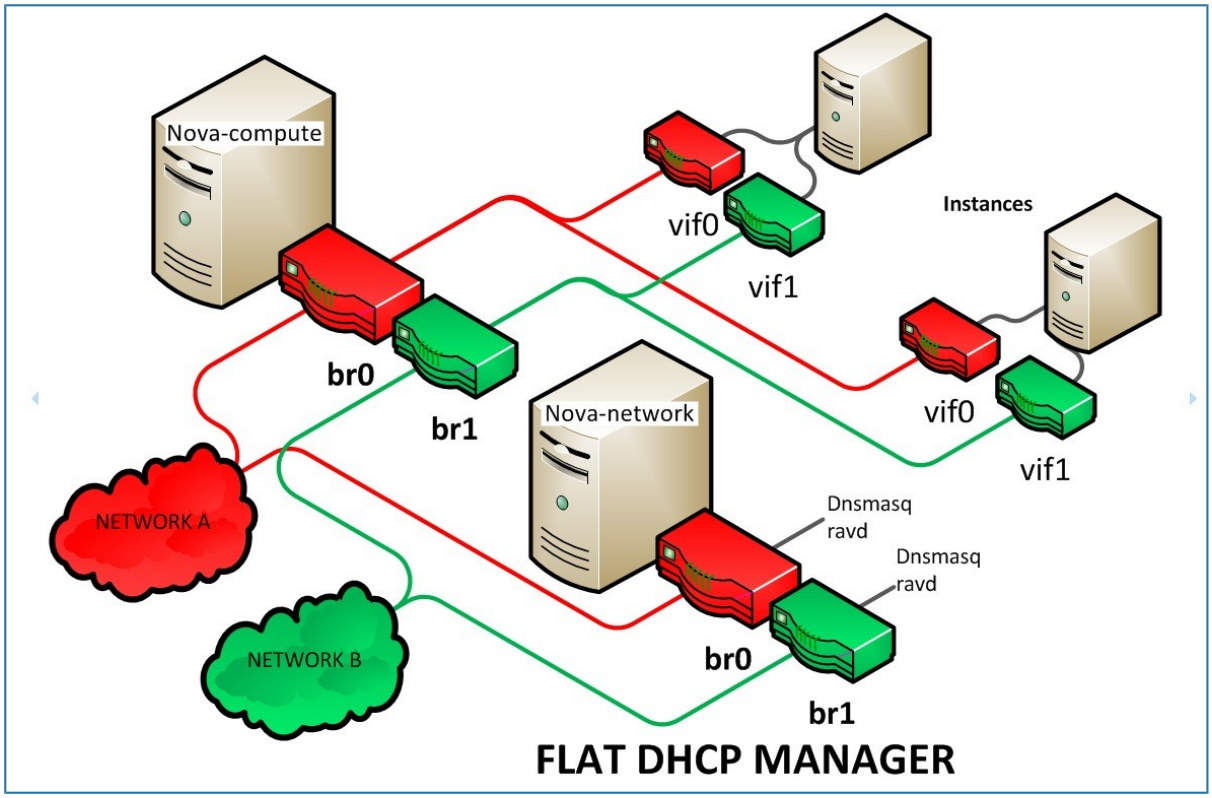
Multiple interfaces for instances (multinic)

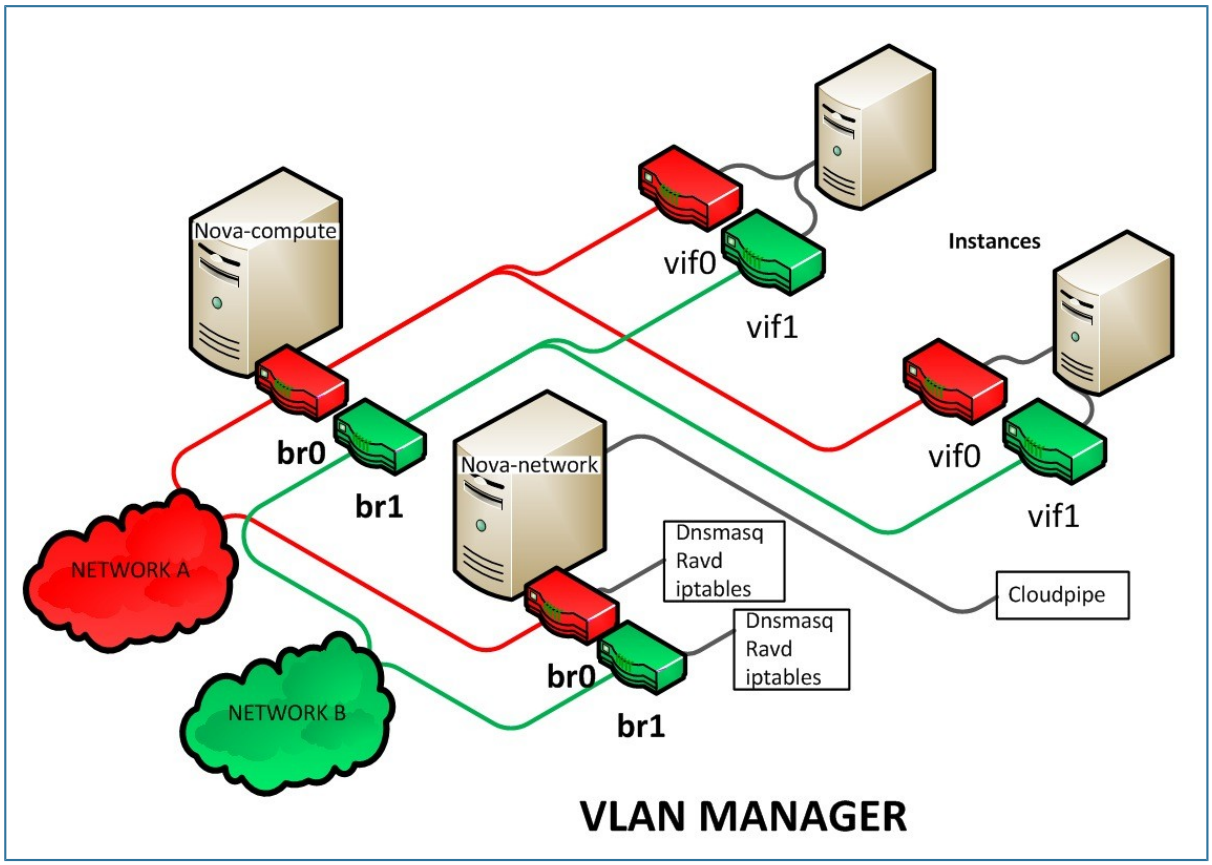
The multinic feature allows you to use more than one interface with your instances. This is useful in several scenarios:

- SSL Configurations (VIPs)
- Services failover/HA
- Bandwidth Allocation
- Administrative/Public access to your instances

Each VIP represents a separate network with its own IP block. Every network mode has its own set of changes regarding multinic usage:







Using multinic

In order to use multinic, create two networks, and attach them to the tenant (named `project` on the command line):

```
$ nova network-create first-net --fixed-range-v4 20.20.0.0/24 --project-id
$your-project
$ nova network-create second-net --fixed-range-v4 20.20.10.0/24 --project-id
$your-project
```

Each new instance will now receive two IP addresses from their respective DHCP servers:

```
$ nova list
```

```
+-----+------------+--------+----------------------------------------+
|  ID |    Name    | Status |                Networks                |
+-----+------------+--------+----------------------------------------+
| 124 | Server 124 | ACTIVE | network2=20.20.0.3; private=20.20.10.14|
+-----+------------+--------+----------------------------------------+
```

> **Note**
>
> Make sure you start the second interface on the instance, or it won't be reachable through the second IP.

This example demonstrates how to set up the interfaces within the instance. This is the configuration that needs to be applied inside the image.

Edit the `/etc/network/interfaces` file:

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

auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet dhcp
```

If the Virtual Network Service Neutron is installed, you can specify the networks to attach to the interfaces by using the *--nic* flag with the **nova** command:

```
$ nova boot --image ed8b2a37-5535-4a5f-a615-443513036d71 --flavor 1 --nic net-
id=NETWORK1_ID --nic net-id=NETWORK2_ID test-vm1
```

Troubleshooting Networking

Cannot reach floating IPs

Problem

You cannot reach your instances through the floating IP address.

Solution

- Check that the default security group allows ICMP (ping) and SSH (port 22), so that you can reach the instances:

```
$ nova secgroup-list-rules default
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port |  IP Range | Source Group |
+-------------+-----------+---------+-----------+--------------+
| icmp        | -1        | -1      | 0.0.0.0/0 |              |
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

- Check the NAT rules have been added to iptables on the node that is running `nova-network`:

```
# iptables -L -nv -t nat
-A nova-network-PREROUTING -d 68.99.26.170/32 -j DNAT --to-destination 10.0.0.3
-A nova-network-floating-snat -s 10.0.0.3/32 -j SNAT --to-source 68.99.26.170
```

- Check that the public address (68.99.26.170 in this example), has been added to your public interface. You should see the address in the listing when you use the **ip addr** command:

```
$ ip addr
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
link/ether xx:xx:xx:17:4b:c2 brd ff:ff:ff:ff:ff:ff
inet 13.22.194.80/24 brd 13.22.194.255 scope global eth0
inet 68.99.26.170/32 scope global eth0
inet6 fe80::82b:2bf:fe1:4b2/64 scope link
valid_lft forever preferred_lft forever
```

> **Note**
>
> You cannot use SSH to access an instance with a public IP from within the same server because the routing configuration does not allow it.

- Use `tcpdump` to identify if packets are being routed to the inbound interface on the compute host. If the packets are reaching the compute hosts but the connection is failing, the issue may be that the packet is being dropped by reverse path filtering. Try disabling reverse-path filtering on the inbound interface. For example, if the inbound interface is `eth2`, run:

  ```
  # sysctl -w net.ipv4.conf.ETH2.rp_filter=0
  ```

  If this solves the problem, add the following line to `/etc/sysctl.conf` so that the reverse-path filter is persistent:

  ```
  net.ipv4.conf.rp_filter=0
  ```

Temporarily disable firewall

Problem

Networking issues prevent administrators accessing or reaching VM's through various pathways.

Solution

You can disable the firewall by setting this option in `/etc/nova/nova.conf`:

```
firewall_driver=nova.virt.firewall.NoopFirewallDriver
```

Packet loss from instances to nova-network server (VLANManager mode)

Problem

If you can access your instances with `SSH` but the network to your instance is slow, or if you find that running certain operations are slower than they should be (for example, `sudo`), packet loss could be occurring on the connection to the instance.

Packet loss can be caused by Linux networking configuration settings related to bridges. Certain settings can cause packets to be dropped between the VLAN interface (for example, `vlan100`) and the associated bridge interface (for example, `br100`) on the host running `nova-network`.

Solution

One way to check whether this is the problem is to open three terminals and run the following commands:

1. In the first terminal, on the host running `nova-network`, use `tcpdump` on the VLAN

interface to monitor DNS-related traffic (UDP, port 53). As root, run:

```
# tcpdump -K -p -i vlan100 -v -vv udp port 53
```

2. In the second terminal, also on the host running `nova-network`, use `tcpdump` to monitor DNS-related traffic on the bridge interface. As root, run:

   ```
   # tcpdump -K -p -i br100 -v -vv udp port 53
   ```

3. In the third terminal, use `SSH` to access the instance and generate DNS requests by using the **nslookup** command:

   ```
   $ nslookup www.google.com
   ```

   The symptoms may be intermittent, so try running **nslookup** multiple times. If the network configuration is correct, the command should return immediately each time. If it is not correct, the command hangs for several seconds before returning.

4. If the **nslookup** command sometimes hangs, and there are packets that appear in the first terminal but not the second, then the problem may be due to filtering done on the bridges. Try disabling filtering, and running these commands as root:

   ```
   # sysctl -w net.bridge.bridge-nf-call-arptables=0
   # sysctl -w net.bridge.bridge-nf-call-iptables=0
   # sysctl -w net.bridge.bridge-nf-call-ip6tables=0
   ```

   If this solves your issue, add the following line to `/etc/sysctl.conf` so that the changes are persistent:

   ```
   net.bridge.bridge-nf-call-arptables=0
   net.bridge.bridge-nf-call-iptables=0
   net.bridge.bridge-nf-call-ip6tables=0
   ```

KVM: Network connectivity works initially, then fails

Problem

With KVM hypervisors, instances running Ubuntu 12.04 sometimes lose network connectivity after functioning properly for a period of time.

Solution

Try loading the `vhost_net` kernel module as a workaround for this issue (see bug #997978) . This kernel module may also improve network performance on KVM. To load the kernel module:

```
# modprobe vhost_net
```

> **Note**
>
> Loading the module has no effect on running instances.

System administration

To effectively administer Compute, you must understand how the different installed nodes interact with each other. Compute can be installed in many different ways using multiple servers, but generally multiple compute nodes control the virtual servers and a cloud controller node contains the remaining Compute services.

The Compute cloud works using a series of daemon processes named `nova-*` that exist persistently on the host machine. These binaries can all run on the same machine or be spread out on multiple boxes in a large deployment. The responsibilities of services and drivers are:

Services

`nova-api`
receives XML requests and sends them to the rest of the system. A WSGI app routes and authenticates requests. Supports the EC2 and OpenStack APIs. A `nova.conf` configuration file is created when Compute is installed.

`nova-cert`
manages certificates.

`nova-compute`
manages virtual machines. Loads a Service object, and exposes the public methods on ComputeManager through a Remote Procedure Call (RPC).

`nova-conductor`
provides database-access support for Compute nodes (thereby reducing security risks).

`nova-consoleauth`
manages console authentication.

`nova-objectstore`
a simple file-based storage system for images that replicates most of the S3 API. It can be replaced with OpenStack Image service and either a simple image manager or OpenStack Object Storage as the virtual machine image storage facility. It must exist on the same node as `nova-compute`.

`nova-network`
manages floating and fixed IPs, DHCP, bridging and VLANs. Loads a Service object which exposes the public methods on one of the subclasses of NetworkManager. Different networking strategies are available by changing the `network_manager` configuration option to `FlatManager`, `FlatDHCPManager`, or `VLANManager` (defaults to

`VLANManager` if nothing is specified).

`nova-scheduler`

dispatches requests for new virtual machines to the correct node.

`nova-novncproxy`

provides a VNC proxy for browsers, allowing VNC consoles to access virtual machines.

> **Note**
>
> Some services have drivers that change how the service implements its core functionality. For example, the `nova-compute` service supports drivers that let you choose which hypervisor type it can use. `nova-network` and `nova-scheduler` also have drivers.

Manage Compute users

Access to the Euca2ools (ec2) API is controlled by an access key and a secret key. The user's access key needs to be included in the request, and the request must be signed with the secret key. Upon receipt of API requests, Compute verifies the signature and runs commands on behalf of the user.

To begin using Compute, you must create a user with the Identity service.

Manage volumes

Depending on the setup of your cloud provider, they may give you an endpoint to use to manage volumes, or there may be an extension under the covers. In either case, you can use the `openstack` CLI to manage volumes.

openstack volume commands

Command	Description
server add volume	Attach a volume to a server.
volume create	Add a new volume.
volume delete	Remove or delete a volume.
server remove volume	Detach or remove a volume from a server.
volume list	List all the volumes.
volume show	Show details about a volume.
snapshot create	Add a new snapshot.
snapshot delete	Remove a snapshot.

Command	Description
snapshot list	List all the snapshots.
snapshot show	Show details about a snapshot.
volume type create	Create a new volume type.
volume type delete	Delete a specific flavor
volume type list	Print a list of available ‘volume types’.

For example, to list IDs and names of volumes, run:

```
$ openstack volume list
+--------+--------------+-----------+------+-------------+
| ID     | Display Name | Status    | Size | Attached to |
+--------+--------------+-----------+------+-------------+
| 86e6cb | testnfs      | available |    1 |             |
| e389f7 | demo         | available |    1 |             |
+--------+--------------+-----------+------+-------------+
```

Flavors

Admin users can use the **openstack flavor** command to customize and manage flavors. To see information for this command, run:

```
$ openstack flavor --help
Command "flavor" matches:
  flavor create
  flavor delete
  flavor list
  flavor set
  flavor show
  flavor unset
```

> **Note**
>
> • Configuration rights can be delegated to additional users by redefining the access controls for `compute_extension:flavormanage` in `/etc/nova/policy.json` on the `nova-api` server.
>
> • The Dashboard simulates the ability to modify a flavor by deleting an existing flavor and creating a new one with the same name.

Flavors define these elements:

Element	Description
Name	A descriptive name. XX.SIZE_NAME is typically not required, though some third party tools may rely on it.
Memory MB	Instance memory in megabytes.
Disk	Virtual root disk size in gigabytes. This is an ephemeral disk that the base image is copied into. When booting from a persistent volume it is not used. The “0” size is a special case which uses the native base image size as the size of the ephemeral root volume.
Ephemeral	Specifies the size of a secondary ephemeral data disk. This is an empty, unformatted disk and exists only for the life of the instance. Default value is `0`.
Swap	Optional swap space allocation for the instance. Default value is `0`.
VCPUs	Number of virtual CPUs presented to the instance.
RXTX Factor	Optional property allows created servers to have a different bandwidth cap than that defined in the network they are attached to. This factor is multiplied by the rxtx_base property of the network. Default value is `1.0`. That is, the same as attached network. This parameter is only available for Xen or NSX based systems.
Is Public	Boolean value, whether flavor is available to all users or private to the tenant it was created in. Defaults to `True`.
Extra Specs	Key and value pairs that define on which compute nodes a flavor can run. These pairs must match corresponding pairs on the compute nodes. Use to

Element	Description
	implement special resources, such as flavors that run on only compute nodes with GPU hardware.

> **Note**
>
> Flavor customization can be limited by the hypervisor in use. For example the libvirt driver enables quotas on CPUs available to a VM, disk tuning, bandwidth I/O, watchdog behavior, random number generator device control, and instance VIF traffic control.

Is Public

Flavors can be assigned to particular projects. By default, a flavor is public and available to all projects. Private flavors are only accessible to those on the access list and are invisible to other projects. To create and assign a private flavor to a project, run this command:

```
$ openstack flavor create --private p1.medium auto 512 40 4
```

Extra Specs

CPU limits

You can configure the CPU limits with control parameters with the nova client. For example, to configure the I/O limit, use:

```
$ openstack flavor set FLAVOR-NAME \
    --property quota:read_bytes_sec=10240000 \
    --property quota:write_bytes_sec=10240000
```

Use these optional parameters to control weight shares, enforcement intervals for runtime quotas, and a quota for maximum allowed bandwidth:

- `cpu_shares`: Specifies the proportional weighted share for the domain. If this element is omitted, the service defaults to the OS provided defaults. There is no unit for the value; it is a relative measure based on the setting of other VMs. For example, a VM configured with value 2048 gets twice as much CPU time as a VM configured with value 1024.
- `cpu_shares_level`: On VMware, specifies the allocation level. Can be `custom`, `high`, `normal`, or `low`. If you choose `custom`, set the number of shares using `cpu_shares_share`.
- `cpu_period`: Specifies the enforcement interval (unit: microseconds) for QEMU and LXC hypervisors. Within a period, each VCPU of the domain is not allowed to

consume more than the quota worth of runtime. The value should be in range `[1000, 1000000]`. A period with value 0 means no value.

- `cpu_limit`: Specifies the upper limit for VMware machine CPU allocation in MHz. This parameter ensures that a machine never uses more than the defined amount of CPU time. It can be used to enforce a limit on the machine's CPU performance.
- `cpu_reservation`: Specifies the guaranteed minimum CPU reservation in MHz for VMware. This means that if needed, the machine will definitely get allocated the reserved amount of CPU cycles.
- `cpu_quota`: Specifies the maximum allowed bandwidth (unit: microseconds). A domain with a negative-value quota indicates that the domain has infinite bandwidth, which means that it is not bandwidth controlled. The value should be in range `[1000, 18446744073709551]` or less than 0. A quota with value 0 means no value. You can use this feature to ensure that all vCPUs run at the same speed. For example:

  ```
  $ openstack flavor set FLAVOR-NAME \
      --property quota:cpu_quota=10000 \
      --property quota:cpu_period=20000
  ```

  In this example, an instance of `FLAVOR-NAME` can only consume a maximum of 50% CPU of a physical CPU computing capability.

Memory limits

For VMware, you can configure the memory limits with control parameters.

Use these optional parameters to limit the memory allocation, guarantee minimum memory reservation, and to specify shares used in case of resource contention:

- `memory_limit`: Specifies the upper limit for VMware machine memory allocation in MB. The utilization of a virtual machine will not exceed this limit, even if there are available resources. This is typically used to ensure a consistent performance of virtual machines independent of available resources.
- `memory_reservation`: Specifies the guaranteed minimum memory reservation in MB for VMware. This means the specified amount of memory will definitely be allocated to the machine.
- `memory_shares_level`: On VMware, specifies the allocation level. This can be `custom`, `high`, `normal` or `low`. If you choose `custom`, set the number of shares using `memory_shares_share`.
- `memory_shares_share`: Specifies the number of shares allocated in the event

that `custom` is used. There is no unit for this value. It is a relative measure based on the settings for other VMs. For example:

```
$ openstack flavor set FLAVOR-NAME \
    --property quota:memory_shares_level=custom \
    --property quota:memory_shares_share=15
```

Disk I/O limits

For VMware, you can configure the resource limits for disk with control parameters.

Use these optional parameters to limit the disk utilization, guarantee disk allocation, and to specify shares used in case of resource contention. This allows the VMware driver to enable disk allocations for the running instance.

- `disk_io_limit`: Specifies the upper limit for disk utilization in I/O per second. The utilization of a virtual machine will not exceed this limit, even if there are available resources. The default value is -1 which indicates unlimited usage.
- `disk_io_reservation`: Specifies the guaranteed minimum disk allocation in terms of *IOPS*.
- `disk_io_shares_level`: Specifies the allocation level. This can be `custom`, `high`, `normal` or `low`. If you choose custom, set the number of shares using `disk_io_shares_share`.
- `disk_io_shares_share`: Specifies the number of shares allocated in the event that `custom` is used. When there is resource contention, this value is used to determine the resource allocation.

  The example below sets the `disk_io_reservation` to 2000 IOPS.

  ```
  $ openstack flavor set FLAVOR-NAME \
      --property quota:disk_io_reservation=2000
  ```

Disk tuning

Using disk I/O quotas, you can set maximum disk write to 10 MB per second for a VM user. For example:

```
$ openstack flavor set FLAVOR-NAME \
    --property quota:disk_write_bytes_sec=10485760
```

The disk I/O options are:

- `disk_read_bytes_sec`
- `disk_read_iops_sec`

- `disk_write_bytes_sec`
- `disk_write_iops_sec`
- `disk_total_bytes_sec`
- `disk_total_iops_sec`

Bandwidth I/O

The vif I/O options are:

- `vif_inbound_average`
- `vif_inbound_burst`
- `vif_inbound_peak`
- `vif_outbound_average`
- `vif_outbound_burst`
- `vif_outbound_peak`

Incoming and outgoing traffic can be shaped independently. The bandwidth element can have at most, one inbound and at most, one outbound child element. If you leave any of these child elements out, no *quality of service (QoS)* is applied on that traffic direction. So, if you want to shape only the network's incoming traffic, use inbound only (and vice versa). Each element has one mandatory attribute average, which specifies the average bit rate on the interface being shaped.

There are also two optional attributes (integer): `peak`, which specifies the maximum rate at which a bridge can send data (kilobytes/second), and `burst`, the amount of bytes that can be burst at peak speed (kilobytes). The rate is shared equally within domains connected to the network.

The example below sets network traffic bandwidth limits for existing flavor as follows:

- Outbound traffic:
  - average: 256 Mbps (32768 kilobytes/second)
  - peak: 512 Mbps (65536 kilobytes/second)
  - burst: 65536 kilobytes
- Inbound traffic:
  - average: 256 Mbps (32768 kilobytes/second)
  - peak: 512 Mbps (65536 kilobytes/second)
  - burst: 65536 kilobytes

```
$ openstack flavor set FLAVOR-NAME \
    --property quota:vif_outbound_average=32768 \
    --property quota:vif_outbound_peak=65536 \
    --property quota:vif_outbound_burst=65536 \
    --property quota:vif_inbound_average=32768 \
```

```
    --property quota:vif_inbound_peak=65536 \
    --property quota:vif_inbound_burst=65536
```

> **Note**
>
> All the speed limit values in above example are specified in kilobytes/second. And burst values are in kilobytes.

Watchdog behavior

For the libvirt driver, you can enable and set the behavior of a virtual hardware watchdog device for each flavor. Watchdog devices keep an eye on the guest server, and carry out the configured action, if the server hangs. The watchdog uses the i6300esb device (emulating a PCI Intel 6300ESB). If `hw:watchdog_action` is not specified, the watchdog is disabled.

To set the behavior, use:

```
$ openstack flavor set FLAVOR-NAME --property hw:watchdog_action=ACTION
```

Valid ACTION values are:

- `disabled`: (default) The device is not attached.
- `reset`: Forcefully reset the guest.
- `poweroff`: Forcefully power off the guest.
- `pause`: Pause the guest.
- `none`: Only enable the watchdog; do nothing if the server hangs.

> **Note**
>
> Watchdog behavior set using a specific image's properties will override behavior set using flavors.

Random-number generator

If a random-number generator device has been added to the instance through its image properties, the device can be enabled and configured using:

```
$ openstack flavor set FLAVOR-NAME \
    --property hw_rng:allowed=True \
    --property hw_rng:rate_bytes=RATE-BYTES \
    --property hw_rng:rate_period=RATE-PERIOD
```

Where:

- RATE-BYTES: (integer) Allowed amount of bytes that the guest can read from the host's entropy per period.
- RATE-PERIOD: (integer) Duration of the read period in seconds.

CPU topology

For the libvirt driver, you can define the topology of the processors in the virtual machine using properties. The properties with `max` limit the number that can be selected by the user with image properties.

```
$ openstack flavor set FLAVOR-NAME \
    --property hw:cpu_sockets=FLAVOR-SOCKETS \
    --property hw:cpu_cores=FLAVOR-CORES \
    --property hw:cpu_threads=FLAVOR-THREADS \
    --property hw:cpu_max_sockets=FLAVOR-SOCKETS \
    --property hw:cpu_max_cores=FLAVOR-CORES \
    --property hw:cpu_max_threads=FLAVOR-THREADS
```

Where:

- FLAVOR-SOCKETS: (integer) The number of sockets for the guest VM. By default, this is set to the number of vCPUs requested.
- FLAVOR-CORES: (integer) The number of cores per socket for the guest VM. By default, this is set to `1`.
- FLAVOR-THREADS: (integer) The number of threads per core for the guest VM. By default, this is set to `1`.

CPU pinning policy

For the libvirt driver, you can pin the virtual CPUs (vCPUs) of instances to the host's physical CPU cores (pCPUs) using properties. You can further refine this by stating how hardware CPU threads in a simultaneous multithreading-based (SMT) architecture be used. These configurations will result in improved per-instance determinism and performance.

> **Note**
>
> SMT-based architectures include Intel processors with Hyper-Threading technology. In these architectures, processor cores share a number of components with one or more other cores. Cores in such architectures are commonly referred to as hardware threads, while the cores that a given core share components with are known as thread siblings.

> **Note**
>
> Host aggregates should be used to separate these pinned instances from unpinned instances as the latter will not respect the resourcing requirements of the former.

```
$ openstack flavor set FLAVOR-NAME \
    --property hw:cpu_policy=CPU-POLICY \
    --property hw:cpu_thread_policy=CPU-THREAD-POLICY
```

Valid CPU-POLICY values are:

- `shared`: (default) The guest vCPUs will be allowed to freely float across host pCPUs, albeit potentially constrained by NUMA policy.
- `dedicated`: The guest vCPUs will be strictly pinned to a set of host pCPUs. In the absence of an explicit vCPU topology request, the drivers typically expose all vCPUs as sockets with one core and one thread. When strict CPU pinning is in effect the guest CPU topology will be setup to match the topology of the CPUs to which it is pinned. This option implies an overcommit ratio of 1.0. For example, if a two vCPU guest is pinned to a single host core with two threads, then the guest will get a topology of one socket, one core, threads threads.

Valid CPU-THREAD-POLICY values are:

- `prefer`: (default) The host may or may not have an SMT architecture. Where an SMT architecture is present, thread siblings are preferred.
- `isolate`: The host must not have an SMT architecture or must emulate a non-SMT architecture. If the host does not have an SMT architecture, each vCPU is placed on a different core as expected. If the host does have an SMT architecture - that is, one or more cores have thread siblings - then each vCPU is placed on a different physical core. No vCPUs from other guests are placed on the same core. All but one thread sibling on each utilized core is therefore guaranteed to be unusable.
- `require`: The host must have an SMT architecture. Each vCPU is allocated on thread siblings. If the host does not have an SMT architecture, then it is not used. If the host has an SMT architecture, but not enough cores with free thread siblings are available, then scheduling fails.

> **Note**
>
> The `hw:cpu_thread_policy` option is only valid if `hw:cpu_policy` is set to `dedicated`.

NUMA topology

For the libvirt driver, you can define the host NUMA placement for the instance vCPU threads as well as the allocation of instance vCPUs and memory from the host NUMA nodes. For flavors whose memory and vCPU allocations are larger than the size of NUMA nodes in the compute hosts, the definition of a NUMA topology allows hosts to better utilize NUMA and improve performance of the instance OS.

```
$ openstack flavor set FLAVOR-NAME \
    --property hw:numa_nodes=FLAVOR-NODES \
    --property hw:numa_cpus.N=FLAVOR-CORES \
    --property hw:numa_mem.N=FLAVOR-MEMORY
```

Where:

- FLAVOR-NODES: (integer) The number of host NUMA nodes to restrict execution of instance vCPU threads to. If not specified, the vCPU threads can run on any number of the host NUMA nodes available.
- N: (integer) The instance NUMA node to apply a given CPU or memory configuration to, where N is in the range `0` to `FLAVOR-NODES - 1`.
- FLAVOR-CORES: (comma-separated list of integers) A list of instance vCPUs to map to instance NUMA node N. If not specified, vCPUs are evenly divided among available NUMA nodes.
- FLAVOR-MEMORY: (integer) The number of MB of instance memory to map to instance instance NUMA node N. If not specified, memory is evenly divided among available NUMA nodes.

> **Note**
>
> `hw:numa_cpus.N` and `hw:numa_mem.N` are only valid if `hw:numa_nodes` is set. Additionally, they are only required if the instance's NUMA nodes have an asymetrical allocation of CPUs and RAM (important for some NFV workloads).

> **Note**
>
> The `N` parameter is an index of *guest* NUMA nodes and may not correspond to *host* NUMA nodes. For example, on a platform with two NUMA nodes, the scheduler may opt to place guest NUMA node 0, as referenced in `hw:numa_mem.0` on host NUMA node 1 and vice versa. Similarly, the integers used for `FLAVOR-CORES` are indexes of *guest* vCPUs and may not correspond to *host* CPUs. As such, this feature cannot be used to constrain instances to specific host CPUs or NUMA nodes.

> **Warning**
>
> If the combined values of `hw:numa_cpus.N` or `hw:numa_mem.N` are greater than the available number of CPUs or memory respectively, an exception is raised.

Large pages allocation

You can configure the size of large pages used to back the VMs.

```
$ openstack flavor set FLAVOR-NAME \
    --property hw:mem_page_size=PAGE_SIZE
```

Valid `PAGE_SIZE` values are:

- `small`: (default) The smallest page size is used. Example: 4 KB on x86.
- `large`: Only use larger page sizes for guest RAM. Example: either 2 MB or 1 GB on x86.
- `any`: It is left up to the compute driver to decide. In this case, the libvirt driver might try to find large pages, but fall back to small pages. Other drivers may choose alternate policies for any.
- pagesize: (string) An explicit page size can be set if the workload has specific requirements. This value can be an integer value for the page size in KB, or can use any standard suffix. Example: `4KB`, `2MB`, `2048`, `1GB`.

> **Note**
>
> Large pages can be enabled for guest RAM without any regard to whether the guest OS will use them or not. If the guest OS chooses not to use huge pages, it will merely see small pages as before. Conversely, if a guest OS does intend to use huge pages, it is very important that the guest RAM be backed by huge pages. Otherwise, the guest OS will not be getting the performance benefit it is expecting.

PCI passthrough

You can assign PCI devices to a guest by specifying them in the flavor.

```
$ openstack flavor set FLAVOR-NAME \
    --property pci_passthrough:alias=ALIAS:COUNT
```

Where:

- ALIAS: (string) The alias which correspond to a particular PCI device class as configured in the nova configuration file (see nova.conf configuration options).

- COUNT: (integer) The amount of PCI devices of type ALIAS to be assigned to a guest.

Compute service node firewall requirements

Console connections for virtual machines, whether direct or through a proxy, are received on ports `5900` to `5999`. The firewall on each Compute service node must allow network traffic on these ports.

This procedure modifies the iptables firewall to allow incoming connections to the Compute services.

Configuring the service-node firewall

1. Log in to the server that hosts the Compute service, as root.
2. Edit the `/etc/sysconfig/iptables` file, to add an INPUT rule that allows TCP traffic on ports from `5900` to `5999`. Make sure the new rule appears before any INPUT rules that REJECT traffic:

   ```
   -A INPUT -p tcp -m multiport --dports 5900:5999 -j ACCEPT
   ```

3. Save the changes to the `/etc/sysconfig/iptables` file, and restart the `iptables` service to pick up the changes:

   ```
   $ service iptables restart
   ```

4. Repeat this process for each Compute service node.

Injecting the administrator password

Compute can generate a random administrator (root) password and inject that password into an instance. If this feature is enabled, users can run **ssh** to an instance without an **ssh** keypair. The random password appears in the output of the **nova boot** command. You can also view and set the admin password from the dashboard.

Password injection using the dashboard

By default, the dashboard will display the `admin` password and allow the user to modify it.

If you do not want to support password injection, disable the password fields by editing the dashboard's `local_settings.py` file.

```
OPENSTACK_HYPERVISOR_FEATURES = {
...
    'can_set_password': False,
}
```

Password injection on libvirt-based hypervisors

For hypervisors that use the libvirt back end (such as KVM, QEMU, and LXC), admin password injection is disabled by default. To enable it, set this option in `/etc/nova/nova.conf`:

```
[libvirt]
inject_password=true
```

When enabled, Compute will modify the password of the admin account by editing the `/etc/shadow` file inside the virtual machine instance.

> **Note**
>
> Users can only use **ssh** to access the instance by using the admin password if the virtual machine image is a Linux distribution, and it has been configured to allow users to use **ssh** as the root user. This is not the case for Ubuntu cloud images which, by default, does not allow users to use **ssh** to access the root account.

Password injection and XenAPI (XenServer/XCP)

When using the XenAPI hypervisor back end, Compute uses the XenAPI agent to inject passwords into guests. The virtual machine image must be configured with the agent for password injection to work.

Password injection and Windows images (all hypervisors)

For Windows virtual machines, configure the Windows image to retrieve the admin password on boot by installing an agent such as cloudbase-init.

Manage the cloud

System administrators can use the **openstack** and **euca2ools** commands to manage their clouds.

The `openstack` client and `euca2ools` can be used by all users, though specific commands might be restricted by the Identity service.

Managing the cloud with the openstack client

1. The `python-openstackclient` package provides an `openstack` shell that enables Compute API interactions from the command line. Install the client, and provide

your user name and password (which can be set as environment variables for convenience), for the ability to administer the cloud from the command line.

To install python-openstackclient, follow the instructions in the OpenStack User Guide.

2. Confirm the installation was successful:

```
$ openstack help
usage: openstack [--version] [-v | -q] [--log-file LOG_FILE] [-h] [--
debug]
            [--os-cloud <cloud-config-name>]
            [--os-region-name <auth-region-name>]
            [--os-cacert <ca-bundle-file>] [--verify | --insecure]
            [--os-default-domain <auth-domain>]
            ...
```

Running **openstack help** returns a list of `openstack` commands and parameters. To get help for a subcommand, run:

```
$ openstack help SUBCOMMAND
```

For a complete list of `openstack` commands and parameters, see the OpenStack Command-Line Reference.

3. Set the required parameters as environment variables to make running commands easier. For example, you can add *--os-username* as an `openstack` option, or set it as an environment variable. To set the user name, password, and tenant as environment variables, use:

```
$ export OS_USERNAME=joecool
$ export OS_PASSWORD=coolword
$ export OS_TENANT_NAME=coolu
```

4. The Identity service gives you an authentication endpoint, which Compute recognizes as `OS_AUTH_URL`:

```
$ export OS_AUTH_URL=http://hostname:5000/v2.0
```

Managing the cloud with euca2ools

The `euca2ools` command-line tool provides a command line interface to EC2 API calls. For more information, see the Official Eucalyptus Documentation.

Show usage statistics for hosts and instances

You can show basic statistics on resource usage for hosts and instances.

> **Note**
>
> For more sophisticated monitoring, see the ceilometer project. You can also use tools, such as Ganglia or Graphite, to gather more detailed data.

Show host usage statistics

The following examples show the host usage statistics for a host called `devstack`.

- List the hosts and the nova-related services that run on them:

```
$ nova host-list
+-----------+-------------+----------+
| host_name | service     | zone     |
+-----------+-------------+----------+
| devstack  | conductor   | internal |
| devstack  | compute     | nova     |
| devstack  | cert        | internal |
| devstack  | network     | internal |
| devstack  | scheduler   | internal |
| devstack  | consoleauth | internal |
+-----------+-------------+----------+
```

- Get a summary of resource usage of all of the instances running on the host:

```
$ nova host-describe devstack
+----------+----------------------------------+-----+-----------+---------+
| HOST     | PROJECT                          | cpu | memory_mb | disk_gb |
+----------+----------------------------------+-----+-----------+---------+
| devstack | (total)                          | 2   | 4003      | 157     |
| devstack | (used_now)                       | 3   | 5120      | 40      |
| devstack | (used_max)                       | 3   | 4608      | 40      |
| devstack | b70d90d65e464582b6b2161cf3603ced | 1   | 512       | 0       |
| devstack | 66265572db174a7aa66eba661f58eb9e | 2   | 4096      | 40      |
+----------+----------------------------------+-----+-----------+---------+
```

  The `cpu` column shows the sum of the virtual CPUs for instances running on the host.

  The `memory_mb` column shows the sum of the memory (in MB) allocated to the instances that run on the host.

  The `disk_gb` column shows the sum of the root and ephemeral disk sizes (in GB) of the instances that run on the host.

  The row that has the value `used_now` in the `PROJECT` column shows the sum of the resources allocated to the instances that run on the host, plus the resources allocated to the virtual machine of the host itself.

The row that has the value `used_max` in the `PROJECT` column shows the sum of the resources allocated to the instances that run on the host.

> **Note**
>
> These values are computed by using information about the flavors of the instances that run on the hosts. This command does not query the CPU usage, memory usage, or hard disk usage of the physical host.

Show instance usage statistics

- Get CPU, memory, I/O, and network statistics for an instance.

  1. List instances:

     ```
     $ nova list
     +----------+----------------------+--------+------------+-------------+------------------+
     | ID       | Name                 | Status | Task State | Power State | Networks         |
     +----------+----------------------+--------+------------+-------------+------------------+
     | 84c6e... | myCirrosServer       | ACTIVE | None       | Running     | private=10.0.0.3 |
     | 8a995... | myInstanceFromVolume | ACTIVE | None       | Running     | private=10.0.0.4 |
     +----------+----------------------+--------+------------+-------------+------------------+
     ```

  2. Get diagnostic statistics:

     ```
     $ nova diagnostics myCirrosServer
     +------------------+----------------+
     | Property         | Value          |
     +------------------+----------------+
     | vnet1_rx         | 1210744        |
     | cpu0_time        | 19624610000000 |
     | vda_read         | 0              |
     | vda_write        | 0              |
     | vda_write_req    | 0              |
     | vnet1_tx         | 863734         |
     | vnet1_tx_errors  | 0              |
     | vnet1_rx_drop    | 0              |
     | vnet1_tx_packets | 3855           |
     | vnet1_tx_drop    | 0              |
     | vnet1_rx_errors  | 0              |
     | memory           | 2097152        |
     | vnet1_rx_packets | 5485           |
     | vda_read_req     | 0              |
     | vda_errors       | -1             |
     +------------------+----------------+
     ```

- Get summary statistics for each tenant:

```
$ nova usage-list
Usage from 2013-06-25 to 2013-07-24:
+----------------------------------+-----------+--------------+-----------+---------------+
| Tenant ID                        | Instances | RAM MB-Hours | CPU Hours | Disk GB-Hours |
+----------------------------------+-----------+--------------+-----------+---------------+
| b70d90d65e464582b6b2161cf3603ced | 1         | 344064.44    | 672.00    | 0.00          |
| 66265572db174a7aa66eba661f58eb9e | 3         | 671626.76    | 327.94    | 6558.86       |
+----------------------------------+-----------+--------------+-----------+---------------+
```

Logging

Logging module

Logging behavior can be changed by creating a configuration file. To specify the configuration file, add this line to the `/etc/nova/nova.conf` file:

```
log-config=/etc/nova/logging.conf
```

To change the logging level, add `DEBUG`, `INFO`, `WARNING`, or `ERROR` as a parameter.

The logging configuration file is an INI-style configuration file, which must contain a section called `logger_nova`. This controls the behavior of the logging facility in the `nova-*` services. For example:

```
[logger_nova]
level = INFO
handlers = stderr
qualname = nova
```

This example sets the debugging level to `INFO` (which is less verbose than the default `DEBUG` setting).

For more about the logging configuration syntax, including the `handlers` and `quaname` variables, see the Python documentation on logging configuration files.

For an example of the `logging.conf` file with various defined handlers, see the OpenStack Configuration Reference.

Syslog

OpenStack Compute services can send logging information to syslog. This is useful if you want to use rsyslog to forward logs to a remote machine. Separately configure the Compute service (nova), the Identity service (keystone), the Image service (glance), and, if you are using it, the Block Storage service (cinder) to send log messages to syslog. Open these configuration files:

- `/etc/nova/nova.conf`
- `/etc/keystone/keystone.conf`

- /etc/glance/glance-api.conf
- /etc/glance/glance-registry.conf
- /etc/cinder/cinder.conf

In each configuration file, add these lines:

```
debug = False
use_syslog = True
syslog_log_facility = LOG_LOCAL0
```

In addition to enabling syslog, these settings also turn off debugging output from the log.

> **Note**
>
> Although this example uses the same local facility for each service (LOG_LOCAL0, which corresponds to syslog facility LOCAL0), we recommend that you configure a separate local facility for each service, as this provides better isolation and more flexibility. For example, you can capture logging information at different severity levels for different services. syslog allows you to define up to eight local facilities, LOCAL0, LOCAL1, ..., LOCAL7. For more information, see the syslog documentation.

Rsyslog

rsyslog is useful for setting up a centralized log server across multiple machines. This section briefly describe the configuration to set up an rsyslog server. A full treatment of rsyslog is beyond the scope of this book. This section assumes rsyslog has already been installed on your hosts (it is installed by default on most Linux distributions).

This example provides a minimal configuration for /etc/rsyslog.conf on the log server host, which receives the log files

```
# provides TCP syslog reception
$ModLoad imtcp
$InputTCPServerRun 1024
```

Add a filter rule to /etc/rsyslog.conf which looks for a host name. This example uses COMPUTE_01 as the compute host name:

```
:hostname, isequal, "COMPUTE_01" /mnt/rsyslog/logs/compute-01.log
```

On each compute host, create a file named /etc/rsyslog.d/60-nova.conf, with the following content:

```
# prevent debug from dnsmasq with the daemon.none parameter
*.*;auth,authpriv.none,daemon.none,local0.none -/var/log/syslog
# Specify a log level of ERROR
local0.error    @@172.20.1.43:1024
```

Once you have created the file, restart the `rsyslog` service. Error-level log messages on the compute hosts should now be sent to the log server.

Serial console

The serial console provides a way to examine kernel output and other system messages during troubleshooting if the instance lacks network connectivity.

Read-only access from server serial console is possible using the `os-GetSerialOutput` server action. Most cloud images enable this feature by default. For more information, see *Common errors and fixes for Compute*.

OpenStack Juno and later supports read-write access using the serial console using the `os-GetSerialConsole` server action. This feature also requires a websocket client to access the serial console.

Configuring read-write serial console access

1. On a compute node, edit the `/etc/nova/nova.conf` file:

   In the `[serial_console]` section, enable the serial console:

   ```
   [serial_console]
   ...
   enabled = true
   ```

2. In the `[serial_console]` section, configure the serial console proxy similar to graphical console proxies:

   ```
   [serial_console]
   ...
   base_url = ws://controller:6083/
   listen = 0.0.0.0
   proxyclient_address = MANAGEMENT_INTERFACE_IP_ADDRESS
   ```

   The `base_url` option specifies the base URL that clients receive from the API upon requesting a serial console. Typically, this refers to the host name of the controller node.

   The `listen` option specifies the network interface nova-compute should listen on for virtual console connections. Typically, 0.0.0.0 will enable listening on all interfaces.

   The `proxyclient_address` option specifies which network interface the proxy should connect to. Typically, this refers to the IP address of the management interface.

   When you enable read-write serial console access, Compute will add serial console information to the Libvirt XML file for the instance. For example:

```
<console type='tcp'>
  <source mode='bind' host='127.0.0.1' service='10000'/>
  <protocol type='raw'/>
  <target type='serial' port='0'/>
  <alias name='serial0'/>
</console>
```

Accessing the serial console on an instance

1. Use the **nova get-serial-proxy** command to retrieve the websocket URL for the serial console on the instance:

   ```
   $ nova get-serial-proxy INSTANCE_NAME
   ```

Type	Url
serial	ws://127.0.0.1:6083/?token=18510769-71ad-4e5a-8348-4218b5613b3d

   Alternatively, use the API directly:

   ```
   $ curl -i 'http://<controller>:8774/v2.1/<tenant_uuid>/servers/
     <instance_uuid>/action' \
     -X POST \
     -H "Accept: application/json" \
     -H "Content-Type: application/json" \
     -H "X-Auth-Project-Id: <project_id>" \
     -H "X-Auth-Token: <auth_token>" \
     -d '{"os-getSerialConsole": {"type": "serial"}}'
   ```

2. Use Python websocket with the URL to generate `.send`, `.recv`, and `.fileno` methods for serial console access. For example:

   ```
   import websocket
   ws = websocket.create_connection(
       'ws://127.0.0.1:6083/?token=18510769-71ad-4e5a-8348-4218b5613b3d',
       subprotocols=['binary', 'base64'])
   ```

Alternatively, use a Python websocket client.

> **Note**
>
> When you enable the serial console, typical instance logging using the **nova console-log** command is disabled. Kernel output and other system messages will not be visible unless you are actively viewing the serial console.

Secure with rootwrap

Rootwrap allows unprivileged users to safely run Compute actions as the root user. Compute previously used **sudo** for this purpose, but this was difficult to maintain, and did not allow advanced filters. The **rootwrap** command replaces **sudo** for Compute.

To use rootwrap, prefix the Compute command with **nova-rootwrap**. For example:

```
$ sudo nova-rootwrap /etc/nova/rootwrap.conf command
```

A generic `sudoers` entry lets the Compute user run **nova-rootwrap** as root. The **nova-rootwrap** code looks for filter definition directories in its configuration file, and loads command filters from them. It then checks if the command requested by Compute matches one of those filters and, if so, executes the command (as root). If no filter matches, it denies the request.

> **Note**
>
> Be aware of issues with using NFS and root-owned files. The NFS share must be configured with the `no_root_squash` option enabled, in order for rootwrap to work correctly.

Rootwrap is fully controlled by the root user. The root user owns the sudoers entry which allows Compute to run a specific rootwrap executable as root, and only with a specific configuration file (which should also be owned by root). The **nova-rootwrap** command imports the Python modules it needs from a cleaned, system-default PYTHONPATH. The root-owned configuration file points to root-owned filter definition directories, which contain root-owned filters definition files. This chain ensures that the Compute user itself is not in control of the configuration or modules used by the **nova-rootwrap** executable.

Configure rootwrap

Configure rootwrap in the `rootwrap.conf` file. Because it is in the trusted security path, it must be owned and writable by only the root user. The `rootwrap_config=entry` parameter specifies the file's location in the sudoers entry and in the `nova.conf` configuration file.

The `rootwrap.conf` file uses an INI file format with these sections and parameters:

rootwrap.conf configuration options

Configuration option=Default value	(Type) Description
[DEFAULT] filters_path=/etc/nova/rootwrap.d,/usr/share/nova/rootwrap	(ListOpt) Comma-separated list of directories containing filter definition files. Defines where rootwrap filters are stored. Directories defined on this line should all exist, and be owned and writable

	only by the root user.

If the root wrapper is not performing correctly, you can add a workaround option into the `nova.conf` configuration file. This workaround re-configures the root wrapper configuration to fall back to running commands as `sudo`, and is a Kilo release feature.

Including this workaround in your configuration file safeguards your environment from issues that can impair root wrapper performance. Tool changes that have impacted Python Build Reasonableness (PBR) for example, are a known issue that affects root wrapper performance.

To set up this workaround, configure the `disable_rootwrap` option in the `[workaround]` section of the `nova.conf` configuration file.

The filters definition files contain lists of filters that rootwrap will use to allow or deny a specific command. They are generally suffixed by `.filters` . Since they are in the trusted security path, they need to be owned and writable only by the root user. Their location is specified in the `rootwrap.conf` file.

Filter definition files use an INI file format with a `[Filters]` section and several lines, each with a unique parameter name, which should be different for each filter you define:

Filters configuration options

Configuration option=Default value	(Type) Description
[Filters] filter_name=kpartx: CommandFilter, /sbin/kpartx, root	(ListOpt) Comma-separated list containing the filter class to use, followed by the Filter arguments (which vary depending on the Filter class selected).

Configure the rootwrap daemon

Administrators can use rootwrap daemon support instead of running rootwrap with **sudo**. The rootwrap daemon reduces the overhead and performance loss that results from running `oslo.rootwrap` with **sudo**. Each call that needs rootwrap privileges requires a new instance of rootwrap. The daemon prevents overhead from the repeated calls. The daemon does not support long running processes, however.

To enable the rootwrap daemon, set `use_rootwrap_daemon` to `True` in the Compute service configuration file.

Configure migrations

> **Note**
>
> Only administrators can perform live migrations. If your cloud is configured to use cells, you can perform live migration within but not between cells.

Migration enables an administrator to move a virtual-machine instance from one compute host to another. This feature is useful when a compute host requires maintenance. Migration can also be useful to redistribute the load when many VM instances are running on a specific physical machine.

The migration types are:

- **Non-live migration** (sometimes referred to simply as 'migration'). The instance is shut down for a period of time to be moved to another hypervisor. In this case, the instance recognizes that it was rebooted.
- **Live migration** (or 'true live migration'). Almost no instance downtime. Useful when the instances must be kept running during the migration. The different types of live migration are:
  - **Shared storage-based live migration**. Both hypervisors have access to shared storage.
  - **Block live migration**. No shared storage is required. Incompatible with read-only devices such as CD-ROMs and Configuration Drive (config_drive).
  - **Volume-backed live migration**. Instances are backed by volumes rather than ephemeral disk, no shared storage is required, and migration is supported (currently only available for libvirt-based hypervisors).

The following sections describe how to configure your hosts and compute nodes for migrations by using the KVM and XenServer hypervisors.

KVM-Libvirt

Shared storage

Prerequisites

- **Hypervisor:** KVM with libvirt
- **Shared storage:** `NOVA-INST-DIR/instances/` (for example, `/var/lib/nova/instances`) has to be mounted by shared storage. This guide uses NFS but other options, including the OpenStack Gluster Connector are available.
- **Instances:** Instance can be migrated with iSCSI-based volumes.

Notes

- Because the Compute service does not use the libvirt live migration functionality by default, guests are suspended before migration and might experience several minutes of downtime. For details, see Enabling true live migration.
- Compute calculates the amount of downtime required using the RAM size of the disk being migrated, in accordance with the `live_migration_downtime` configuration parameters. Migration downtime is measured in steps, with an exponential backoff between each step. This means that the maximum downtime between each step starts off small, and is increased in ever larger amounts as Compute waits for the migration to complete. This gives the guest a chance to complete the migration successfully, with a minimum amount of downtime.
- This guide assumes the default value for `instances_path` in your `nova.conf` file (`NOVA-INST-DIR/instances`). If you have changed the `state_path` or `instances_path` variables, modify the commands accordingly.
- You must specify `vncserver_listen=0.0.0.0` or live migration will not work correctly.
- You must specify the `instances_path` in each node that runs `nova-compute`. The mount point for `instances_path` must be the same value for each node, or live migration will not work correctly.

Example Compute installation environment

- Prepare at least three servers. In this example, we refer to the servers as `HostA`, `HostB`, and `HostC`:

  - `HostA` is the Cloud Controller, and should run these services: `nova-api`, `nova-scheduler`, `nova-network`, `cinder-volume`, and `nova-objectstore`.
  - `HostB` and `HostC` are the compute nodes that run `nova-compute`.

  Ensure that `NOVA-INST-DIR` (set with `state_path` in the `nova.conf` file) is the same on all hosts.

- In this example, `HostA` is the NFSv4 server that exports `NOVA-INST-DIR/instances` directory. `HostB` and `HostC` are NFSv4 clients that mount `HostA`.

Configuring your system

1. Configure your DNS or `/etc/hosts` and ensure it is consistent across all hosts. Make sure that the three hosts can perform name resolution with each other. As a test, use the **ping** command to ping each host from one another:

   ```
   $ ping HostA
   $ ping HostB
   $ ping HostC
   ```

2. Ensure that the UID and GID of your Compute and libvirt users are identical between each of your servers. This ensures that the permissions on the NFS mount works correctly.

3. Ensure you can access SSH without a password and without StrictHostKeyChecking between `HostB` and `HostC` as `nova` user (set with the owner of `nova-compute` service). Direct access from one compute host to another is needed to copy the VM file across. It is also needed to detect if the source and target compute nodes share a storage subsystem.

4. Export `NOVA-INST-DIR/instances` from `HostA`, and ensure it is readable and writable by the Compute user on `HostB` and `HostC`.

   For more information, see: SettingUpNFSHowTo or CentOS/Red Hat: Setup NFS v4.0 File Server

5. Configure the NFS server at `HostA` by adding the following line to the `/etc/exports` file:

   ```
   NOVA-INST-DIR/instances HostA/255.255.0.0(rw,sync,fsid=0,no_root_squash)
   ```

   Change the subnet mask (`255.255.0.0`) to the appropriate value to include the IP addresses of `HostB` and `HostC`. Then restart the `NFS` server:

   ```
   # /etc/init.d/nfs-kernel-server restart
   # /etc/init.d/idmapd restart
   ```

6. On both compute nodes, enable the `execute/search` bit on your shared directory to allow qemu to be able to use the images within the directories. On all hosts, run the following command:

   ```
   $ chmod o+x NOVA-INST-DIR/instances
   ```

7. Configure NFS on `HostB` and `HostC` by adding the following line to the `/etc/fstab` file

   ```
   HostA:/ /NOVA-INST-DIR/instances nfs4 defaults 0 0
   ```

   Ensure that you can mount the exported directory

   ```
   $ mount -a -v
   ```

   Check that `HostA` can see the `NOVA-INST-DIR/instances/` directory

   ```
   $ ls -ld NOVA-INST-DIR/instances/
   drwxr-xr-x 2 nova nova 4096 2012-05-19 14:34 nova-install-dir/instances/
   ```

   Perform the same check on `HostB` and `HostC`, paying special attention to the permissions (Compute should be able to write)

```
$ ls -ld NOVA-INST-DIR/instances/
drwxr-xr-x 2 nova nova 4096 2012-05-07 14:34 nova-install-dir/instances/

$ df -k
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda1            921514972   4180880 870523828   1% /
none                  16498340      1228  16497112   1% /dev
none                  16502856         0  16502856   0% /dev/shm
none                  16502856       368  16502488   1% /var/run
none                  16502856         0  16502856   0% /var/lock
none                  16502856         0  16502856   0% /lib/init/rw
HostA:               921515008 101921792 772783104  12%
/var/lib/nova/instances  ( <--- this line is important.)
```

8. Update the libvirt configurations so that the calls can be made securely. These methods enable remote access over TCP and are not documented here.

   - SSH tunnel to libvirtd's UNIX socket
   - libvirtd TCP socket, with GSSAPI/Kerberos for auth+data encryption
   - libvirtd TCP socket, with TLS for encryption and x509 client certs for authentication
   - libvirtd TCP socket, with TLS for encryption and Kerberos for authentication

   Restart `libvirt`. After you run the command, ensure that libvirt is successfully restarted

   ```
   # stop libvirt-bin && start libvirt-bin
   $ ps -ef | grep libvirt
   root 1145 1 0 Nov27 ? 00:00:03 /usr/sbin/libvirtd -d -l\
   ```

9. Configure your firewall to allow libvirt to communicate between nodes. By default, libvirt listens on TCP port 16509, and an ephemeral TCP range from 49152 to 49261 is used for the KVM communications. Based on the secure remote access TCP configuration you chose, be careful which ports you open, and always understand who has access. For information about ports that are used with libvirt, see the libvirt documentation.

10. Configure the downtime required for the migration by adjusting these parameters in the `nova.conf` file:

    ```
    live_migration_downtime = 500
    live_migration_downtime_steps = 10
    live_migration_downtime_delay = 75
    ```

    The `live_migration_downtime` parameter sets the maximum permitted downtime for a live migration, in milliseconds. This setting defaults to 500 milliseconds.

The `live_migration_downtime_steps` parameter sets the total number of incremental steps to reach the maximum downtime value. This setting defaults to 10 steps.

The `live_migration_downtime_delay` parameter sets the amount of time to wait between each step, in seconds. This setting defaults to 75 seconds.

11. You can now configure other options for live migration. In most cases, you will not need to configure any options. For advanced configuration options, see the OpenStack Configuration Reference Guide.

Enabling true live migration

Prior to the Kilo release, the Compute service did not use the libvirt live migration function by default. To enable this function, add the following line to the `[libvirt]` section of the `nova.conf` file:

```
live_migration_flag=VIR_MIGRATE_UNDEFINE_SOURCE,VIR_MIGRATE_PEER2PEER,VIR_MIGRATE_LIVE,VIR_MIGRATE_TUNNELLED
```

On versions older than Kilo, the Compute service does not use libvirt's live migration by default because there is a risk that the migration process will never end. This can happen if the guest operating system uses blocks on the disk faster than they can be migrated.

Block migration

Configuring KVM for block migration is exactly the same as the above configuration in *Shared storage* the section called shared storage, except that `NOVA-INST-DIR/instances` is local to each host rather than shared. No NFS client or server configuration is required.

> **Note**
>
> - To use block migration, you must use the *--block-migrate* parameter with the live migration command.
>
> - Block migration is incompatible with read-only devices such as CD-ROMs and Configuration Drive (config_drive).
>
> - Since the ephemeral drives are copied over the network in block migration, migrations of instances with heavy I/O loads may never complete if the drives are writing faster than the data can be copied over the network.

XenServer

Shared storage

Prerequisites

- **Compatible XenServer hypervisors**. For more information, see the Requirements for Creating Resource Pools section of the XenServer Administrator's Guide.
- **Shared storage**. An NFS export, visible to all XenServer hosts.

> **Note**
>
> For the supported NFS versions, see the NFS VHD section of the XenServer Administrator's Guide.

To use shared storage live migration with XenServer hypervisors, the hosts must be joined to a XenServer pool. To create that pool, a host aggregate must be created with specific metadata. This metadata is used by the XAPI plug-ins to establish the pool.

Using shared storage live migrations with XenServer Hypervisors

1. Add an NFS VHD storage to your master XenServer, and set it as the default storage repository. For more information, see NFS VHD in the XenServer Administrator's Guide.
2. Configure all compute nodes to use the default storage repository (`sr`) for pool operations. Add this line to your `nova.conf` configuration files on all compute nodes:

   ```
   sr_matching_filter=default-sr:true
   ```

3. Create a host aggregate. This command creates the aggregate, and then displays a table that contains the ID of the new aggregate

   ```
   $ nova aggregate-create POOL_NAME AVAILABILITY_ZONE
   ```

   Add metadata to the aggregate, to mark it as a hypervisor pool

   ```
   $ nova aggregate-set-metadata AGGREGATE_ID hypervisor_pool=true

   $ nova aggregate-set-metadata AGGREGATE_ID operational_state=created
   ```

   Make the first compute node part of that aggregate

   ```
   $ nova aggregate-add-host AGGREGATE_ID MASTER_COMPUTE_NAME
   ```

   The host is now part of a XenServer pool.

4. Add hosts to the pool

```
$ nova aggregate-add-host AGGREGATE_ID COMPUTE_HOST_NAME
```

> **Note**
>
> The added compute node and the host will shut down to join the host to the XenServer pool. The operation will fail if any server other than the compute node is running or suspended on the host.

Block migration

- **Compatible XenServer hypervisors**. The hypervisors must support the Storage XenMotion feature. See your XenServer manual to make sure your edition has this feature.

> **Note**
>
> - To use block migration, you must use the *--block-migrate* parameter with the live migration command.
>
> - Block migration works only with EXT local storage storage repositories, and the server must not have any volumes attached.

Migrate instances

This section discusses how to migrate running instances from one OpenStack Compute server to another OpenStack Compute server.

Before starting a migration, review the Configure migrations section. *Configure migrations*.

> **Note**
>
> Although the **nova** command is called **live-migration**, under the default Compute configuration options, the instances are suspended before migration. For more information, see Configure migrations. in the OpenStack Configuration Reference.

Migrating instances

1. Check the ID of the instance to be migrated:

```
$ nova list
```

ID	Name	Status	Networks
d1df1b5a-70c4-4fed-98b7-423362f2c47c	vm1	ACTIVE	private=a.b.c.d
d693db9e-a7cf-45ef-a7c9-b3ecb5f22645	vm2	ACTIVE	private=e.f.g.h

2. Check the information associated with the instance. In this example, `vm1` is running on `HostB`:

```
$ nova show d1df1b5a-70c4-4fed-98b7-423362f2c47c
```

Property	Value
...	...
OS-EXT-SRV-ATTR:host	HostB
...	...
flavor	m1.tiny
id	d1df1b5a-70c4-4fed-98b7-423362f2c47c
name	vm1
private network	a.b.c.d
status	ACTIVE
...	...

3. Select the compute node the instance will be migrated to. In this example, we will migrate the instance to `HostC`, because `nova-compute` is running on it:

nova service-list

Binary	Host	Zone	Status	State	Updated_at	Disabled Reason
nova-consoleauth	HostA	internal	enabled	up	2014-03-25T10:33:25.000000	•
nova-scheduler	HostA	internal	enabled	up	2014-03-25T10:33:25.000000	•
nova-conductor	HostA	internal	enabled	up	2014-03-25T10:33:27.000000	•
nova-compute	HostB	nova	enabled	up	2014-03-25T10:33:31.000000	•
nova-compute	HostC	nova	enabled	up	2014-03-25T10:33:31.000000	•
nova-cert	HostA	internal	enabled	up	2014-03-25T10:33:31.000000	•

4. Check that `HostC` has enough resources for migration:

```
# nova host-describe HostC
```

HOST	PROJECT	cpu	memory_mb	disk_gb
HostC	(total)	16	32232	878
HostC	(used_now)	22	21284	442
HostC	(used_max)	22	21284	422
HostC	p1	22	21284	422
HostC	p2	22	21284	422

   - `cpu`: Number of CPUs
   - `memory_mb`: Total amount of memory, in MB
   - `disk_gb`: Total amount of space for NOVA-INST-DIR/instances, in GB

   In this table, the first row shows the total amount of resources available on the physical server. The second line shows the currently used resources. The third line shows the maximum used resources. The fourth line and below shows the resources available for each project.

5. Migrate the instance using the **nova live-migration** command:

```
$ nova live-migration SERVER HOST_NAME
```

   In this example, SERVER can be the ID or name of the instance. Another example:

```
$ nova live-migration d1df1b5a-70c4-4fed-98b7-423362f2c47c HostC
Migration of d1df1b5a-70c4-4fed-98b7-423362f2c47c initiated.
```

> **Warning**
>
> When using live migration to move workloads between Icehouse and Juno compute nodes, it may cause data loss because libvirt live migration with shared block storage was buggy (potential loss of data) before version 3.32. This issue can be solved when we upgrade to RPC API version 4.0.

6. Check that the instance has been migrated successfully, using **nova list**. If the instance is still running on `HostB`, check the log files at `src/dest` for `nova-compute` and `nova-scheduler` to determine why.

Configure remote console access

To provide a remote console or remote desktop access to guest virtual machines, use VNC or SPICE HTML5 through either the OpenStack dashboard or the command line. Best practice is to select one or the other to run.

About nova-consoleauth

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

Do not confuse the `nova-consoleauth` shared service with `nova-console`, which is a XenAPI-specific service that most recent VNC proxy architectures do not use.

SPICE console

OpenStack Compute supports VNC consoles to guests. The VNC protocol is fairly limited, lacking support for multiple monitors, bi-directional audio, reliable cut-and-paste, video streaming and more. SPICE is a new protocol that aims to address the limitations in VNC and provide good remote desktop support.

SPICE support in OpenStack Compute shares a similar architecture to the VNC implementation. The OpenStack dashboard uses a SPICE-HTML5 widget in its console tab that communicates to the `nova-spicehtml5proxy` service by using SPICE-over-websockets. The `nova-spicehtml5proxy` service communicates directly with the hypervisor process by using SPICE.

VNC must be explicitly disabled to get access to the SPICE console. Set the `vnc_enabled` option to `False` in the `[DEFAULT]` section to disable the VNC console.

Use the following options to configure SPICE as the console for OpenStack Compute:

Description of SPICE configuration options

[spice]	
Spice configuration option = Default value	Description
`agent_enabled = True`	(BoolOpt) Enable spice guest agent support
`enabled = False`	(BoolOpt) Enable spice related features
`html5proxy_base_url = http://127.0.0.1:6082/spice_auto.html`	(StrOpt) Location of spice HTML5 console proxy, in the form "http://127.0.0.1:6082/spice_auto.html"
`html5proxy_host = 0.0.0.0`	(StrOpt) Host on which to listen for incoming requests

[spice]	
`html5proxy_port = 6082`	(IntOpt) Port on which to listen for incoming requests
`keymap = en-us`	(StrOpt) Keymap for spice
`server_listen = 127.0.0.1`	(StrOpt) IP address on which instance spice server should listen
`server_proxyclient_address = 127.0.0.1`	(StrOpt) The address to which proxy clients (like nova-spicehtml5proxy) should connect

VNC console proxy

The VNC proxy is an OpenStack component that enables compute service users to access their instances through VNC clients.

> **Note**
>
> The web proxy console URLs do not support the websocket protocol scheme (ws://) on python versions less than 2.7.4.

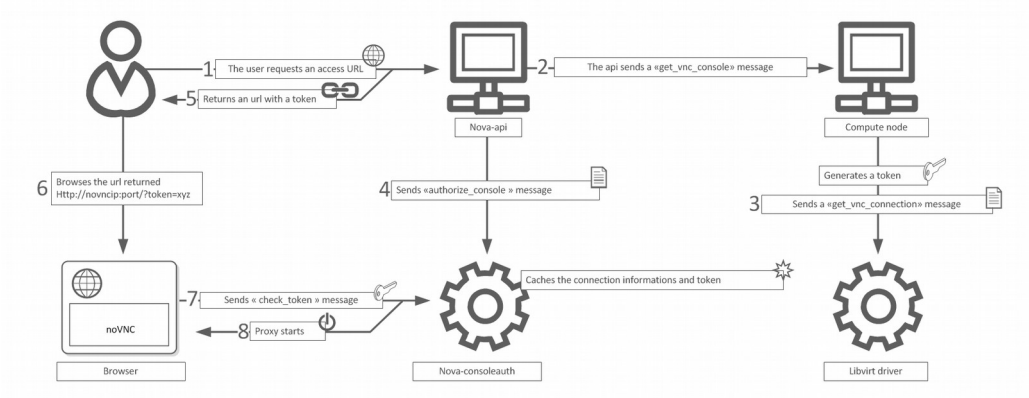
The VNC console connection works as follows:

1. A user connects to the API and gets an `access_url` such as, `http://ip:port/?token=xyz`.
2. The user pastes the URL in a browser or uses it as a client parameter.
3. The browser or client connects to the proxy.
4. The proxy talks to `nova-consoleauth` to authorize the token for the user, and maps the token to the *private* host and port of the VNC server for an instance.

   The compute host specifies the address that the proxy should use to connect through the `nova.conf` file option, `vncserver_proxyclient_address`. In this way, the VNC proxy works as a bridge between the public network and private host network.
5. The proxy initiates the connection to VNC server and continues to proxy until the session ends.

The proxy also tunnels the VNC protocol over WebSockets so that the `noVNC` client can talk to VNC servers. In general, the VNC proxy:

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



VNC configuration options

To customize the VNC console, use the following configuration options in your `nova.conf` file:

> **Note**
>
> To support *live migration*, you cannot specify a specific IP address for `vncserver_listen`, because that IP address does not exist on the destination host.

Description of VNC configuration options

Configuration option = Default value	Description
[DEFAULT]	
`daemon = False`	(BoolOpt) Become a daemon (background process)
`key = None`	(StrOpt) SSL key file (if separate from cert)
`novncproxy_host = 0.0.0.0`	(StrOpt) Host on which to listen for incoming requests
`novncproxy_port = 6080`	(IntOpt) Port on which to listen for incoming requests
`record = False`	(BoolOpt) Record sessions to FILE. [session_number]
`source_is_ipv6 = False`	(BoolOpt) Source is ipv6
`ssl_only = False`	(BoolOpt) Disallow non-encrypted connections
`web = /usr/share/spice-html5`	(StrOpt) Run webserver on same port. Serve files from DIR.
[vmware]	
`vnc_port = 5900`	(IntOpt) VNC starting port

Configuration option = Default value	Description
vnc_port_total = 10000	vnc_port_total = 10000
[vnc]	
enabled = True	(BoolOpt) Enable VNC related features
novncproxy_base_url = http://127.0.0.1:6080/vnc_auto.html	(StrOpt) Location of VNC console proxy, in the form "http://127.0.0.1:6080/vnc_auto.html"
vncserver_listen = 127.0.0.1	(StrOpt) IP address on which instance vncservers should listen
vncserver_proxyclient_address = 127.0.0.1	(StrOpt) The address to which proxy clients (like nova-xvpvncproxy) should connect
xvpvncproxy_base_url = http://127.0.0.1:6081/console	(StrOpt) Location of nova xvp VNC console proxy, in the form "http://127.0.0.1:6081/console"

> **Note**
>
> - The `vncserver_proxyclient_address` defaults to `127.0.0.1`, which is the address of the compute host that Compute instructs proxies to use when connecting to instance servers.
> - For all-in-one XenServer domU deployments, set this to `169.254.0.1`.
> - For multi-host XenServer domU deployments, set to a `dom0 management IP` on the same network as the proxies.
> - For multi-host libvirt deployments, set to a host management IP on the same network as the proxies.

Typical deployment

A typical deployment has the following components:

- A `nova-consoleauth` process. Typically runs on the controller host.
- One or more `nova-novncproxy` services. Supports browser-based noVNC clients. For simple deployments, this service typically runs on the same machine as `nova-api` because it operates as a proxy between the public network and the private compute host network.
- One or more `nova-xvpvncproxy` services. Supports the special Java client discussed

here. For simple deployments, this service typically runs on the same machine as `nova-api` because it acts as a proxy between the public network and the private compute host network.

- One or more compute hosts. These compute hosts must have correctly configured options, as follows.

nova-novncproxy (noVNC)

You must install the noVNC package, which contains the `nova-novncproxy` service. As root, run the following command:

```
# apt-get install nova-novncproxy
```

The service starts automatically on installation.

To restart the service, run:

```
# service nova-novncproxy restart
```

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

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

To connect the service to your Compute deployment, add the following configuration options to your `nova.conf` file:

- `vncserver_listen=0.0.0.0`

  Specifies the address on which the VNC service should bind. Make sure it is assigned one of the compute node interfaces. This address is the one used by your domain file.

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

> **Note**
>
> To use live migration, use the 0.0.0.0 address.

- `vncserver_proxyclient_address=127.0.0.1`

  The address of the compute host that Compute instructs proxies to use when connecting to instance `vncservers`.

Frequently asked questions about VNC access to virtual machines

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

  A: `nova-xvpvncproxy`, which ships with OpenStack Compute, is a 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 OpenStack 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 OpenStack 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 waits 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 must set 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**
  >
  > `novncproxy_base_url` and `xvpvncproxy_base_url` use a public IP; this is the URL that is ultimately returned to clients, which generally do not have access to

> your private network. Your PROXYSERVER must be able to reach `vncserver_proxyclient_address`, because that is the address over which the VNC connection is proxied.

- **Q: My noVNC does not work with recent versions of web browsers. Why?**

  A: Make sure you have installed `python-numpy`, which is required to support a newer version of the WebSocket protocol (HyBi-07+).

- **Q: How do I adjust the dimensions of the VNC window image in the OpenStack dashboard?**

  A: These values are hard-coded in a Django HTML template. To alter them, edit the `_detail_vnc.html` template file. The location of this file varies based on Linux distribution. On Ubuntu 14.04, the file is at `/usr/share/pyshared/horizon/dashboards/nova/instances/templates/instances/_detail_vnc.html`.

  Modify the `width` and `height` options, as follows:

  ```
  <iframe src="{{ vnc_url }}" width="720" height="430"></iframe>
  ```

- **Q: My noVNC connections failed with ValidationError: Origin header protocol does not match. Why?**

  A: Make sure the `base_url` match your TLS setting. If you are using https console connections, make sure that the value of `novncproxy_base_url` is set explicitly where the `nova-novncproxy` service is running.

Configure Compute service groups

The Compute service must know the status of each compute node to effectively manage and use them. This can include events like a user launching a new VM, the scheduler sending a request to a live node, or a query to the ServiceGroup API to determine if a node is live.

When a compute worker running the nova-compute daemon starts, it calls the join API to join the compute group. Any service (such as the scheduler) can query the group's membership and the status of its nodes. Internally, the ServiceGroup client driver automatically updates the compute worker status.

Database ServiceGroup driver

By default, Compute uses the database driver to track if a node is live. In a compute worker, this driver periodically sends a `db update` command to the database, saying "I'm OK" with a timestamp. Compute uses a pre-defined timeout (`service_down_time`) to determine if a node is dead.

The driver has limitations, which can be problematic depending on your environment. If a lot of compute worker nodes need to be checked, the database can be put under heavy load, which can cause the timeout to trigger, and a live node could incorrectly be considered dead. By default, the timeout is 60 seconds. Reducing the timeout value can help in this situation, but you must also make the database update more frequently, which again increases the database workload.

The database contains data that is both transient (such as whether the node is alive) and persistent (such as entries for VM owners). With the ServiceGroup abstraction, Compute can treat each type separately.

ZooKeeper ServiceGroup driver

The ZooKeeper ServiceGroup driver works by using ZooKeeper ephemeral nodes. ZooKeeper, unlike databases, is a distributed system, with its load divided among several servers. On a compute worker node, the driver can establish a ZooKeeper session, then create an ephemeral znode in the group directory. Ephemeral znodes have the same lifespan as the session. If the worker node or the nova-compute daemon crashes, or a network partition is in place between the worker and the ZooKeeper server quorums, the ephemeral znodes are removed automatically. The driver can be given group membership by running the **ls** command in the group directory.

The ZooKeeper driver requires the ZooKeeper servers and client libraries. Setting up ZooKeeper servers is outside the scope of this guide (for more information, see Apache Zookeeper). These client-side Python libraries must be installed on every compute node:

python-zookeeper
: The official Zookeeper Python binding

evzookeeper
: This library makes the binding work with the eventlet threading model.

This example assumes the ZooKeeper server addresses and ports are `192.168.2.1:2181`, `192.168.2.2:2181`, and `192.168.2.3:2181`.

These values in the `/etc/nova/nova.conf` file are required on every node for the ZooKeeper driver:

```
# Driver for the ServiceGroup service
servicegroup_driver="zk"

[zookeeper]
address="192.168.2.1:2181,192.168.2.2:2181,192.168.2.3:2181"
```

Memcache ServiceGroup driver

The memcache ServiceGroup driver uses memcached, a distributed memory object caching system that is used to increase site performance. For more details, see

memcached.org.

To use the memcache driver, you must install memcached. You might already have it installed, as the same driver is also used for the OpenStack Object Storage and OpenStack dashboard. If you need to install memcached, see the instructions in the OpenStack Installation Guide.

These values in the `/etc/nova/nova.conf` file are required on every node for the memcache driver:

```
# Driver for the ServiceGroup service
servicegroup_driver = "mc"

# Memcached servers. Use either a list of memcached servers to use for caching
(list value),
# or "<None>" for in-process caching (default).
memcached_servers = <None>

# Timeout; maximum time since last check-in for up service (integer value).
# Helps to define whether a node is dead
service_down_time = 60
```

Security hardening

OpenStack Compute can be integrated with various third-party technologies to increase security. For more information, see the OpenStack Security Guide.

Trusted compute pools

Administrators can designate a group of compute hosts as trusted using trusted compute pools. The trusted hosts use hardware-based security features, such as the Intel Trusted Execution Technology (TXT), to provide an additional level of security. Combined with an external stand-alone, web-based remote attestation server, cloud providers can ensure that the compute node runs only software with verified measurements and can ensure a secure cloud stack.

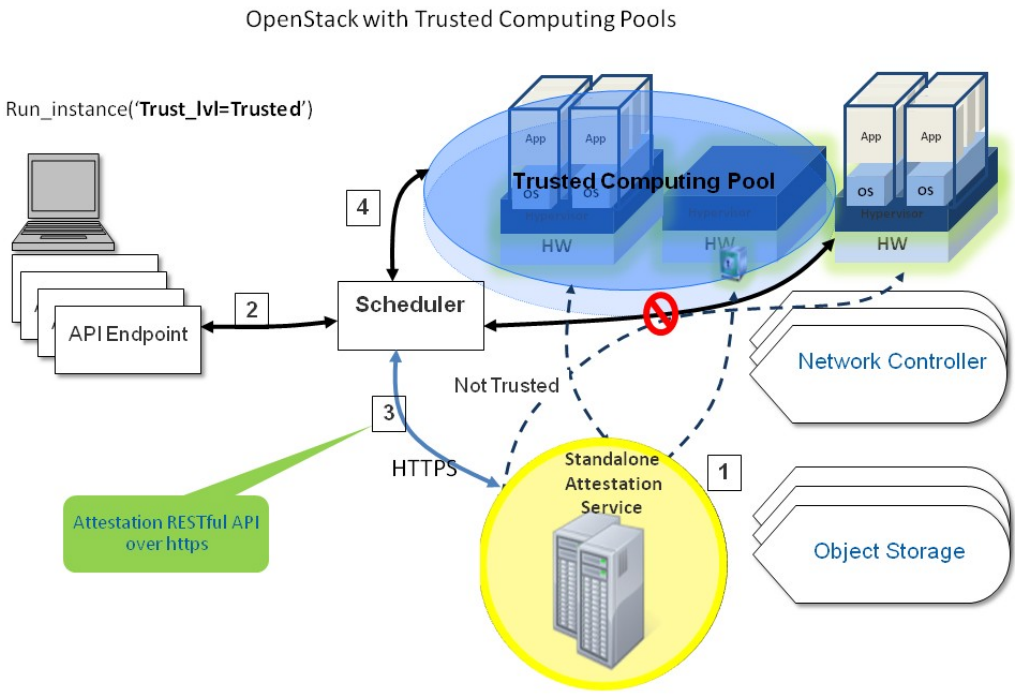
Trusted compute pools provide the ability for cloud subscribers to request services run only on verified compute nodes.

The remote attestation server performs node verification like this:

1. Compute nodes boot with Intel TXT technology enabled.
2. The compute node BIOS, hypervisor, and operating system are measured.
3. When the attestation server challenges the compute node, the measured data is sent to the attestation server.
4. The attestation server verifies the measurements against a known good database to

determine node trustworthiness.

A description of how to set up an attestation service is beyond the scope of this document. For an open source project that you can use to implement an attestation service, see the Open Attestation project.



Configuring Compute to use trusted compute pools

1. Enable scheduling support for trusted compute pools by adding these lines to the DEFAULT section of the `/etc/nova/nova.conf` file:

   ```
   [DEFAULT]
   compute_scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler
   scheduler_available_filters=nova.scheduler.filters.all_filters
   scheduler_default_filters=AvailabilityZoneFilter,RamFilter,ComputeFilter,T
   rustedFilter
   ```

2. Specify the connection information for your attestation service by adding these lines to the `trusted_computing` section of the `/etc/nova/nova.conf` file:

   ```
   [trusted_computing]
   attestation_server = 10.1.71.206
   attestation_port = 8443
   # If using OAT v2.0 after, use this port:
   # attestation_port = 8181
   attestation_server_ca_file = /etc/nova/ssl.10.1.71.206.crt
   # If using OAT v1.5, use this api_url:
   attestation_api_url = /AttestationService/resources
   ```

```
# If using OAT pre-v1.5, use this api_url:
# attestation_api_url = /OpenAttestationWebServices/V1.0
attestation_auth_blob = i-am-openstack
```

In this example:

server

Host name or IP address of the host that runs the attestation service

port

HTTPS port for the attestation service

server_ca_file

Certificate file used to verify the attestation server's identity

api_url

The attestation service's URL path

auth_blob

An authentication blob, required by the attestation service.

3. Save the file, and restart the `nova-compute` and `nova-scheduler` service to pick up the changes.

To customize the trusted compute pools, use these configuration option settings:

Description of trusted computing configuration options

Configuration option = Default value	Description
[trusted_computing]	
attestation_api_url = /OpenAttestationWebServices/V1.0	(StrOpt) Attestation web API URL
attestation_auth_blob = None	(StrOpt) Attestation authorization blob - must change
attestation_auth_timeout = 60	(IntOpt) Attestation status cache valid period length
attestation_insecure_ssl = False	(BoolOpt) Disable SSL cert verification

Configuration option = Default value	Description
[trusted_computing]	
	for Attestation service
attestation_port = 8443	(StrOpt) Attestation server port
attestation_server = None	(StrOpt) Attestation server HTTP
attestation_server_ca_file = None	(StrOpt) Attestation server Cert file for Identity verification

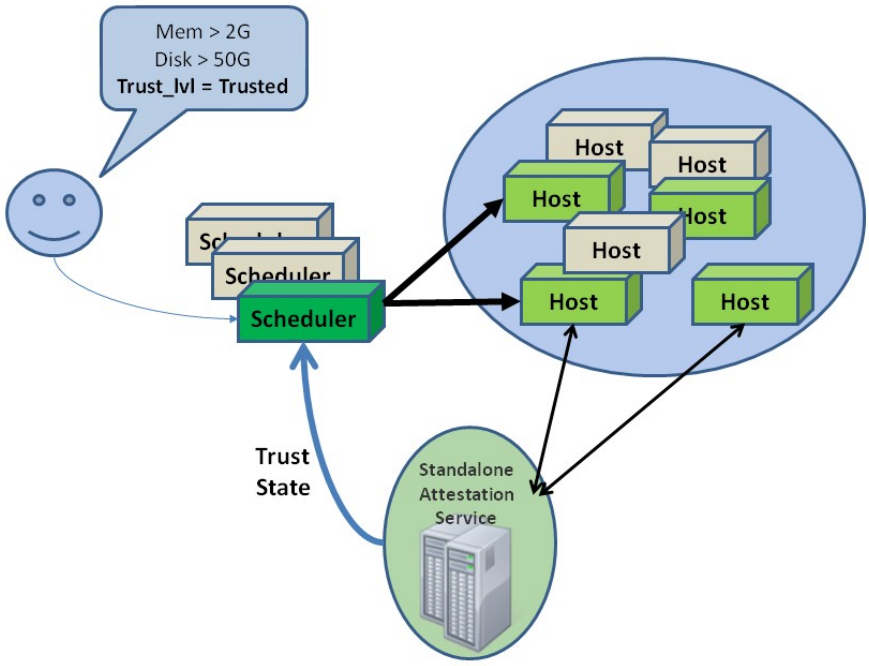
Specifying trusted flavors

1. Flavors can be designated as trusted using the **nova flavor-key set** command. In this example, the `m1.tiny` flavor is being set as trusted:

   ```
   $ nova flavor-key m1.tiny set trust:trusted_host=trusted
   ```

2. You can request that your instance is run on a trusted host by specifying a trusted flavor when booting the instance:

   ```
   $ nova boot --flavor m1.tiny --key_name myKeypairName --image myImageID
   newInstanceName
   ```



Encrypt Compute metadata traffic

Enabling SSL encryption

OpenStack supports encrypting Compute metadata traffic with HTTPS. Enable SSL encryption in the `metadata_agent.ini` file.

1. Enable the HTTPS protocol.

   ```
   nova_metadata_protocol = https
   ```

2. Determine whether insecure SSL connections are accepted for Compute metadata server requests. The default value is `False`.

   ```
   nova_metadata_insecure = False
   ```

3. Specify the path to the client certificate.

   ```
   nova_client_cert = PATH_TO_CERT
   ```

4. Specify the path to the private key.

   ```
   nova_client_priv_key = PATH_TO_KEY
   ```

Recover from a failed compute node

If you deploy Compute with a shared file system, you can use several methods to quickly recover from a node failure. This section discusses manual recovery.

Evacuate instances

If a hardware malfunction or other error causes the cloud compute node to fail, you can use the **nova evacuate** command to evacuate instances. See the OpenStack Administrator Guide.

Manual recovery

To manually recover a failed compute node:

1. Identify the VMs on the affected hosts by using a combination of the **nova list** and **nova show** commands or the **euca-describe-instances** command.

   For example, this command displays information about the i-000015b9 instance that runs on the np-rcc54 node:

   ```
   $ euca-describe-instances
   i-000015b9 at3-ui02 running nectarkey (376, np-rcc54) 0 m1.xxlarge 2012-
   06-19T00:48:11.000Z 115.146.93.60
   ```

2. Query the Compute database for the status of the host. This example converts an EC2 API instance ID to an OpenStack ID. If you use the **nova** commands, you can substitute the ID directly. This example output is truncated:

```
mysql> SELECT * FROM instances WHERE id = CONV('15b9', 16, 10) \G;
*************************** 1. row ***************************
created_at: 2012-06-19 00:48:11
updated_at: 2012-07-03 00:35:11
deleted_at: NULL
...
id: 5561
...
power_state: 5
vm_state: shutoff
...
hostname: at3-ui02
host: np-rcc54
...
uuid: 3f57699a-e773-4650-a443-b4b37eed5a06
...
task_state: NULL
...
```

> **Note**
>
> Find the credentials for your database in `/etc/nova.conf` file.

3. Decide to which compute host to move the affected VM. Run this database command to move the VM to that host:

```
mysql> UPDATE instances SET host = 'np-rcc46' WHERE uuid = '3f57699a-e773-
4650-a443-b4b37eed5a06';
```

4. If you use a hypervisor that relies on libvirt, such as KVM, update the `libvirt.xml` file in `/var/lib/nova/instances/[instance ID]` with these changes:
   - Change the `DHCPSERVER` value to the host IP address of the new compute host.
   - Update the VNC IP to `0.0.0.0`.
5. Reboot the VM:

```
$ nova reboot 3f57699a-e773-4650-a443-b4b37eed5a06
```

Typically, the database update and **nova reboot** command recover a VM from a failed host. However, if problems persist, try one of these actions:

- Use **virsh** to recreate the network filter configuration.
- Restart Compute services.
- Update the `vm_state` and `power_state` fields in the Compute database.

Recover from a UID/GID mismatch

Sometimes when you run Compute with a shared file system or an automated configuration tool, files on your compute node might use the wrong UID or GID. This UID or GID mismatch can prevent you from running live migrations or starting virtual machines.

This procedure runs on `nova-compute` hosts, based on the KVM hypervisor:

1. Set the nova UID to the same number in `/etc/passwd` on all hosts. For example, set the UID to `112`.

   > **Note**
   >
   > Choose UIDs or GIDs that are not in use for other users or groups.

2. Set the `libvirt-qemu` UID to the same number in the `/etc/passwd` file on all hosts. For example, set the UID to `119`.
3. Set the `nova` group to the same number in the `/etc/group` file on all hosts. For example, set the group to `120`.
4. Set the `libvirtd` group to the same number in the `/etc/group` file on all hosts. For example, set the group to `119`.
5. Stop the services on the compute node.
6. Change all files that the nova user or group owns. For example:

   ```
   # find / -uid 108 -exec chown nova {} \;
   # note the 108 here is the old nova UID before the change
   # find / -gid 120 -exec chgrp nova {} \;
   ```

7. Repeat all steps for the `libvirt-qemu` files, if required.
8. Restart the services.
9. To verify that all files use the correct IDs, run the **find** command.

Recover cloud after disaster

This section describes how to manage your cloud after a disaster and back up persistent storage volumes. Backups are mandatory, even outside of disaster scenarios.

For a definition of a disaster recovery plan (DRP), see http://en.wikipedia.org/wiki/Disaster_Recovery_Plan.

A disk crash, network loss, or power failure can affect several components in your cloud architecture. The worst disaster for a cloud is a power loss. A power loss affects these components:

- A cloud controller (`nova-api`, `nova-objectstore`, `nova-network`)
- A compute node (`nova-compute`)
- A storage area network (SAN) used by OpenStack Block Storage (`cinder-volumes`)

Before a power loss:

- Create an active iSCSI session from the SAN to the cloud controller (used for the `cinder-volumes` LVM's VG).
- Create an active iSCSI session from the cloud controller to the compute node (managed by `cinder-volume`).
- Create an iSCSI session for every volume (so 14 EBS volumes requires 14 iSCSI sessions).
- Create `iptables` or `ebtables` rules from the cloud controller to the compute node. This allows access from the cloud controller to the running instance.
- Save the current state of the database, the current state of the running instances, and the attached volumes (mount point, volume ID, volume status, etc), at least from the cloud controller to the compute node.

After power resumes and all hardware components restart:

- The iSCSI session from the SAN to the cloud no longer exists.
- The iSCSI session from the cloud controller to the compute node no longer exists.
- nova-network reapplies configurations on boot and, as a result, recreates the iptables and ebtables from the cloud controller to the compute node.
- Instances stop running.

  Instances are not lost because neither `destroy` nor `terminate` ran. The files for the instances remain on the compute node.
- The database does not update.

Begin recovery

> **Warning**
>
> Do not add any steps or change the order of steps in this procedure.

1. Check the current relationship between the volume and its instance, so that you can recreate the attachment.

   Use the **nova volume-list** command to get this information. Note that the **nova** client can get volume information from OpenStack Block Storage.

2. Update the database to clean the stalled state. Do this for every volume by using these queries:

   ```
   mysql> use cinder;
   mysql> update volumes set mountpoint=NULL;
   mysql> update volumes set status="available" where status
   <>"error_deleting";
   mysql> update volumes set attach_status="detached";
   mysql> update volumes set instance_id=0;
   ```

   Use **nova volume-list** command to list all volumes.

3. Restart the instances by using the **nova reboot INSTANCE** command.

> **Important**
>
> Some instances completely reboot and become reachable, while some might stop at the plymouth stage. This is expected behavior. DO NOT reboot a second time.
>
> Instance state at this stage depends on whether you added an /etc/fstab entry for that volume. Images built with the cloud-init package remain in a `pending` state, while others skip the missing volume and start. You perform this step to ask Compute to reboot every instance so that the stored state is preserved. It does not matter if not all instances come up successfully. For more information about cloud-init, see help.ubuntu.com/community/CloudInit/.

4. If required, run the **nova volume-attach** command to reattach the volumes to their respective instances. This example uses a file of listed volumes to reattach them:

```
#!/bin/bash

while read line; do
    volume=`echo $line | $CUT -f 1 -d " "`
    instance=`echo $line | $CUT -f 2 -d " "`
    mount_point=`echo $line | $CUT -f 3 -d " "`
        echo "ATTACHING VOLUME FOR INSTANCE - $instance"
    nova volume-attach $instance $volume $mount_point
    sleep 2
done < $volumes_tmp_file
```

Instances that were stopped at the plymouth stage now automatically continue booting and start normally. Instances that previously started successfully can now see the volume.

5. Log in to the instances with SSH and reboot them.

   If some services depend on the volume or if a volume has an entry in fstab, you can now restart the instance. Restart directly from the instance itself and not through **nova**:

   ```
   # shutdown -r now
   ```

   When you plan for and complete a disaster recovery, follow these tips:

- Use the `errors=remount` option in the `fstab` file to prevent data corruption.

  In the event of an I/O error, this option prevents writes to the disk. Add this configuration option into the cinder-volume server that performs the iSCSI connection to the SAN and into the instances' `fstab` files.

- Do not add the entry for the SAN's disks to the cinder-volume's `fstab` file.

  Some systems hang on that step, which means you could lose access to your cloud-controller. To re-run the session manually, run this command before performing the mount:

  ```
  # iscsiadm -m discovery -t st -p $SAN_IP $ iscsiadm -m node --target-name $IQN -p $SAN_IP -l
  ```

- On your instances, if you have the whole `/home/` directory on the disk, leave a user's directory with the user's bash files and the `authorized_keys` file instead of emptying the `/home/` directory and mapping the disk on it.

  This action enables you to connect to the instance without the volume attached, if you allow only connections through public keys.

To script the disaster recovery plan (DRP), use the https://github.com/Razique bash script.

This script completes these steps:

1. Creates an array for instances and their attached volumes.
2. Updates the MySQL database.
3. Restarts all instances with euca2ools.
4. Reattaches the volumes.
5. Uses Compute credentials to make an SSH connection into every instance.

The script includes a `test mode`, which enables you to perform the sequence for only one instance.

To reproduce the power loss, connect to the compute node that runs that instance and close the iSCSI session. Do not detach the volume by using the **nova volume-detach** command. You must manually close the iSCSI session. This example closes an iSCSI session with the number `15`:

```
# iscsiadm -m session -u -r 15
```

Do not forget the *-r* option. Otherwise, all sessions close.

> **Warning**
>
> There is potential for data loss while running instances during this procedure. If you are using Liberty or earlier, ensure you have the correct patch and set the options appropriately.

Advanced configuration

OpenStack clouds run on platforms that differ greatly in the capabilities that they provide. By default, the Compute service seeks to abstract the underlying hardware that it runs on, rather than exposing specifics about the underlying host platforms. This abstraction manifests itself in many ways. For example, rather than exposing the types and topologies of CPUs running on hosts, the service exposes a number of generic CPUs (virtual CPUs, or vCPUs) and allows for overcommitting of these. In a similar manner, rather than exposing the individual types of network devices available on hosts, generic software-powered network ports are provided. These features are designed to allow high resource utilization and allows the service to provide a generic cost-effective and highly-scalable cloud upon which to build applications.

This abstraction is beneficial for most workloads. However, there are some workloads where determinism and per-instance performance are important, if not vital. In these cases, instances can be expected to deliver near-native performance. The Compute service provides features to improve individual instance for these kind of workloads.

Attaching physical PCI devices to guests

The PCI passthrough feature in OpenStack allows full access and direct control of a physical PCI device in guests. This mechanism is generic for any kind of PCI devices (for example: Network Interface Card (NIC), Graphics Processing Unit (GPU) or any other devices that can be attached to a PCI bus). Correct driver installation is the only requirement for the guest to properly use the devices.

PCI devices, if supported by the hardware, can provide Single Root I/O Virtualization and Sharing (SR-IOV) functionality. In this case, a physical device is virtualized and appears as multiple PCI devices. Virtual PCI devices are assigned to the same or different guests. In the case of PCI passthrough, the full physical device is assigned to only one guest and cannot be shared.

> **Note**
>
> For Attaching virtual SR-IOV devices to guests, refer to the Networking Guide.

To enable PCI passthrough, follow the steps below:

1. Configure nova-scheduler (Controller)
2. Configure nova-api (Controller)**
3. Configure a flavor (Controller)
4. Enable PCI passthrough (Compute)
5. Configure PCI devices in nova-compute (Compute)

> **Note**
>
> The PCI device with address `0000:41:00.0` is as an example. Expect to change this according to your actual environment.

Configure nova-scheduler (Controller)

1. Configure `nova-scheduler` as specified in Configure nova-scheduler.
2. Restart `nova-scheduler` with **service nova-scheduler restart**.

Configure nova-api (Controller)

1. Specify the PCI alias for the device.

   Configure a PCI alias `a1` to request a PCI device with a `vendor_id` of `0x8086` and a `product_id` of `0x154d`. The `vendor_id` and `product_id` correspond the PCI device with address `0000:41:00.0`.

   Edit `/etc/nova/nova.conf`:

   ```
   [default]
   pci_alias = { "vendor_id":"8086", "product_id":"154d",
   "device_type":"type-PF", "name":"a1" }
   ```

   For more information about the syntax of `pci_alias`, refer to nova.conf configuration options.

2. Restart `nova-api` with **service nova-api restart**.

Configure a flavor (Controller)

Configure a flavor to request two PCI devices, each with `vendor_id` as `0x8086` and `product_id` as `0x154d`.

```
# openstack flavor set m1.large --property "pci_passthrough:alias"="a1:2"
```

For more information about the syntax for `pci_passthrough:alias`, refer to flavor.

Enable PCI passthrough (Compute)

Enable VT-d and IOMMU. For more information, refer to steps one and two in Create Virtual Functions.

Configure PCI devices `nova-compute` (Compute)

1. Configure `nova-compute` to allow the PCI device to be passed through to VMs. Edit `/etc/nova/nova.conf`:

   ```
   [default]
   pci_passthrough_whitelist = { "address": "0000:41:00.0" }
   ```

   Alternatively specify multiple PCI devices using whitelisting:

   ```
   [default]
   pci_passthrough_whitelist = { "vendor_id": "8086", "product_id": "10fb" }
   ```

All PCI devices matching the `vendor_id` and `product_id` are added to the pool of PCI devices available for passthrough to VMs.

For more information about the syntax of `pci_passthrough_whitelist`, refer to nova.conf configuration options.

2. Specify the PCI alias for the device.

   From the Newton release, to resize guest with PCI device, configure the PCI alias on the compute node as well.

   Configure a PCI alias `a1` to request a PCI device with a `vendor_id` of `0x8086` and a `product_id` of `0x154d`. The `vendor_id` and `product_id` correspond the PCI device with address `0000:41:00.0`.

   Edit `/etc/nova/nova.conf`:

   ```
   [default]
   pci_alias = { "vendor_id":"8086", "product_id":"154d",
   "device_type":"type-PF", "name":"a1" }
   ```

   For more information about the syntax of `pci_alias`, refer to nova.conf configuration options.

3. Restart `nova-compute` with **service nova-compute restart**.

Create instances with PCI passthrough devices

The `nova-scheduler` selects a destination host that has PCI devices available with the specified `vendor_id` and `product_id` that matches the `pci_alias` from the flavor.

```
# openstack server create --flavor m1.large --image cirros-0.3.4-x86_64-uec
--wait test-pci
```

Enabling advanced CPU topologies in guests

The NUMA topology and CPU pinning features in OpenStack provide high level control over how instances run on host CPUs, and the topology of CPUs inside the instance. These features can be used to minimize latency and maximize per-instance performance.

SMP, NUMA, and SMT overviews

Symmetric multiprocessing (SMP) is a design found in many modern multi-core systems. In an SMP system, there are two or more CPUs and these CPUs are connected by some interconnect. This provides CPUs with equal access to system resources like memory and IO ports.

Non-uniform memory access (NUMA) is a derivative of the SMP design that is found in many multi-socket systems. In a NUMA system, system memory is divided into cells or

nodes that are associated with particular CPUs. Requests for memory on other nodes are possible through an interconnect bus, however, bandwidth across this shared bus is limited. As a result, competition for this this resource can incur performance penalties.

Simultaneous Multi-Threading (SMT), known as as Hyper-Threading on Intel platforms, is a design that is complementary to SMP. Whereas CPUs in SMP systems share a bus and some memory, CPUs in SMT systems share many more components. CPUs that share components are known as thread siblings. All CPUs appear as usable CPUs on the system and can execute workloads in parallel, however, as with NUMA, threads compete for shared resources.

Customizing instance NUMA topologies

> **Important**
>
> The functionality described below is currently only supported by the libvirt/KVM driver.

When running workloads on NUMA hosts, it is important that the vCPUs executing processes are on the same NUMA node as the memory used by these processes. This ensures all memory accesses are local to the node and thus do not consume the limited cross-node memory bandwidth, adding latency to memory accesses. Similarly, large pages are assigned from memory and benefit from the same performance improvements as memory allocated using standard pages, thus, they also should be local. Finally, PCI devices are directly associated with specific NUMA nodes for the purposes of DMA. Instances that use PCI or SR-IOV devices should be placed on the NUMA node associated with the said devices.

By default, an instance will float across all NUMA nodes on a host. NUMA awareness can be enabled implicitly, through the use of hugepages or pinned CPUs, or explicitly, through the use of flavor extra specs or image metadata. In all cases, the `NUMATopologyFilter` filter must be enabled. Details on this filter are provided in Scheduling configuration guide.

> **Caution**
>
> The NUMA node(s) used are normally chosen at random. However, if a PCI passthrough or SR-IOV device is attached to the instance, then the NUMA node that the device is associated with will be used. This can provide important performance improvements, however, booting a large number of similar instances can result in unbalanced NUMA node usage. Care should be taken to mitigate this issue. See this discussion for more details.

> **Caution**
>
> Inadequate per-node resources will result in scheduling failures. Resources that are specific to a node include not only CPUs and memory, but also PCI and SR-IOV resources. It is not possible to use multiple resources from different nodes without requesting a multi-node layout. As such, it may be necessary to ensure PCI or SR-IOV resources are associated with the same NUMA node or force a multi-node layout.

When used, NUMA awareness allows the operating system of the instance to intelligently schedule the workloads that it runs and minimize cross-node memory bandwidth. To restrict an instance's vCPUs to a single host NUMA node, run:

```
# openstack flavor set m1.large --property hw:numa_nodes=1
```

Some workloads have very demanding requirements for memory access latency or bandwidth which exceed the memory bandwidth available from a single NUMA node. For such workloads, it is beneficial to spread the instance across multiple host NUMA nodes, even if the instance's RAM/vCPUs could theoretically fit on a single NUMA node. To force an instance's vCPUs to spread across two host NUMA nodes, run:

```
# openstack flavor set m1.large --property hw:numa_nodes=2
```

The allocation of instances vCPUs and memory from different host NUMA nodes can be configured. This allows for asymmetric allocation of vCPUs and memory, which can be important for some workloads. To spread the six vCPUs and 6 GB of memory of an instance across two NUMA nodes and create an asymmetric 1:2 vCPU and memory mapping between the two nodes, run:

```
# openstack flavor set m1.large --property hw:numa_nodes=2
# openstack flavor set m1.large \  # configure guest node 0
    --property hw:numa_cpus.0=0,1 \
    --property hw:numa_mem.0=2048
# openstack flavor set m1.large \  # configure guest node 1
    --property hw:numa_cpus.1=2,3,4,5 \
    --property hw:numa_mem.1=4096
```

For more information about the syntax for `hw:numa_nodes`, `hw:numa_cpus.N` and `hw:num_mem.N`, refer to the Flavors guide.

Customizing instance CPU policies

> **Important**
>
> The functionality described below is currently only supported by the libvirt/KVM driver.

By default, instance vCPU processes are not assigned to any particular host CPU, instead, they float across host CPUs like any other process. This allows for features like overcommitting of CPUs. In heavily contended systems, this provides optimal system performance at the expense of performance and latency for individual instances.

Some workloads require real-time or near real-time behavior, which is not possible with the latency introduced by the default CPU policy. For such workloads, it is beneficial to control which host CPUs are bound to an instance's vCPUs. This process is known as pinning. No instance with pinned CPUs can use the CPUs of another pinned instance, thus preventing resource contention between instances. To configure a flavor to use pinned vCPUs, a use a dedicated CPU policy. To force this, run:

```
# openstack flavor set m1.large --property hw:cpu_policy=dedicated
```

> **Caution**
>
> Host aggregates should be used to separate pinned instances from unpinned instances as the latter will not respect the resourcing requirements of the former.

When running workloads on SMT hosts, it is important to be aware of the impact that thread siblings can have. Thread siblings share a number of components and contention on these components can impact performance. To configure how to use threads, a CPU thread policy should be specified. For workloads where sharing benefits performance, use thread siblings. To force this, run:

```
# openstack flavor set m1.large \
    --property hw:cpu_policy=dedicated \
    --property hw:cpu_thread_policy=require
```

For other workloads where performance is impacted by contention for resources, use non-thread siblings or non-SMT hosts. To force this, run:

```
# openstack flavor set m1.large \
    --property hw:cpu_policy=dedicated \
    --property hw:cpu_thread_policy=isolate
```

Finally, for workloads where performance is minimally impacted, use thread siblings if available. This is the default, but it can be set explicitly:

```
# openstack flavor set m1.large \
    --property hw:cpu_policy=dedicated \
    --property hw:cpu_thread_policy=prefer
```

For more information about the syntax for `hw:cpu_policy` and `hw:cpu_thread_policy`, refer to the Flavors guide.

Applications are frequently packaged as images. For applications that require real-time or near real-time behavior, configure image metadata to ensure created instances are always pinned regardless of flavor. To configure an image to use pinned vCPUs and avoid thread siblings, run:

```
# openstack image set [IMAGE_ID] \
    --property hw_cpu_policy=dedicated \
    --property hw_cpu_thread_policy=isolate
```

Image metadata takes precedence over flavor extra specs: configuring competing policies will result in an exception. By setting a `shared` policy through image metadata, administrators can prevent users configuring CPU policies in flavors and impacting resource utilization. To configure this policy, run:

```
# openstack image set [IMAGE_ID] --property hw_cpu_policy=shared
```

> **Note**
>
> There is no correlation required between the NUMA topology exposed in the instance and how the instance is actually pinned on the host. This is by design. See this bug for more information.

For more information about image metadata, refer to the Image metadata guide.

Troubleshoot Compute

Common problems for Compute typically involve misconfigured networking or credentials that are not sourced properly in the environment. Also, most flat networking configurations do not enable **ping** or **ssh** from a compute node to the instances that run on that node. Another common problem is trying to run 32-bit images on a 64-bit compute node. This section shows you how to troubleshoot Compute.

Compute service logging

Compute stores a log file for each service in `/var/log/nova`. For example, `nova-compute.log` is the log for the `nova-compute` service. You can set the following options to format log strings for the `nova.log` module in the `nova.conf` file:

- `logging_context_format_string`
- `logging_default_format_string`

If the log level is set to `debug`, you can also specify `logging_debug_format_suffix` to append extra formatting. For information about what variables are available for the

formatter see http://docs.python.org/library/logging.html#formatter-objects.

You have two logging options for OpenStack Compute based on configuration settings. In `nova.conf`, include the `logfile` option to enable logging. Alternatively you can set `use_syslog = 1` so that the nova daemon logs to syslog.

Guru Meditation reports

A Guru Meditation report is sent by the Compute service upon receipt of the `SIGUSR2` signal (`SIGUSR1` before Mitaka). This report is a general-purpose error report that includes details about the current state of the service. The error report is sent to `stderr`.

For example, if you redirect error output to `nova-api-err.log` using **nova-api 2>/var/log/nova/nova-api-err.log**, resulting in the process ID 8675, you can then run:

```
# kill -USR2 8675
```

This command triggers the Guru Meditation report to be printed to `/var/log/nova/nova-api-err.log`.

The report has the following sections:

- Package: Displays information about the package to which the process belongs, including version information.
- Threads: Displays stack traces and thread IDs for each of the threads within the process.
- Green Threads: Displays stack traces for each of the green threads within the process (green threads do not have thread IDs).
- Configuration: Lists all configuration options currently accessible through the CONF object for the current process.

For more information, see Guru Meditation Reports.

Common errors and fixes for Compute

The ask.openstack.org site offers a place to ask and answer questions, and you can also mark questions as frequently asked questions. This section describes some errors people have posted previously. Bugs are constantly being fixed, so online resources are a great way to get the most up-to-date errors and fixes.

Credential errors, 401, and 403 forbidden errors

Problem

Missing credentials cause a `403 forbidden` error.

Solution

To resolve this issue, use one of these methods:

1. **Manual method**

   Gets the `novarc` file from the project ZIP file, saves existing credentials in case of override, and manually sources the `novarc` file.

2. **Script method**

   Generates `novarc` from the project ZIP file and sources it for you.

When you run `nova-api` the first time, it generates the certificate authority information, including `openssl.cnf`. If you start the CA services before this, you might not be able to create your ZIP file. Restart the services. When your CA information is available, create your ZIP file.

Also, check your HTTP proxy settings to see whether they cause problems with `novarc` creation.

Instance errors

Problem

Sometimes a particular instance shows `pending` or you cannot SSH to it. Sometimes the image itself is the problem. For example, when you use flat manager networking, you do not have a DHCP server and certain images do not support interface injection; you cannot connect to them.

Solution

To fix instance errors use an image that does support this method, such as Ubuntu, which obtains an IP address correctly with FlatManager network settings.

To troubleshoot other possible problems with an instance, such as an instance that stays in a spawning state, check the directory for the particular instance under `/var/lib/nova/instances` on the `nova-compute` host and make sure that these files are present:

- `libvirt.xml`
- `disk`
- `disk-raw`
- `kernel`
- `ramdisk`
- `console.log`, after the instance starts.

If any files are missing, empty, or very small, the `nova-compute` service did not successfully download the images from the Image service.

Also check `nova-compute.log` for exceptions. Sometimes they do not appear in the console output.

Next, check the log file for the instance in the `/var/log/libvirt/qemu` directory to see if it exists and has any useful error messages in it.

Finally, from the `/var/lib/nova/instances` directory for the instance, see if this command returns an error:

```
# virsh create libvirt.xml
```

Empty log output for Linux instances

Problem

You can view the log output of running instances from either the *Log* tab of the dashboard or the output of **nova console-log**. In some cases, the log output of a running Linux instance will be empty or only display a single character (for example, the ? character).

This occurs when the Compute service attempts to retrieve the log output of the instance via a serial console while the instance itself is not configured to send output to the console.

Solution

To rectify this, append the following parameters to kernel arguments specified in the instance's boot loader:

```
console=tty0 console=ttyS0,115200n8
```

Upon rebooting, the instance will be configured to send output to the Compute service.

Reset the state of an instance

Problem

Instances can remain in an intermediate state, such as `deleting`.

Solution

You can use the **nova reset-state** command to manually reset the state of an instance to an error state. You can then delete the instance. For example:

```
$ nova reset-state c6bbbf26-b40a-47e7-8d5c-eb17bf65c485
```

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

You can also use the *--active* parameter to force the instance back to an active state instead of an error state. For example:

```
$ nova reset-state --active c6bbbf26-b40a-47e7-8d5c-eb17bf65c485
```

Injection problems

Problem

Instances may boot slowly, or do not boot. File injection can cause this problem.

Solution

To disable injection in libvirt, set the following in `nova.conf`:

```
[libvirt]
inject_partition = -2
```

> **Note**
>
> If you have not enabled the configuration drive and you want to make user-specified files available from the metadata server for to improve performance and avoid boot failure if injection fails, you must disable injection.

Disable live snapshotting

Problem

Administrators using libvirt version `1.2.2` may experience problems with live snapshot creation. Occasionally, libvirt version `1.2.2` fails to create live snapshots under the load of creating concurrent snapshot.

Solution

To effectively disable the libvirt live snapshotting, until the problem is resolved, configure the `disable_libvirt_livesnapshot` option. You can turn off the live snapshotting mechanism by setting up its value to `True` in the `[workarounds]` section of the `nova.conf` file:

```
[workarounds]
disable_libvirt_livesnapshot = True
```

Object Storage

Introduction to Object Storage

OpenStack Object Storage (swift) is used for redundant, scalable data storage using clusters of standardized servers to store petabytes of accessible data. It is a long-term storage system for large amounts of static data which can be retrieved and updated. Object Storage uses a distributed architecture with no central point of control, providing greater scalability, redundancy, and permanence. Objects are written to multiple hardware devices, with the OpenStack software responsible for ensuring data replication and integrity across the cluster. Storage clusters scale horizontally by adding new nodes. Should a node fail, OpenStack works to replicate its content from other active nodes. Because OpenStack uses software logic to ensure data replication and distribution across different devices, inexpensive commodity hard drives and servers can be used in lieu of more expensive equipment.

Object Storage is ideal for cost effective, scale-out storage. It provides a fully distributed, API-accessible storage platform that can be integrated directly into applications or used for backup, archiving, and data retention.

Features and benefits

Features	Benefits
Leverages commodity hardware	No lock-in, lower price/GB.
HDD/node failure agnostic	Self-healing, reliable, data redundancy protects from failures.
Unlimited storage	Large and flat namespace, highly scalable read/write access, able to serve content directly from storage system.
Multi-dimensional scalability	Scale-out architecture: Scale vertically and horizontally-distributed storage. Backs up and archives large amounts of data with linear performance.

Features	Benefits
Account/container/object structure	No nesting, not a traditional file system: Optimized for scale, it scales to multiple petabytes and billions of objects.
Built-in replication 3× + data redundancy (compared with 2× on RAID)	A configurable number of accounts, containers and object copies for high availability.
Easily add capacity (unlike RAID resize)	Elastic data scaling with ease.
No central database	Higher performance, no bottlenecks.
RAID not required	Handle many small, random reads and writes efficiently.
Built-in management utilities	Account management: Create, add, verify, and delete users; Container management: Upload, download, and verify; Monitoring: Capacity, host, network, log trawling, and cluster health.
Drive auditing	Detect drive failures preempting data corruption.
Expiring objects	Users can set an expiration time or a TTL on an object to control access.
Direct object access	Enable direct browser access to content, such as for a control panel.
Realtime visibility into client requests	Know what users are requesting.
Supports S3 API	Utilize tools that were designed for the popular S3 API.
Restrict containers per account	Limit access to control usage by user.
Support for NetApp, Nexenta, Solidfire	Unified support for block volumes using a variety of storage systems.
Snapshot and backup API for block volumes.	Data protection and recovery for VM data.
Standalone volume API available	Separate endpoint and API for integration with other compute systems.

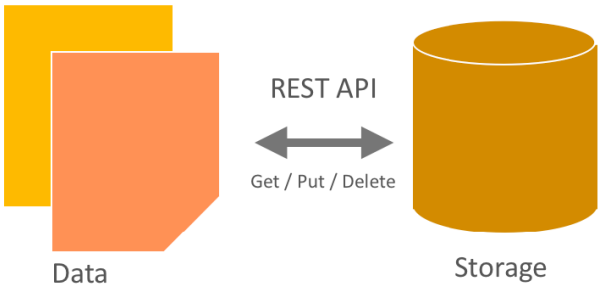
Features	Benefits
Integration with Compute	Fully integrated with Compute for attaching block volumes and reporting on usage.

Object Storage characteristics

The key characteristics of Object Storage are that:

- All objects stored in Object Storage have a URL.
- All objects stored are replicated 3× in as-unique-as-possible zones, which can be defined as a group of drives, a node, a rack, and so on.
- All objects have their own metadata.
- Developers interact with the object storage system through a RESTful HTTP API.
- Object data can be located anywhere in the cluster.
- The cluster scales by adding additional nodes without sacrificing performance, which allows a more cost-effective linear storage expansion than fork-lift upgrades.
- Data does not have to be migrated to an entirely new storage system.
- New nodes can be added to the cluster without downtime.
- Failed nodes and disks can be swapped out without downtime.
- It runs on industry-standard hardware, such as Dell, HP, and Supermicro.

Object Storage (swift)



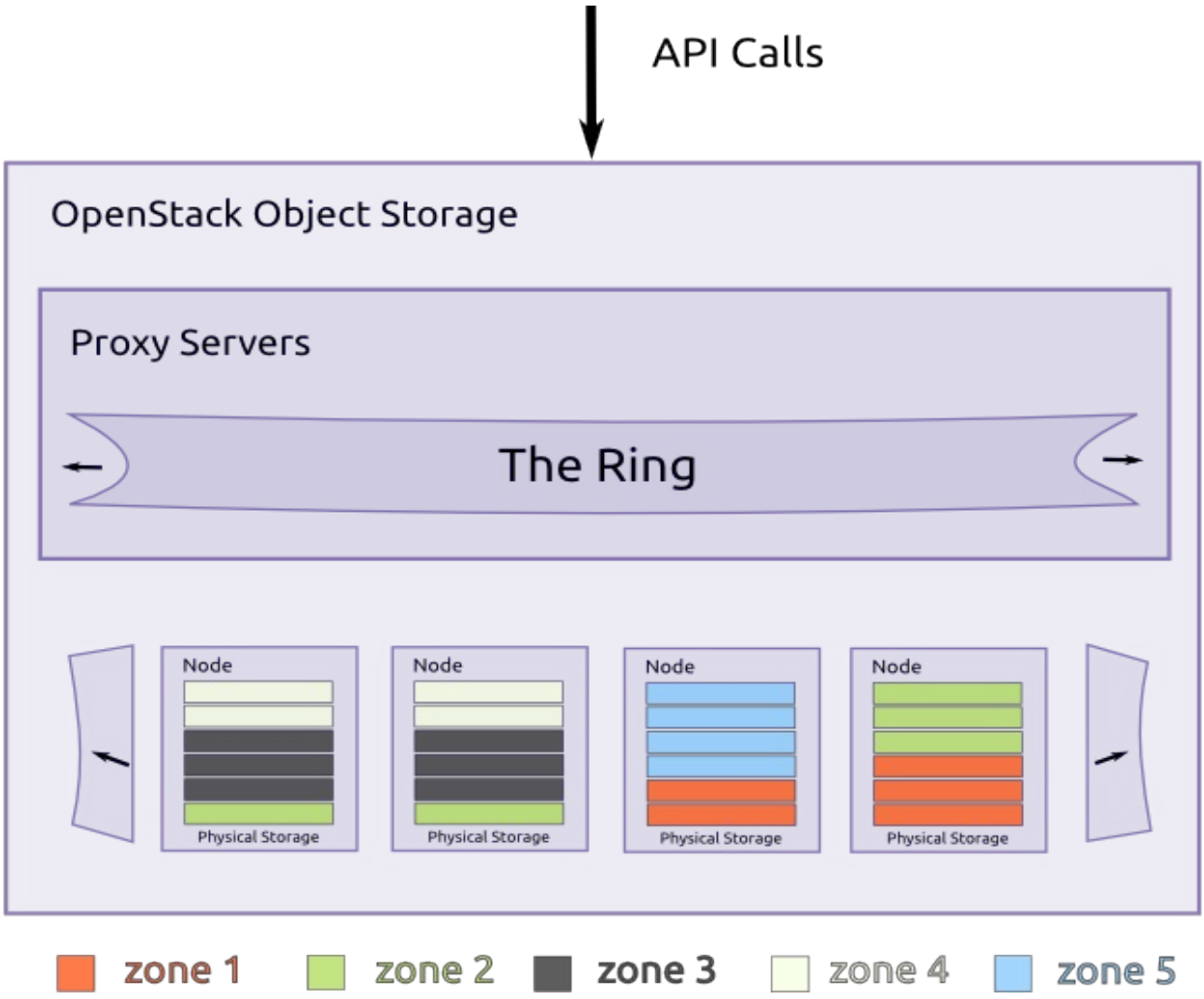
Developers can either write directly to the Swift API or use one of the many client libraries that exist for all of the popular programming languages, such as Java, Python, Ruby, and C#. Amazon S3 and RackSpace Cloud Files users should be very familiar with Object Storage. Users new to object storage systems will have to adjust to a different approach and mindset than those required for a traditional filesystem.

Components

Object Storage uses the following components to deliver high availability, high durability, and high concurrency:

- **Proxy servers** - Handle all of the incoming API requests.
- **Rings** - Map logical names of data to locations on particular disks.
- **Zones** - Isolate data from other zones. A failure in one zone does not impact the rest of the cluster as data replicates across zones.
- **Accounts and containers** - Each account and container are individual databases that are distributed across the cluster. An account database contains the list of containers in that account. A container database contains the list of objects in that container.
- **Objects** - The data itself.
- **Partitions** - A partition stores objects, account databases, and container databases and helps manage locations where data lives in the cluster.

Object Storage building blocks



Proxy servers

Proxy servers are the public face of Object Storage and handle all of the incoming API requests. Once a proxy server receives a request, it determines the storage node based on the object's URL, for example: https://swift.example.com/v1/account/container/object. Proxy servers also coordinate responses, handle failures, and coordinate timestamps.

Proxy servers use a shared-nothing architecture and can be scaled as needed based on projected workloads. A minimum of two proxy servers should be deployed for redundancy. If one proxy server fails, the others take over.

For more information concerning proxy server configuration, see Configuration Reference.

Rings

A ring represents a mapping between the names of entities stored on disks and their physical locations. There are separate rings for accounts, containers, and objects. When other components need to perform any operation on an object, container, or account, they need to interact with the appropriate ring to determine their location in the cluster.

The ring maintains this mapping using zones, devices, partitions, and replicas. Each partition in the ring is replicated, by default, three times across the cluster, and partition locations are stored in the mapping maintained by the ring. The ring is also responsible for determining which devices are used for handoff in failure scenarios.

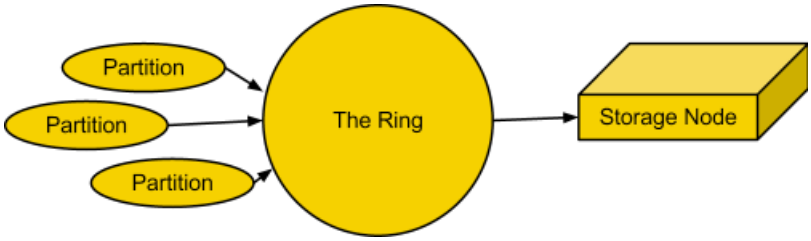
Data can be isolated into zones in the ring. Each partition replica is guaranteed to reside in a different zone. A zone could represent a drive, a server, a cabinet, a switch, or even a data center.

The partitions of the ring are equally divided among all of the devices in the Object Storage installation. When partitions need to be moved around (for example, if a device is added to the cluster), the ring ensures that a minimum number of partitions are moved at a time, and only one replica of a partition is moved at a time.

You can use weights to balance the distribution of partitions on drives across the cluster. This can be useful, for example, when differently sized drives are used in a cluster.

The ring is used by the proxy server and several background processes (like replication).

The ring



These rings are externally managed. The server processes themselves do not modify the rings, they are instead given new rings modified by other tools.

The ring uses a configurable number of bits from an `MD5` hash for a path as a partition index that designates a device. The number of bits kept from the hash is known as the partition power, and 2 to the partition power indicates the partition count. Partitioning the full `MD5` hash ring allows other parts of the cluster to work in batches of items at once which ends up either more efficient or at least less complex than working with each item separately or the entire cluster all at once.

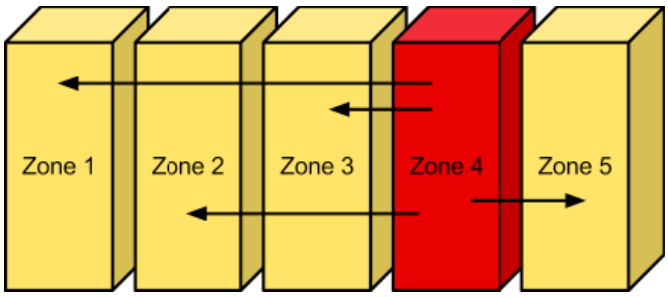
Another configurable value is the replica count, which indicates how many of the partition-device assignments make up a single ring. For a given partition number, each replica's device will not be in the same zone as any other replica's device. Zones can be used to group devices based on physical locations, power separations, network separations, or any other attribute that would improve the availability of multiple replicas at the same time.

Zones

Object Storage allows configuring zones in order to isolate failure boundaries. If possible,

each data replica resides in a separate zone. At the smallest level, a zone could be a single drive or a grouping of a few drives. If there were five object storage servers, then each server would represent its own zone. Larger deployments would have an entire rack (or multiple racks) of object servers, each representing a zone. The goal of zones is to allow the cluster to tolerate significant outages of storage servers without losing all replicas of the data.

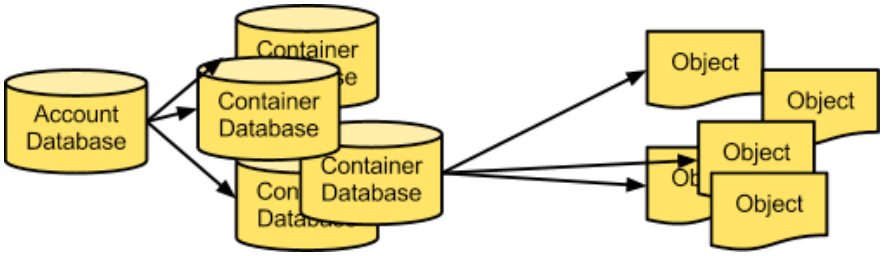
Zones



Accounts and containers

Each account and container is an individual SQLite database that is distributed across the cluster. An account database contains the list of containers in that account. A container database contains the list of objects in that container.

Accounts and containers



To keep track of object data locations, each account in the system has a database that references all of its containers, and each container database references each object.

Partitions

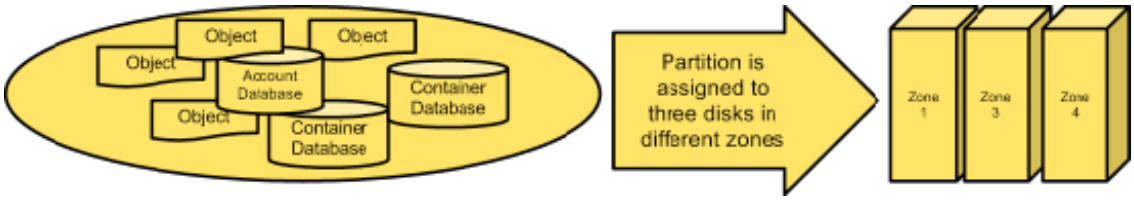
A partition is a collection of stored data. This includes account databases, container databases, and objects. Partitions are core to the replication system.

Think of a partition as a bin moving throughout a fulfillment center warehouse. Individual orders get thrown into the bin. The system treats that bin as a cohesive entity as it moves throughout the system. A bin is easier to deal with than many little things. It makes for fewer moving parts throughout the system.

System replicators and object uploads/downloads operate on partitions. As the system scales up, its behavior continues to be predictable because the number of partitions is a fixed number.

Implementing a partition is conceptually simple, a partition is just a directory sitting on a disk with a corresponding hash table of what it contains.

Partitions



Replicators

In order to ensure that there are three copies of the data everywhere, replicators continuously examine each partition. For each local partition, the replicator compares it against the replicated copies in the other zones to see if there are any differences.

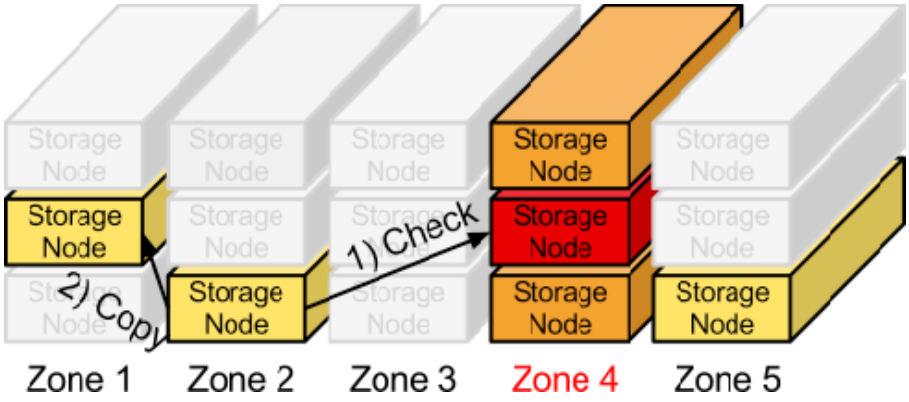
The replicator knows if replication needs to take place by examining hashes. A hash file is created for each partition, which contains hashes of each directory in the partition. Each of the three hash files is compared. For a given partition, the hash files for each of the partition's copies are compared. If the hashes are different, then it is time to replicate, and

the directory that needs to be replicated is copied over.

This is where partitions come in handy. With fewer things in the system, larger chunks of data are transferred around (rather than lots of little TCP connections, which is inefficient) and there is a consistent number of hashes to compare.

The cluster eventually has a consistent behavior where the newest data has a priority.

Replication



If a zone goes down, one of the nodes containing a replica notices and proactively copies data to a handoff location.

Use cases

The following sections show use cases for object uploads and downloads and introduce the components.

Upload

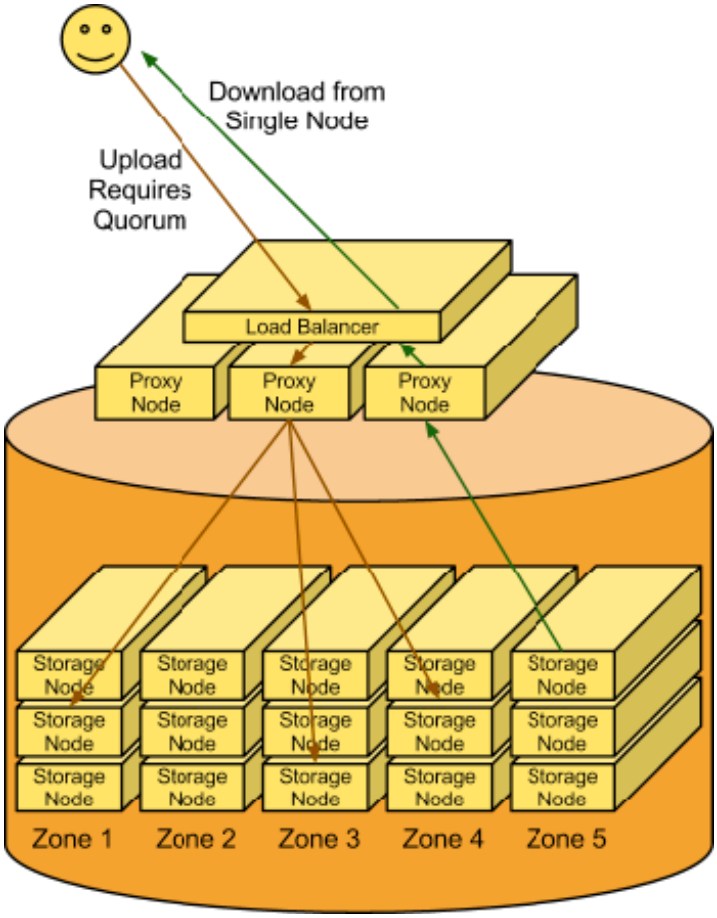
A client uses the REST API to make a HTTP request to PUT an object into an existing container. The cluster receives the request. First, the system must figure out where the data is going to go. To do this, the account name, container name, and object name are all used to determine the partition where this object should live.

Then a lookup in the ring figures out which storage nodes contain the partitions in question.

The data is then sent to each storage node where it is placed in the appropriate partition. At least two of the three writes must be successful before the client is notified that the upload was successful.

Next, the container database is updated asynchronously to reflect that there is a new object in it.

Object Storage in use



Download

A request comes in for an account/container/object. Using the same consistent hashing, the partition name is generated. A lookup in the ring reveals which storage nodes contain that partition. A request is made to one of the storage nodes to fetch the object and, if that fails, requests are made to the other nodes.

Ring-builder

Use the swift-ring-builder utility to build and manage rings. This utility assigns partitions to devices and writes an optimized Python structure to a gzipped, serialized file on disk for transmission to the servers. The server processes occasionally check the modification time of the file and reload in-memory copies of the ring structure as needed. If you use a slightly older version of the ring, one of the three replicas for a partition subset will be incorrect because of the way the ring-builder manages changes to the ring. You can work around this issue.

The ring-builder also keeps its own builder file with the ring information and additional data required to build future rings. It is very important to keep multiple backup copies of these builder files. One option is to copy the builder files out to every server while copying

the ring files themselves. Another is to upload the builder files into the cluster itself. If you lose the builder file, you have to create a new ring from scratch. Nearly all partitions would be assigned to different devices and, therefore, nearly all of the stored data would have to be replicated to new locations. So, recovery from a builder file loss is possible, but data would be unreachable for an extended time.

Ring data structure

The ring data structure consists of three top level fields: a list of devices in the cluster, a list of lists of device ids indicating partition to device assignments, and an integer indicating the number of bits to shift an MD5 hash to calculate the partition for the hash.

Partition assignment list

This is a list of `array('H')` of devices ids. The outermost list contains an `array('H')` for each replica. Each `array('H')` has a length equal to the partition count for the ring. Each integer in the `array('H')` is an index into the above list of devices. The partition list is known internally to the Ring class as `_replica2part2dev_id`.

So, to create a list of device dictionaries assigned to a partition, the Python code would look like:

```
devices = [self.devs[part2dev_id[partition]] for
part2dev_id in self._replica2part2dev_id]
```

That code is a little simplistic because it does not account for the removal of duplicate devices. If a ring has more replicas than devices, a partition will have more than one replica on a device.

`array('H')` is used for memory conservation as there may be millions of partitions.

Overload

The ring builder tries to keep replicas as far apart as possible while still respecting device weights. When it can not do both, the overload factor determines what happens. Each device takes an extra fraction of its desired partitions to allow for replica dispersion; after that extra fraction is exhausted, replicas are placed closer together than optimal.

The overload factor lets the operator trade off replica dispersion (durability) against data dispersion (uniform disk usage).

The default overload factor is 0, so device weights are strictly followed.

With an overload factor of 0.1, each device accepts 10% more partitions than it otherwise would, but only if it needs to maintain partition dispersion.

For example, consider a 3-node cluster of machines with equal-size disks; node A has 12 disks, node B has 12 disks, and node C has 11 disks. The ring has an overload factor of 0.1 (10%).

Without the overload, some partitions would end up with replicas only on nodes A and B. However, with the overload, every device can accept up to 10% more partitions for the sake of dispersion. The missing disk in C means there is one disk's worth of partitions to spread across the remaining 11 disks, which gives each disk in C an extra 9.09% load. Since this is less than the 10% overload, there is one replica of each partition on each node.

However, this does mean that the disks in node C have more data than the disks in nodes A and B. If 80% full is the warning threshold for the cluster, node C's disks reach 80% full while A and B's disks are only 72.7% full.

Replica counts

To support the gradual change in replica counts, a ring can have a real number of replicas and is not restricted to an integer number of replicas.

A fractional replica count is for the whole ring and not for individual partitions. It indicates the average number of replicas for each partition. For example, a replica count of 3.2 means that 20 percent of partitions have four replicas and 80 percent have three replicas.

The replica count is adjustable. For example:

```
$ swift-ring-builder account.builder set_replicas 4
$ swift-ring-builder account.builder rebalance
```

You must rebalance the replica ring in globally distributed clusters. Operators of these clusters generally want an equal number of replicas and regions. Therefore, when an operator adds or removes a region, the operator adds or removes a replica. Removing unneeded replicas saves on the cost of disks.

You can gradually increase the replica count at a rate that does not adversely affect cluster performance. For example:

```
$ swift-ring-builder object.builder set_replicas 3.01
$ swift-ring-builder object.builder rebalance
<distribute rings and wait>...

$ swift-ring-builder object.builder set_replicas 3.02
$ swift-ring-builder object.builder rebalance
<distribute rings and wait>...
```

Changes take effect after the ring is rebalanced. Therefore, if you intend to change from 3 replicas to 3.01 but you accidentally type 2.01, no data is lost.

Additionally, the **swift-ring-builder X.builder create** command can now take a decimal argument for the number of replicas.

Partition shift value

The partition shift value is known internally to the Ring class as `_part_shift`. This value is used to shift an MD5 hash to calculate the partition where the data for that hash should reside. Only the top four bytes of the hash is used in this process. For example, to compute the partition for the `/account/container/object` path using Python:

```
partition = unpack_from('>I',
md5('/account/container/object').digest())[0] >>
self._part_shift
```

For a ring generated with part_power P, the partition shift value is `32 - P`.

Build the ring

The ring builder process includes these high-level steps:

1. The utility calculates the number of partitions to assign to each device based on the weight of the device. For example, for a partition at the power of 20, the ring has 1,048,576 partitions. One thousand devices of equal weight each want 1,048.576 partitions. The devices are sorted by the number of partitions they desire and kept in order throughout the initialization process.

   > **Note**
   >
   > Each device is also assigned a random tiebreaker value that is used when two devices desire the same number of partitions. This tiebreaker is not stored on disk anywhere, and so two different rings created with the same parameters will have different partition assignments. For repeatable partition assignments, `RingBuilder.rebalance()` takes an optional seed value that seeds the Python pseudo-random number generator.

2. The ring builder assigns each partition replica to the device that requires most partitions at that point while keeping it as far away as possible from other replicas. The ring builder prefers to assign a replica to a device in a region that does not already have a replica. If no such region is available, the ring builder searches for a device in a different zone, or on a different server. If it does not find one, it looks for a device with no replicas. Finally, if all options are exhausted, the ring builder assigns the replica to the device that has the fewest replicas already assigned.

> **Note**
>
> The ring builder assigns multiple replicas to one device only if the ring has fewer devices than it has replicas.

3. When building a new ring from an old ring, the ring builder recalculates the desired number of partitions that each device wants.
4. The ring builder unassigns partitions and gathers these partitions for reassignment, as follows:
   - The ring builder unassigns any assigned partitions from any removed devices and adds these partitions to the gathered list.
   - The ring builder unassigns any partition replicas that can be spread out for better durability and adds these partitions to the gathered list.
   - The ring builder unassigns random partitions from any devices that have more partitions than they need and adds these partitions to the gathered list.
5. The ring builder reassigns the gathered partitions to devices by using a similar method to the one described previously.
6. When the ring builder reassigns a replica to a partition, the ring builder records the time of the reassignment. The ring builder uses this value when it gathers partitions for reassignment so that no partition is moved twice in a configurable amount of time. The RingBuilder class knows this configurable amount of time as `min_part_hours`. The ring builder ignores this restriction for replicas of partitions on removed devices because removal of a device happens on device failure only, and reassignment is the only choice.

These steps do not always perfectly rebalance a ring due to the random nature of gathering partitions for reassignment. To help reach a more balanced ring, the rebalance process is repeated until near perfect (less than 1 percent off) or when the balance does not improve by at least 1 percent (indicating we probably cannot get perfect balance due to wildly imbalanced zones or too many partitions recently moved).

Cluster architecture

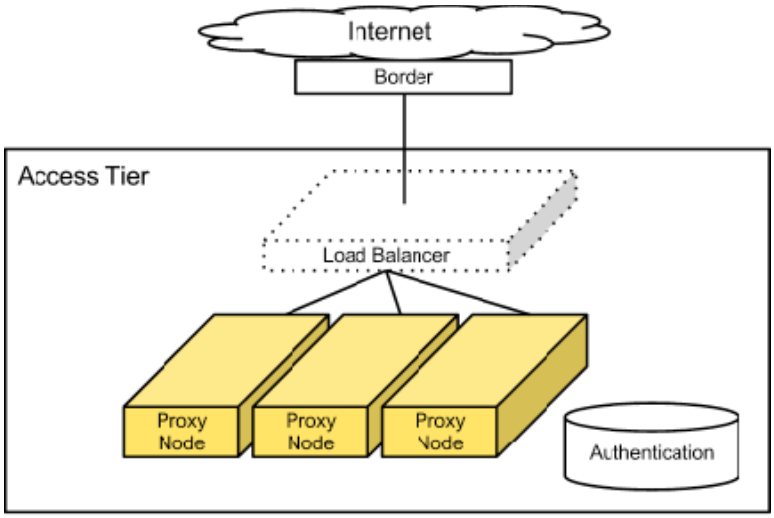
Access tier

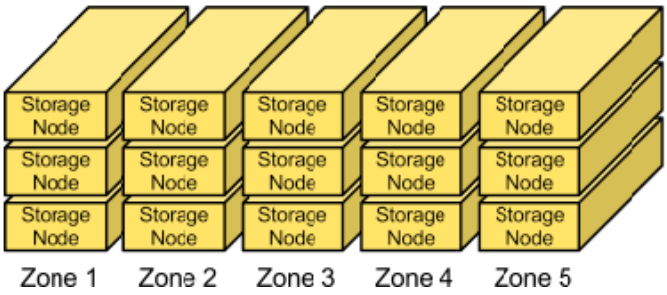
Large-scale deployments segment off an access tier, which is considered the Object Storage system’s central hub. The access tier fields the incoming API requests from clients and moves data in and out of the system. This tier consists of front-end load balancers, ssl-terminators, and authentication services. It runs the (distributed) brain of the Object

Storage system: the proxy server processes.

> **Note**
>
> If you want to use OpenStack Identity API v3 for authentication, you have the following options available in `/etc/swift/dispersion.conf`: `auth_version`, `user_domain_name`, `project_domain_name`, and `project_name`.

Object Storage architecture





Because access servers are collocated in their own tier, you can scale out read/write access regardless of the storage capacity. For example, if a cluster is on the public Internet, requires SSL termination, and has a high demand for data access, you can provision many access servers. However, if the cluster is on a private network and used primarily for archival purposes, you need fewer access servers.

Since this is an HTTP addressable storage service, you may incorporate a load balancer into the access tier.

Typically, the tier consists of a collection of 1U servers. These machines use a moderate amount of RAM and are network I/O intensive. Since these systems field each incoming API request, you should provision them with two high-throughput (10GbE) interfaces - one for the incoming `front-end` requests and the other for the `back-end` access to the object storage nodes to put and fetch data.

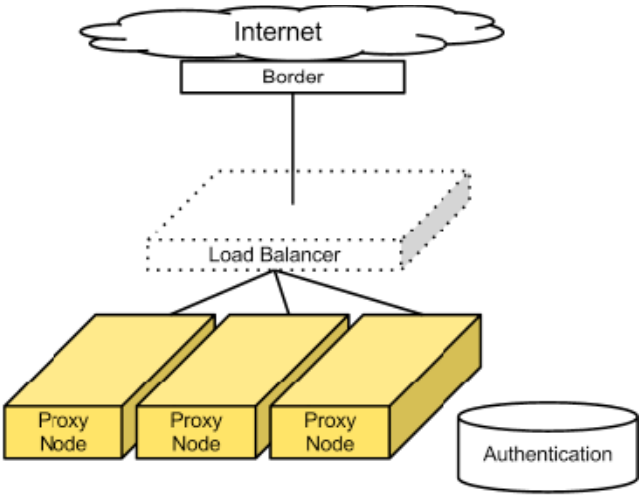
Factors to consider

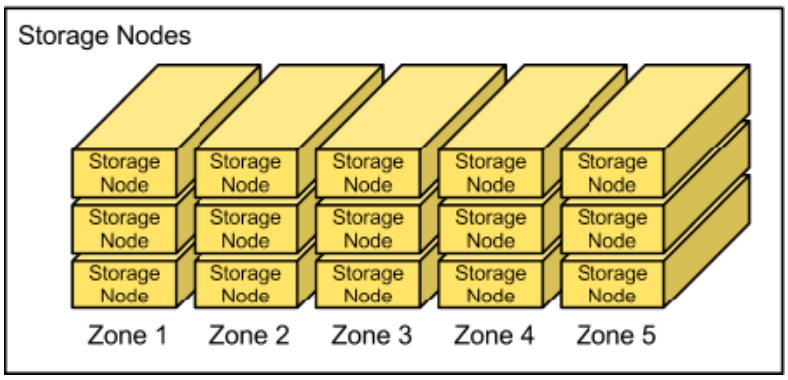
For most publicly facing deployments as well as private deployments available across a wide-reaching corporate network, you use SSL to encrypt traffic to the client. SSL adds significant processing load to establish sessions between clients, which is why you have to provision more capacity in the access layer. SSL may not be required for private deployments on trusted networks.

Storage nodes

In most configurations, each of the five zones should have an equal amount of storage capacity. Storage nodes use a reasonable amount of memory and CPU. Metadata needs to be readily available to return objects quickly. The object stores run services not only to field incoming requests from the access tier, but to also run replicators, auditors, and reapers. You can provision object stores provisioned with single gigabit or 10 gigabit network interface depending on the expected workload and desired performance.

Object Storage (swift)





Currently, a 2 TB or 3 TB SATA disk delivers good performance for the price. You can use desktop-grade drives if you have responsive remote hands in the datacenter and enterprise-grade drives if you don’t.

Factors to consider

You should keep in mind the desired I/O performance for single-threaded requests. This system does not use RAID, so a single disk handles each request for an object. Disk performance impacts single-threaded response rates.

To achieve apparent higher throughput, the object storage system is designed to handle concurrent uploads/downloads. The network I/O capacity (1GbE, bonded 1GbE pair, or 10GbE) should match your desired concurrent throughput needs for reads and writes.

Replication

Because each replica in Object Storage functions independently and clients generally require only a simple majority of nodes to respond to consider an operation successful, transient failures like network partitions can quickly cause replicas to diverge. These differences are eventually reconciled by asynchronous, peer-to-peer replicator processes. The replicator processes traverse their local file systems and concurrently perform operations in a manner that balances load across physical disks.

Replication uses a push model, with records and files generally only being copied from local to remote replicas. This is important because data on the node might not belong there (as in the case of hand offs and ring changes), and a replicator cannot know which data it should pull in from elsewhere in the cluster. Any node that contains data must ensure that data gets to where it belongs. The ring handles replica placement.

To replicate deletions in addition to creations, every deleted record or file in the system is marked by a tombstone. The replication process cleans up tombstones after a time period known as the `consistency window`. This window defines the duration of the replication and how long transient failure can remove a node from the cluster. Tombstone cleanup must be tied to replication to reach replica convergence.

If a replicator detects that a remote drive has failed, the replicator uses the `get_more_nodes` interface for the ring to choose an alternate node with which to synchronize. The replicator can maintain desired levels of replication during disk failures, though some replicas might not be in an immediately usable location.

> **Note**
>
> The replicator does not maintain desired levels of replication when failures such as entire node failures occur; most failures are transient.

The main replication types are:

- **Database replication**

  Replicates containers and objects.

- **Object replication**

  Replicates object data.

Database replication

Database replication completes a low-cost hash comparison to determine whether two replicas already match. Normally, this check can quickly verify that most databases in the system are already synchronized. If the hashes differ, the replicator synchronizes the databases by sharing records added since the last synchronization point.

This synchronization point is a high water mark that notes the last record at which two databases were known to be synchronized, and is stored in each database as a tuple of the remote database ID and record ID. Database IDs are unique across all replicas of the database, and record IDs are monotonically increasing integers. After all new records are pushed to the remote database, the entire synchronization table of the local database is pushed, so the remote database can guarantee that it is synchronized with everything with which the local database was previously synchronized.

If a replica is missing, the whole local database file is transmitted to the peer by using rsync(1) and is assigned a new unique ID.

In practice, database replication can process hundreds of databases per concurrency setting per second (up to the number of available CPUs or disks) and is bound by the number of database transactions that must be performed.

Object replication

The initial implementation of object replication performed an rsync to push data from a local partition to all remote servers where it was expected to reside. While this worked at small scale, replication times skyrocketed once directory structures could no longer be held in RAM. This scheme was modified to save a hash of the contents for each suffix directory to a per-partition hashes file. The hash for a suffix directory is no longer valid when the contents of that suffix directory is modified.

The object replication process reads in hash files and calculates any invalidated hashes. Then, it transmits the hashes to each remote server that should hold the partition, and only suffix directories with differing hashes on the remote server are rsynced. After pushing files to the remote server, the replication process notifies it to recalculate hashes

for the rsynced suffix directories.

The number of uncached directories that object replication must traverse, usually as a result of invalidated suffix directory hashes, impedes performance. To provide acceptable replication speeds, object replication is designed to invalidate around 2 percent of the hash space on a normal node each day.

Large object support

Object Storage (swift) uses segmentation to support the upload of large objects. By default, Object Storage limits the download size of a single object to 5GB. Using segmentation, uploading a single object is virtually unlimited. The segmentation process works by fragmenting the object, and automatically creating a file that sends the segments together as a single object. This option offers greater upload speed with the possibility of parallel uploads.

Large objects

The large object is comprised of two types of objects:

- **Segment objects** store the object content. You can divide your content into segments, and upload each segment into its own segment object. Segment objects do not have any special features. You create, update, download, and delete segment objects just as you would normal objects.
- A **manifest object** links the segment objects into one logical large object. When you download a manifest object, Object Storage concatenates and returns the contents of the segment objects in the response body of the request. The manifest object types are:
    - **Static large objects**
    - **Dynamic large objects**

To find out more information on large object support, see Large objects in the OpenStack End User Guide, or Large Object Support in the developer documentation.

Object Auditor

On system failures, the XFS file system can sometimes truncate files it is trying to write and produce zero-byte files. The object-auditor will catch these problems but in the case of a system crash it is advisable to run an extra, less rate limited sweep, to check for these specific files. You can run this command as follows:

```
$ swift-object-auditor /path/to/object-server/config/file.conf once -z 1000
```

> **Note**
>
> "-z" means to only check for zero-byte files at 1000 files per second.

It is useful to run the object auditor on a specific device or set of devices. You can run the object-auditor once as follows:

```
$ swift-object-auditor /path/to/object-server/config/file.conf once \
  --devices=sda,sdb
```

> **Note**
>
> This will run the object auditor on only the sda and sdb devices. This parameter accepts a comma-separated list of values.

Erasure coding

Erasure coding is a set of algorithms that allows the reconstruction of missing data from a set of original data. In theory, erasure coding uses less capacity with similar durability characteristics as replicas. From an application perspective, erasure coding support is transparent. Object Storage (swift) implements erasure coding as a Storage Policy. See Storage Policies for more details.

There is no external API related to erasure coding. Create a container using a Storage Policy; the interaction with the cluster is the same as any other durability policy. Because support implements as a Storage Policy, you can isolate all storage devices that associate with your cluster's erasure coding capability. It is entirely possible to share devices between storage policies, but for erasure coding it may make more sense to use not only separate devices but possibly even entire nodes dedicated for erasure coding.

> **Important**
>
> The erasure code support in Object Storage is considered beta in Kilo. Most major functionality is included, but it has not been tested or validated at large scale. This feature relies on ssync for durability. We recommend deployers do extensive testing and not deploy production data using an erasure code storage policy. If any bugs are found during testing, please report them to https://bugs.launchpad.net/swift

Account reaper

The purpose of the account reaper is to remove data from the deleted accounts.

A reseller marks an account for deletion by issuing a `DELETE` request on the account's storage URL. This action sets the `status` column of the account_stat table in the account database and replicas to `DELETED`, marking the account's data for deletion.

Typically, a specific retention time or undelete are not provided. However, you can set a `delay_reaping` value in the `[account-reaper]` section of the `account-server.conf` file to delay the actual deletion of data. At this time, to undelete you have to update the account database replicas directly, set the status column to an empty string and update the put_timestamp to be greater than the delete_timestamp.

> **Note**
>
> It is on the development to-do list to write a utility that performs this task, preferably through a REST call.

The account reaper runs on each account server and scans the server occasionally for account databases marked for deletion. It only fires up on the accounts for which the server is the primary node, so that multiple account servers aren't trying to do it simultaneously. Using multiple servers to delete one account might improve the deletion speed but requires coordination to avoid duplication. Speed really is not a big concern with data deletion, and large accounts aren't deleted often.

Deleting an account is simple. For each account container, all objects are deleted and then the container is deleted. Deletion requests that fail will not stop the overall process but will cause the overall process to fail eventually (for example, if an object delete times out, you will not be able to delete the container or the account). The account reaper keeps trying to delete an account until it is empty, at which point the database reclaim process within the db_replicator will remove the database files.

A persistent error state may prevent the deletion of an object or container. If this happens, you will see a message in the log, for example:

```
Account <name> has not been reaped since <date>
```

You can control when this is logged with the `reap_warn_after` value in the `[account-reaper]` section of the `account-server.conf` file. The default value is 30 days.

Configure tenant-specific image locations with Object Storage

For some deployers, it is not ideal to store all images in one place to enable all tenants and users to access them. You can configure the Image service to store image data in tenant-specific image locations. Then, only the following tenants can use the Image service to access the created image:

- The tenant who owns the image
- Tenants that are defined in `swift_store_admin_tenants` and that have admin-level accounts

To configure tenant-specific image locations

1. Configure swift as your `default_store` in the `glance-api.conf` file.
2. Set these configuration options in the `glance-api.conf` file:
    - **swift_store_multi_tenant**

      Set to `True` to enable tenant-specific storage locations. Default is `False`.
    - **swift_store_admin_tenants**

      Specify a list of tenant IDs that can grant read and write access to all Object Storage containers that are created by the Image service.

With this configuration, images are stored in an Object Storage service (swift) endpoint that is pulled from the service catalog for the authenticated user.

Object Storage monitoring

> **Note**
>
> This section was excerpted from a blog post by Darrell Bishop and has since been edited.

An OpenStack Object Storage cluster is a collection of many daemons that work together across many nodes. With so many different components, you must be able to tell what is going on inside the cluster. Tracking server-level meters like CPU utilization, load, memory consumption, disk usage and utilization, and so on is necessary, but not sufficient.

Swift Recon

The Swift Recon middleware (see Defining Storage Policies) provides general machine statistics, such as load average, socket statistics, `/proc/meminfo` contents, as well as Swift-specific meters:

- The `MD5` sum of each ring file.
- The most recent object replication time.
- Count of each type of quarantined file: Account, container, or object.
- Count of “async_pendings” (deferred container updates) on disk.

Swift Recon is middleware that is installed in the object servers pipeline and takes one required option: A local cache directory. To track `async_pendings`, you must set up an additional cron job for each object server. You access data by either sending HTTP requests directly to the object server or using the `swift-recon` command-line client.

There are Object Storage cluster statistics but the typical server meters overlap with existing server monitoring systems. To get the Swift-specific meters into a monitoring system, they must be polled. Swift Recon acts as a middleware meters collector. The process that feeds meters to your statistics system, such as `collectd` and `gmond`, should already run on the storage node. You can choose to either talk to Swift Recon or collect the meters directly.

Swift-Informant

Swift-Informant middleware (see swift-informant) has real-time visibility into Object Storage client requests. It sits in the pipeline for the proxy server, and after each request to the proxy server it sends three meters to a `StatsD` server:

- A counter increment for a meter like `obj.GET.200` or `cont.PUT.404`.
- Timing data for a meter like `acct.GET.200` or `obj.GET.200`. [The README says the meters look like `duration.acct.GET.200`, but I do not see the `duration` in the code. I am not sure what the Etsy server does but our StatsD server turns timing meters into five derivative meters with new segments appended, so it probably works as coded. The first meter turns into `acct.GET.200.lower`, `acct.GET.200.upper`, `acct.GET.200.mean`, `acct.GET.200.upper_90`, and `acct.GET.200.count`].
- A counter increase by the bytes transferred for a meter like `tfer.obj.PUT.201`.

This is used for receiving information on the quality of service clients experience with the timing meters, as well as sensing the volume of the various modifications of a request server type, command, and response code. Swift-Informant requires no change to core Object Storage code because it is implemented as middleware. However, it gives no insight into the workings of the cluster past the proxy server. If the responsiveness of one

storage node degrades, you can only see that some of the requests are bad, either as high latency or error status codes.

Statsdlog

The Statsdlog project increments StatsD counters based on logged events. Like Swift-Informant, it is also non-intrusive, however statsdlog can track events from all Object Storage daemons, not just proxy-server. The daemon listens to a UDP stream of syslog messages, and StatsD counters are incremented when a log line matches a regular expression. Meter names are mapped to regex match patterns in a JSON file, allowing flexible configuration of what meters are extracted from the log stream.

Currently, only the first matching regex triggers a StatsD counter increment, and the counter is always incremented by one. There is no way to increment a counter by more than one or send timing data to StatsD based on the log line content. The tool could be extended to handle more meters for each line and data extraction, including timing data. But a coupling would still exist between the log textual format and the log parsing regexes, which would themselves be more complex to support multiple matches for each line and data extraction. Also, log processing introduces a delay between the triggering event and sending the data to StatsD. It would be preferable to increment error counters where they occur and send timing data as soon as it is known to avoid coupling between a log string and a parsing regex and prevent a time delay between events and sending data to StatsD.

The next section describes another method for gathering Object Storage operational meters.

Swift StatsD logging

StatsD (see http://codeascraft.etsy.com/2011/02/15/measure-anything-measure-everything/) was designed for application code to be deeply instrumented. Meters are sent in real-time by the code that just noticed or did something. The overhead of sending a meter is extremely low: a `sendto` of one UDP packet. If that overhead is still too high, the StatsD client library can send only a random portion of samples and StatsD approximates the actual number when flushing meters upstream.

To avoid the problems inherent with middleware-based monitoring and after-the-fact log processing, the sending of StatsD meters is integrated into Object Storage itself. The submitted change set (see https://review.openstack.org/#change,6058) currently reports 124 meters across 15 Object Storage daemons and the tempauth middleware. Details of the meters tracked are in the Administrator's Guide.

The sending of meters is integrated with the logging framework. To enable, configure `log_statsd_host` in the relevant config file. You can also specify the port and a default sample rate. The specified default sample rate is used unless a specific call to a statsd

logging method (see the list below) overrides it. Currently, no logging calls override the sample rate, but it is conceivable that some meters may require accuracy (`sample_rate=1`) while others may not.

```
[DEFAULT]
     ...
log_statsd_host = 127.0.0.1
log_statsd_port = 8125
log_statsd_default_sample_rate = 1
```

Then the LogAdapter object returned by `get_logger()`, usually stored in `self.logger`, has these new methods:

- `set_statsd_prefix(self, prefix)` Sets the client library stat prefix value which gets prefixed to every meter. The default prefix is the `name` of the logger such as `object-server`, `container-auditor`, and so on. This is currently used to turn `proxy-server` into one of `proxy-server.Account`, `proxy-server.Container`, or `proxy-server.Object` as soon as the Controller object is determined and instantiated for the request.
- `update_stats(self, metric, amount, sample_rate=1)` Increments the supplied meter by the given amount. This is used when you need to add or subtract more that one from a counter, like incrementing `suffix.hashes` by the number of computed hashes in the object replicator.
- `increment(self, metric, sample_rate=1)` Increments the given counter meter by one.
- `decrement(self, metric, sample_rate=1)` Lowers the given counter meter by one.
- `timing(self, metric, timing_ms, sample_rate=1)` Record that the given meter took the supplied number of milliseconds.
- `timing_since(self, metric, orig_time, sample_rate=1)` Convenience method to record a timing meter whose value is “now” minus an existing timestamp.

> **Note**
>
> These logging methods may safely be called anywhere you have a logger object. If StatsD logging has not been configured, the methods are no-ops. This avoids messy conditional logic each place a meter is recorded. These example usages show the new logging methods:

```
    # swift/obj/replicator.py def update(self, job): # ... begin = time.time() try: hashed,
  local_hash = tpool.execute(tpooled_get_hashes, job['path'],
  do_listdir=(self.replication_count % 10) == 0, reclaim_age=self.reclaim_age) # See
  tpooled_get_hashes "Hack". if isinstance(hashed, BaseException): raise hashed
  self.suffix_hash += hashed self.logger.update_stats('suffix.hashes', hashed) # ...
  finally: self.partition_times.append(time.time() - begin)
  self.logger.timing_since('partition.update.timing', begin)
```

```
  # swift/container/updater.py
def process_container(self, dbfile):
    # ...
    start_time = time.time()
    # ...
        for event in events:
            if 200 <= event.wait() < 300:
                successes += 1
            else:
                failures += 1
        if successes > failures:
          self.logger.increment('successes')
            # ...
        else:
            self.logger.increment('failures')
            # ...
        # Only track timing data for attempted updates:
        self.logger.timing_since('timing', start_time)
    else:
        self.logger.increment('no_changes')
        self.no_changes += 1
```

System administration for Object Storage

By understanding Object Storage concepts, you can better monitor and administer your storage solution. The majority of the administration information is maintained in developer documentation at docs.openstack.org/developer/swift/.

See the OpenStack Configuration Reference for a list of configuration options for Object Storage.

Troubleshoot Object Storage

For Object Storage, everything is logged in `/var/log/syslog` (or `messages` on some distros). Several settings enable further customization of logging, such as `log_name`, `log_facility`, and `log_level`, within the object server configuration files.

Drive failure

Problem

Drive failure can prevent Object Storage performing replication.

Solution

In the event that a drive has failed, the first step is to make sure the drive is unmounted. This will make it easier for Object Storage to work around the failure until it has been resolved. If the drive is going to be replaced immediately, then it is just best to replace the drive, format it, remount it, and let replication fill it up.

If you cannot replace the drive immediately, then it is best to leave it unmounted, and remove the drive from the ring. This will allow all the replicas that were on that drive to be replicated elsewhere until the drive is replaced. Once the drive is replaced, it can be re-added to the ring.

You can look at error messages in the `/var/log/kern.log` file for hints of drive failure.

Server failure

Problem

The server is potentially offline, and may have failed, or require a reboot.

Solution

If a server is having hardware issues, it is a good idea to make sure the Object Storage services are not running. This will allow Object Storage to work around the failure while you troubleshoot.

If the server just needs a reboot, or a small amount of work that should only last a couple of hours, then it is probably best to let Object Storage work around the failure and get the machine fixed and back online. When the machine comes back online, replication will make sure that anything that is missing during the downtime will get updated.

If the server has more serious issues, then it is probably best to remove all of the server’s

devices from the ring. Once the server has been repaired and is back online, the server's devices can be added back into the ring. It is important that the devices are reformatted before putting them back into the ring as it is likely to be responsible for a different set of partitions than before.

Detect failed drives

Problem

When drives fail, it can be difficult to detect that a drive has failed, and the details of the failure.

Solution

It has been our experience that when a drive is about to fail, error messages appear in the `/var/log/kern.log` file. There is a script called `swift-drive-audit` that can be run via cron to watch for bad drives. If errors are detected, it will unmount the bad drive, so that Object Storage can work around it. The script takes a configuration file with the following settings:

Description of configuration options for [drive-audit] in drive-audit.conf

Configuration option = Default value	Description
`device_dir = /srv/node`	Directory devices are mounted under
`error_limit = 1`	Number of errors to find before a device is unmounted
`log_address = /dev/log`	Location where syslog sends the logs to
`log_facility = LOG_LOCAL0`	Syslog log facility
`log_file_pattern = /var/log/kern.*[!.][!g][!z]`	Location of the log file with globbing pattern to check against device errors locate device blocks with errors in the log file
`log_level = INFO`	Logging level
`log_max_line_length = 0`	Caps the length of log lines to the value given; no limit if set to 0, the default.
`log_to_console = False`	No help text available for this option.

Configuration option = Default value	Description
minutes = 60	Number of minutes to look back in /var/log/kern.log
recon_cache_path = /var/cache/swift	Directory where stats for a few items will be stored
regex_pattern_1 = \berror\b.*\b(dm-[0-9]{1,2}\d?)\b	No help text available for this option.
unmount_failed_device = True	No help text available for this option.

> **Warning**
>
> This script has only been tested on Ubuntu 10.04; use with caution on other operating systems in production.

Emergency recovery of ring builder files

Problem

An emergency might prevent a successful backup from restoring the cluster to operational status.

Solution

You should always keep a backup of swift ring builder files. However, if an emergency occurs, this procedure may assist in returning your cluster to an operational state.

Using existing swift tools, there is no way to recover a builder file from a `ring.gz` file. However, if you have a knowledge of Python, it is possible to construct a builder file that is pretty close to the one you have lost.

> **Warning**
>
> This procedure is a last-resort for emergency circumstances. It requires knowledge of the swift python code and may not succeed.

1. Load the ring and a new ringbuilder object in a Python REPL:

```
>>> from swift.common.ring import RingData, RingBuilder
>>> ring = RingData.load('/path/to/account.ring.gz')
```

2. Start copying the data we have in the ring into the builder:

```
>>> import math
>>> partitions = len(ring._replica2part2dev_id[0])
>>> replicas = len(ring._replica2part2dev_id)

>>> builder = RingBuilder(int(math.log(partitions, 2)), replicas, 1)
>>> builder.devs = ring.devs
>>> builder._replica2part2dev = ring._replica2part2dev_id
>>> builder._last_part_moves_epoch = 0
>>> from array import array
>>> builder._last_part_moves = array('B', (0 for _ in xrange(partitions)))
>>> builder._set_parts_wanted()
>>> for d in builder._iter_devs():
            d['parts'] = 0
>>> for p2d in builder._replica2part2dev:
            for dev_id in p2d:
                builder.devs[dev_id]['parts'] += 1

This is the extent of the recoverable fields.
```

3. For `min_part_hours` you either have to remember what the value you used was, or just make up a new one:

```
>>> builder.change_min_part_hours(24) # or whatever you want it to be
```

4. Validate the builder. If this raises an exception, check your previous code:

```
>>> builder.validate()
```

5. After it validates, save the builder and create a new `account.builder`:

```
>>> import pickle
>>> pickle.dump(builder.to_dict(), open('account.builder', 'wb'),
protocol=2)
>>> exit ()
```

6. You should now have a file called `account.builder` in the current working directory. Run **swift-ring-builder account.builder write_ring** and compare the new `account.ring.gz` to the `account.ring.gz` that you started from. They probably are not byte-for-byte identical, but if you load them in a REPL and their `_replica2part2dev_id` and `devs` attributes are the same (or nearly so), then you are in good shape.

7. Repeat the procedure for `container.ring.gz` and `object.ring.gz`, and you might get usable builder files.

Block Storage

The OpenStack Block Storage service works through the interaction of a series of daemon processes named `cinder-*` that reside persistently on the host machine or machines. You can run all the binaries from a single node, or spread across multiple nodes. You can also run them on the same node as other OpenStack services.

To administer the OpenStack Block Storage service, it is helpful to understand a number of concepts. You must make certain choices when you configure the Block Storage service in OpenStack. The bulk of the options come down to two choices - single node or multi-node install. You can read a longer discussion about Storage Decisions in the OpenStack Operations Guide.

OpenStack Block Storage enables you to add extra block-level storage to your OpenStack Compute instances. This service is similar to the Amazon EC2 Elastic Block Storage (EBS) offering.

- Increase Block Storage API service throughput
- Manage volumes
- Troubleshoot your installation

Increase Block Storage API service throughput

By default, the Block Storage API service runs in one process. This limits the number of API requests that the Block Storage service can process at any given time. In a production environment, you should increase the Block Storage API throughput by allowing the Block Storage API service to run in as many processes as the machine capacity allows.

> **Note**
>
> The Block Storage API service is named `openstack-cinder-api` on the following distributions: CentOS, Fedora, openSUSE, Red Hat Enterprise Linux, and SUSE Linux Enterprise. In Ubuntu and Debian distributions, the Block Storage API service is named `cinder-api`.

To do so, use the Block Storage API service option `osapi_volume_workers`. This option allows you to specify the number of API service workers (or OS processes) to launch for the Block Storage API service.

To configure this option, open the `/etc/cinder/cinder.conf` configuration file and set the `osapi_volume_workers` configuration key to the number of CPU cores/threads on a machine.

On distributions that include `openstack-config`, you can configure this by running the following command instead:

```
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT osapi_volume_workers CORES
```

Replace `CORES` with the number of CPU cores/threads on a machine.

Manage volumes

The default OpenStack Block Storage service implementation is an iSCSI solution that uses *Logical Volume Manager (LVM)* for Linux.

> **Note**
>
> The OpenStack Block Storage service is not a shared storage solution like a Network Attached Storage (NAS) of NFS volumes where you can attach a volume to multiple servers. With the OpenStack Block Storage service, you can attach a volume to only one instance at a time.
>
> The OpenStack Block Storage service also provides drivers that enable you to use several vendors' back-end storage devices in addition to the base LVM implementation. These storage devices can also be used instead of the base LVM installation.

This high-level procedure shows you how to create and attach a volume to a server instance.

To create and attach a volume to an instance

1. Configure the OpenStack Compute and the OpenStack Block Storage services through the `/etc/cinder/cinder.conf` file.
2. Use the **openstack volume create ` command to create a volume. This command creates an LV into the volume group (VG) ``cinder-volumes`**.
3. Use the **openstack server add volume** command to attach the volume to an instance. This command creates a unique *IQN* that is exposed to the compute node.
    - The compute node, which runs the instance, now has an active iSCSI session and new local storage (usually a `/dev/sdX` disk).
    - Libvirt uses that local storage as storage for the instance. The instance gets a new disk (usually a `/dev/vdX` disk).

For this particular walkthrough, one cloud controller runs `nova-api`, `nova-scheduler`, `nova-objectstore`, `nova-network` and `cinder-*` services. Two additional compute nodes run `nova-compute`. The walkthrough uses a custom partitioning scheme that carves out 60 GB of space and labels it as LVM. The network uses the `FlatManager` and `NetworkManager` settings for OpenStack Compute.

The network mode does not interfere with OpenStack Block Storage operations, but you must set up networking for Block Storage to work. For details, see *Networking*.

To set up Compute to use volumes, ensure that Block Storage is installed along with `lvm2`. This guide describes how to troubleshoot your installation and back up your Compute volumes.

Boot from volume

In some cases, you can store and run instances from inside volumes. For information, see the Launch an instance from a volume section in the OpenStack End User Guide.

Configure an NFS storage back end

This section explains how to configure OpenStack Block Storage to use NFS storage. You must be able to access the NFS shares from the server that hosts the `cinder` volume service.

> **Note**
>
> The `cinder` volume service is named `openstack-cinder-volume` on the following distributions:
>
> - CentOS
> - Fedora
> - openSUSE
> - Red Hat Enterprise Linux
> - SUSE Linux Enterprise
>
> In Ubuntu and Debian distributions, the `cinder` volume service is named `cinder-volume`.

Configure Block Storage to use an NFS storage back end

1. Log in as `root` to the system hosting the `cinder` volume service.
2. Create a text file named `nfsshares` in the `/etc/cinder/` directory.
3. Add an entry to `/etc/cinder/nfsshares` for each NFS share that the `cinder` volume service should use for back end storage. Each entry should be a separate line, and should use the following format:

   ```
   HOST:SHARE
   ```

   Where:

   - HOST is the IP address or host name of the NFS server.
   - SHARE is the absolute path to an existing and accessible NFS share.

4. Set `/etc/cinder/nfsshares` to be owned by the `root` user and the `cinder` group:

   ```
   # chown root:cinder /etc/cinder/nfsshares
   ```

5. Set `/etc/cinder/nfsshares` to be readable by members of the cinder group:

   ```
   # chmod 0640 /etc/cinder/nfsshares
   ```

6. Configure the `cinder` volume service to use the `/etc/cinder/nfsshares` file created earlier. To do so, open the `/etc/cinder/cinder.conf` configuration file and set the `nfs_shares_config` configuration key to `/etc/cinder/nfsshares`.

   On distributions that include `openstack-config`, you can configure this by running the following command instead:

   ```
   # openstack-config --set /etc/cinder/cinder.conf \
     DEFAULT nfs_shares_config /etc/cinder/nfsshares
   ```

   The following distributions include openstack-config:

   - CentOS
   - Fedora
   - openSUSE
   - Red Hat Enterprise Linux
   - SUSE Linux Enterprise

7. Optionally, provide any additional NFS mount options required in your environment in the `nfs_mount_options` configuration key of `/etc/cinder/cinder.conf`. If your NFS shares do not require any additional mount options (or if you are unsure), skip this step.

   On distributions that include `openstack-config`, you can configure this by running the following command instead:

   ```
   # openstack-config --set /etc/cinder/cinder.conf \
     DEFAULT nfs_mount_options OPTIONS
   ```

   Replace OPTIONS with the mount options to be used when accessing NFS shares. See the manual page for NFS for more information on available mount options (**man nfs**).

8. Configure the `cinder` volume service to use the correct volume driver, namely `cinder.volume.drivers.nfs.NfsDriver`. To do so, open the `/etc/cinder/cinder.conf` configuration file and set the volume_driver configuration key to `cinder.volume.drivers.nfs.NfsDriver`.

   On distributions that include `openstack-config`, you can configure this by running the following command instead:

   ```
   # openstack-config --set /etc/cinder/cinder.conf \
     DEFAULT volume_driver cinder.volume.drivers.nfs.NfsDriver
   ```

9. You can now restart the service to apply the configuration.

> **Note**
>
> The `nfs_sparsed_volumes` configuration key determines whether volumes are created as sparse files and grown as needed or fully allocated up front. The default and recommended value is `true`, which ensures volumes are initially created as sparse files.
>
> Setting `nfs_sparsed_volumes` to `false` will result in volumes being fully allocated at the time of creation. This leads to increased delays in volume creation.
>
> However, should you choose to set `nfs_sparsed_volumes` to `false`, you can do so directly in `/etc/cinder/cinder.conf`.
>
> On distributions that include `openstack-config`, you can configure this by running the following command instead:
>
> ```
> # openstack-config --set /etc/cinder/cinder.conf \ DEFAULT nfs_sparsed_volumes
> false
> ```

> **Warning**
>
> If a client host has SELinux enabled, the `virt_use_nfs` boolean should also be enabled if the host requires access to NFS volumes on an instance. To enable this boolean, run the following command as the `root` user:
>
> ```
> # setsebool -P virt_use_nfs on
> ```

This command also makes the boolean persistent across reboots. Run this command on all client hosts that require access to NFS volumes on an instance. This includes all compute nodes.

Configure a GlusterFS back end

This section explains how to configure OpenStack Block Storage to use GlusterFS as a back end. You must be able to access the GlusterFS shares from the server that hosts the `cinder` volume service.

> **Note**
>
> The cinder volume service is named `openstack-cinder-volume` on the following distributions:
>
> - CentOS
> - Fedora
> - openSUSE
> - Red Hat Enterprise Linux
> - SUSE Linux Enterprise
>
> In Ubuntu and Debian distributions, the `cinder` volume service is named `cinder-volume`.

Mounting GlusterFS volumes requires utilities and libraries from the `glusterfs-fuse` package. This package must be installed on all systems that will access volumes backed by GlusterFS.

> **Note**
>
> The utilities and libraries required for mounting GlusterFS volumes on Ubuntu and Debian distributions are available from the `glusterfs-client` package instead.

For information on how to install and configure GlusterFS, refer to the GlusterDocumentation page.

Configure GlusterFS for OpenStack Block Storage

The GlusterFS server must also be configured accordingly in order to allow OpenStack Block Storage to use GlusterFS shares:

1. Log in as `root` to the GlusterFS server.
2. Set each Gluster volume to use the same UID and GID as the `cinder` user:

```
# gluster volume set VOL_NAME storage.owner-uid CINDER_UID
# gluster volume set VOL_NAME storage.owner-gid CINDER_GID
```

Where:

- VOL_NAME is the Gluster volume name.
- CINDER_UID is the UID of the `cinder` user.
- CINDER_GID is the GID of the `cinder` user.

> **Note**
>
> The default UID and GID of the `cinder` user is 165 on most distributions.

3. Configure each Gluster volume to accept `libgfapi` connections. To do this, set each Gluster volume to allow insecure ports:

```
# gluster volume set VOL_NAME server.allow-insecure on
```

4. Enable client connections from unprivileged ports. To do this, add the following line to `/etc/glusterfs/glusterd.vol`:

```
option rpc-auth-allow-insecure on
```

5. Restart the `glusterd` service:

```
# service glusterd restart
```

Configure Block Storage to use a GlusterFS back end

After you configure the GlusterFS service, complete these steps:

1. Log in as `root` to the system hosting the Block Storage service.
2. Create a text file named `glusterfs` in `/etc/cinder/` directory.
3. Add an entry to `/etc/cinder/glusterfs` for each GlusterFS share that OpenStack Block Storage should use for back end storage. Each entry should be a separate line, and should use the following format:

```
HOST:/VOL_NAME
```

Where:

- HOST is the IP address or host name of the Red Hat Storage server.
- VOL_NAME is the name of an existing and accessible volume on the GlusterFS server.

Optionally, if your environment requires additional mount options for a share, you can add them to the share's entry:

```
HOST:/VOL_NAME -o OPTIONS
```

Replace OPTIONS with a comma-separated list of mount options.

4. Set `/etc/cinder/glusterfs` to be owned by the root user and the `cinder` group:

```
# chown root:cinder /etc/cinder/glusterfs
```

5. Set `/etc/cinder/glusterfs` to be readable by members of the `cinder` group:

```
# chmod 0640 /etc/cinder/glusterfs
```

6. Configure OpenStack Block Storage to use the `/etc/cinder/glusterfs` file created earlier. To do so, open the `/etc/cinder/cinder.conf` configuration file and set the `glusterfs_shares_config` configuration key to `/etc/cinder/glusterfs`.

   On distributions that include openstack-config, you can configure this by running the following command instead:

```
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT glusterfs_shares_config /etc/cinder/glusterfs
```

   The following distributions include `openstack-config`:

   - CentOS
   - Fedora
   - openSUSE
   - Red Hat Enterprise Linux
   - SUSE Linux Enterprise

7. Configure OpenStack Block Storage to use the correct volume driver, namely `cinder.volume.drivers.glusterfs.GlusterfsDriver`. To do so, open the `/etc/cinder/cinder.conf` configuration file and set the `volume_driver` configuration key to `cinder.volume.drivers.glusterfs.GlusterfsDriver`.

   On distributions that include `openstack-config`, you can configure this by running the following command instead:

```
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT volume_driver cinder.volume.drivers.glusterfs.GlusterfsDriver
```

8. You can now restart the service to apply the configuration.

OpenStack Block Storage is now configured to use a GlusterFS back end.

> **Warning**
>
> If a client host has SELinux enabled, the `virt_use_fusefs` boolean should also be enabled if the host requires access to GlusterFS volumes on an instance. To enable this

Boolean, run the following command as the `root` user:

```
# setsebool -P virt_use_fusefs on
```

This command also makes the Boolean persistent across reboots. Run this command on all client hosts that require access to GlusterFS volumes on an instance. This includes all compute nodes.

Configure multiple-storage back ends

When you configure multiple-storage back ends, you can create several back-end storage solutions that serve the same OpenStack Compute configuration and one `cinder-volume` is launched for each back-end storage or back-end storage pool.

In a multiple-storage back-end configuration, each back end has a name (`volume_backend_name`). Several back ends can have the same name. In that case, the scheduler properly decides which back end the volume has to be created in.

The name of the back end is declared as an extra-specification of a volume type (such as, `volume_backend_name=LVM`). When a volume is created, the scheduler chooses an appropriate back end to handle the request, according to the volume type specified by the user.

Enable multiple-storage back ends

To enable a multiple-storage back ends, you must set the enabled_backends flag in the `cinder.conf` file. This flag defines the names (separated by a comma) of the configuration groups for the different back ends: one name is associated to one configuration group for a back end (such as, `[lvmdriver-1]`).

Note

The configuration group name is not related to the `volume_backend_name`.

Note

After setting the `enabled_backends` flag on an existing cinder service, and restarting the Block Storage services, the original `host` service is replaced with a new host service. The new service appears with a name like `host@backend`. Use:

```
$ cinder-manage volume update_host --currenthost CURRENTHOST --newhost
CURRENTHOST@BACKEND
```

to convert current block devices to the new host name.

The options for a configuration group must be defined in the group (or default options are used). All the standard Block Storage configuration options (`volume_group`, `volume_driver`, and so on) might be used in a configuration group. Configuration values in the `[DEFAULT]` configuration group are not used.

These examples show three back ends:

```
enabled_backends=lvmdriver-1,lvmdriver-2,lvmdriver-3
[lvmdriver-1]
volume_group=cinder-volumes-1
volume_driver=cinder.volume.drivers.lvm.LVMVolumeDriver
volume_backend_name=LVM
[lvmdriver-2]
volume_group=cinder-volumes-2
volume_driver=cinder.volume.drivers.lvm.LVMVolumeDriver
volume_backend_name=LVM
[lvmdriver-3]
volume_group=cinder-volumes-3
volume_driver=cinder.volume.drivers.lvm.LVMVolumeDriver
volume_backend_name=LVM_b
```

In this configuration, `lvmdriver-1` and `lvmdriver-2` have the same `volume_backend_name`. If a volume creation requests the `LVM` back end name, the scheduler uses the capacity filter scheduler to choose the most suitable driver, which is either `lvmdriver-1` or `lvmdriver-2`. The capacity filter scheduler is enabled by default. The next section provides more information. In addition, this example presents a `lvmdriver-3` back end.

> **Note**
>
> For Fiber Channel drivers that support multipath, the configuration group requires the `use_multipath_for_image_xfer=true` option. In the example below, you can see details for HPE 3PAR and EMC Fiber Channel drivers.

```
[3par]
use_multipath_for_image_xfer = true
volume_driver = cinder.volume.drivers.hpe.hpe_3par_fc.HPE3PARFCDriver
volume_backend_name = 3parfc

[emc]
use_multipath_for_image_xfer = true
volume_driver = cinder.volume.drivers.emc.emc_smis_fc.EMCSMISFCDriver
volume_backend_name = emcfc
```

Configure Block Storage scheduler multi back end

You must enable the filter_scheduler option to use multiple-storage back ends. The filter scheduler:

1. Filters the available back ends. By default, `AvailabilityZoneFilter`, `CapacityFilter` and `CapabilitiesFilter` are enabled.
2. Weights the previously filtered back ends. By default, the CapacityWeigher option is enabled. When this option is enabled, the filter scheduler assigns the highest weight to back ends with the most available capacity.

The scheduler uses filters and weights to pick the best back end to handle the request. The scheduler uses volume types to explicitly create volumes on specific back ends. For more information about filter and weighing, see *Configure and use driver filter and weighing for scheduler*.

Volume type

Before using it, a volume type has to be declared to Block Storage. This can be done by the following command:

```
$ cinder --os-username admin --os-tenant-name admin type-create lvm
```

Then, an extra-specification has to be created to link the volume type to a back end name. Run this command:

```
$ cinder --os-username admin --os-tenant-name admin type-key lvm set \
  volume_backend_name=LVM_iSCSI
```

This example creates a `lvm` volume type with `volume_backend_name=LVM_iSCSI` as extra-specifications.

Create another volume type:

```
$ cinder --os-username admin --os-tenant-name admin type-create lvm_gold

$ cinder --os-username admin --os-tenant-name admin type-key lvm_gold set \
  volume_backend_name=LVM_iSCSI_b
```

This second volume type is named `lvm_gold` and has `LVM_iSCSI_b` as back end name.

> **Note**
>
> To list the extra-specifications, use this command:
>
> ```
> $ cinder --os-username admin --os-tenant-name admin extra-specs-list
> ```

> **Note**
>
> If a volume type points to a `volume_backend_name` that does not exist in the Block Storage configuration, the `filter_scheduler` returns an error that it cannot find a valid host with the suitable back end.

Usage

When you create a volume, you must specify the volume type. The extra-specifications of the volume type are used to determine which back end has to be used.

```
$ cinder create --volume_type lvm --display_name test_multi_backend 1
```

Considering the `cinder.conf` described previously, the scheduler creates this volume on `lvmdriver-1` or `lvmdriver-2`.

```
$ cinder create --volume_type lvm_gold --display_name test_multi_backend 1
```

This second volume is created on `lvmdriver-3`.

Back up Block Storage service disks

While you can use the LVM snapshot to create snapshots, you can also use it to back up your volumes. By using LVM snapshot, you reduce the size of the backup; only existing data is backed up instead of the entire volume.

To back up a volume, you must create a snapshot of it. An LVM snapshot is the exact copy of a logical volume, which contains data in a frozen state. This prevents data corruption because data cannot be manipulated during the volume creation process. Remember that the volumes created through a **nova volume-create** command exist in an LVM logical volume.

You must also make sure that the operating system is not using the volume and that all data has been flushed on the guest file systems. This usually means that those file systems have to be unmounted during the snapshot creation. They can be mounted again as soon as the logical volume snapshot has been created.

Before you create the snapshot you must have enough space to save it. As a precaution, you should have at least twice as much space as the potential snapshot size. If insufficient space is available, the snapshot might become corrupted.

For this example assume that a 100 GB volume named `volume-00000001` was created for an instance while only 4 GB are used. This example uses these commands to back up only those 4 GB:

- **lvm2** command. Directly manipulates the volumes.
- **kpartx** command. Discovers the partition table created inside the instance.
- **tar** command. Creates a minimum-sized backup.
- **sha1sum** command. Calculates the backup checksum to check its consistency.

You can apply this process to volumes of any size.

To back up Block Storage service disks

1. Create a snapshot of a used volume

   - Use this command to list all volumes

     ```
     # lvdisplay
     ```

   - Create the snapshot; you can do this while the volume is attached to an instance:

     ```
     # lvcreate --size 10G --snapshot --name volume-00000001-snapshot \
       /dev/cinder-volumes/volume-00000001
     ```

     Use the *--snapshot* configuration option to tell LVM that you want a snapshot of an already existing volume. The command includes the size of the space reserved for the snapshot volume, the name of the snapshot, and the path of an already existing volume. Generally, this path is `/dev/cinder-volumes/VOLUME_NAME`.

     The size does not have to be the same as the volume of the snapshot. The *--size* parameter defines the space that LVM reserves for the snapshot volume. As a precaution, the size should be the same as that of the original volume, even if the whole space is not currently used by the snapshot.

   - Run the **lvdisplay** command again to verify the snapshot:

     ```
     --- Logical volume ---
     LV Name                /dev/cinder-volumes/volume-00000001
     VG Name                cinder-volumes
     LV UUID                gI8hta-p21U-IW2q-hRN1-nTzN-UC2G-dKbdKr
     LV Write Access        read/write
     LV snapshot status     source of
                            /dev/cinder-volumes/volume-00000026-snap
     [active]
     LV Status              available
     # open                 1
     LV Size                15,00 GiB
     Current LE             3840
     Segments               1
     Allocation             inherit
     ```

```
Read ahead sectors     auto
- currently set to     256
Block device           251:13

--- Logical volume ---
LV Name                /dev/cinder-volumes/volume-00000001-snap
VG Name                cinder-volumes
LV UUID                HlW3Ep-g5I8-KGQb-IRvi-IRYU-lIKe-wE9zYr
LV Write Access        read/write
LV snapshot status     active destination for /dev/cinder-
volumes/volume-00000026
LV Status              available
# open                 0
LV Size                15,00 GiB
Current LE             3840
COW-table size         10,00 GiB
COW-table LE           2560
Allocated to snapshot  0,00%
Snapshot chunk size    4,00 KiB
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     256
Block device           251:14
```

2. Partition table discovery

   - To exploit the snapshot with the **tar** command, mount your partition on the Block Storage service server.

     The **kpartx** utility discovers and maps table partitions. You can use it to view partitions that are created inside the instance. Without using the partitions created inside instances, you cannot see its content and create efficient backups.

     ```
     # kpartx -av /dev/cinder-volumes/volume-00000001-snapshot
     ```

     > **Note**
     >
     > On a Debian-based distribution, you can use the **apt-get install kpartx** command to install **kpartx**.

     If the tools successfully find and map the partition table, no errors are returned.

- To check the partition table map, run this command:

  ```
  $ ls /dev/mapper/nova*
  ```

  You can see the `cinder--volumes-volume--00000001--snapshot1` partition.

  If you created more than one partition on that volume, you see several partitions; for example: `cinder--volumes-volume--00000001--snapshot2`, `cinder--volumes-volume--00000001--snapshot3`, and so on.

- Mount your partition

  ```
  # mount /dev/mapper/cinder--volumes-volume--volume--00000001--
  snapshot1 /mnt
  ```

  If the partition mounts successfully, no errors are returned.

  You can directly access the data inside the instance. If a message prompts you for a partition or you cannot mount it, determine whether enough space was allocated for the snapshot or the **kpartx** command failed to discover the partition table.

  Allocate more space to the snapshot and try the process again.

3. Use the **tar** command to create archives

   Create a backup of the volume:

   ```
   $ tar --exclude="lost+found" --exclude="some/data/to/exclude" -czf \
     volume-00000001.tar.gz -C /mnt/ /backup/destination
   ```

   This command creates a `tar.gz` file that contains the data, *and data only*. This ensures that you do not waste space by backing up empty sectors.

4. Checksum calculation I

   You should always have the checksum for your backup files. When you transfer the same file over the network, you can run a checksum calculation to ensure that your file was not corrupted during its transfer. The checksum is a unique ID for a file. If the checksums are different, the file is corrupted.

   Run this command to run a checksum for your file and save the result to a file:

   ```
   $ sha1sum volume-00000001.tar.gz > volume-00000001.checksum
   ```

> **Note**
>
> Use the **sha1sum** command carefully because the time it takes to complete the calculation is directly proportional to the size of the file.
>
> Depending on your CPU, the process might take a long time for files larger than around 4 to 6 GB.

5. After work cleaning

   Now that you have an efficient and consistent backup, use this command to clean up the file system:

   - Unmount the volume.

     ```
     $ umount /mnt
     ```

   - Delete the partition table.

     ```
     $ kpartx -dv /dev/cinder-volumes/volume-00000001-snapshot
     ```

   - Remove the snapshot.

     ```
     $ lvremove -f /dev/cinder-volumes/volume-00000001-snapshot
     ```

   Repeat these steps for all your volumes.

6. Automate your backups

   Because more and more volumes might be allocated to your Block Storage service, you might want to automate your backups. The SCR_5005_V01_NUAC-OPENSTACK-EBS-volumes-backup.sh script assists you with this task. The script performs the operations from the previous example, but also provides a mail report and runs the backup based on the `backups_retention_days` setting.

   Launch this script from the server that runs the Block Storage service.

   This example shows a mail report:

   ```
   Backup Start Time - 07/10 at 01:00:01
   Current retention - 7 days

   The backup volume is mounted. Proceed...
   Removing old backups...  : /BACKUPS/EBS-VOL/volume-00000019/volume-
   00000019_28_09_2011.tar.gz
        /BACKUPS/EBS-VOL/volume-00000019 - 0 h 1 m and 21 seconds. Size -
   3,5G
   ```

```
The backup volume is mounted. Proceed...
Removing old backups...  : /BACKUPS/EBS-VOL/volume-0000001a/volume-
0000001a_28_09_2011.tar.gz
     /BACKUPS/EBS-VOL/volume-0000001a - 0 h 4 m and 15 seconds. Size -
6,9G
---------------------------------------
Total backups size - 267G - Used space : 35%
Total execution time - 1 h 75 m and 35 seconds
```

The script also enables you to SSH to your instances and run a **mysqldump** command into them. To make this work, enable the connection to the Compute project keys. If you do not want to run the **mysqldump** command, you can add `enable_mysql_dump=0` to the script to turn off this functionality.

Migrate volumes

OpenStack has the ability to migrate volumes between back-ends which support its volume-type. Migrating a volume transparently moves its data from the current back-end for the volume to a new one. This is an administrator function, and can be used for functions including storage evacuation (for maintenance or decommissioning), or manual optimizations (for example, performance, reliability, or cost).

These workflows are possible for a migration:

1. If the storage can migrate the volume on its own, it is given the opportunity to do so. This allows the Block Storage driver to enable optimizations that the storage might be able to perform. If the back-end is not able to perform the migration, the Block Storage uses one of two generic flows, as follows.
2. If the volume is not attached, the Block Storage service creates a volume and copies the data from the original to the new volume.

> **Note**
>
> While most back-ends support this function, not all do. See the driver documentation in the OpenStack Configuration Reference for more details.

3. If the volume is attached to a VM instance, the Block Storage creates a volume, and calls Compute to copy the data from the original to the new volume. Currently this is supported only by the Compute libvirt driver.

As an example, this scenario shows two LVM back-ends and migrates an attached volume from one to the other. This scenario uses the third migration flow.

First, list the available back-ends:

```
# cinder get-pools
+----------+------------------------------------------------------+
| Property |                        Value                         |
+----------+------------------------------------------------------+
|   name   |           server1@lvmstorage-1#lvmstorage-1          |
+----------+------------------------------------------------------+
+----------+------------------------------------------------------+
| Property |                       Value                          |
+----------+------------------------------------------------------+
|   name   |           server2@lvmstorage-2#lvmstorage-2          |
+----------+------------------------------------------------------+
```

> **Note**
>
> Only Block Storage V2 API supports **get-pools**.

You can also get available back-ends like following:

```
# cinder-manage host list
server1@lvmstorage-1    zone1
server2@lvmstorage-2    zone1
```

But it needs to add pool name in the end. For example, `server1@lvmstorage-1#zone1`.

Next, as the admin user, you can see the current status of the volume (replace the example ID with your own):

```
$ cinder show 6088f80a-f116-4331-ad48-9afb0dfb196c
+------------------------------+--------------------------------------+
|           Property           |                Value                 |
+------------------------------+--------------------------------------+
|          attachments         |                [...]                 |
|       availability_zone      |                zone1                 |
|            bootable          |                False                 |
|           created_at         |      2013-09-01T14:53:22.000000      |
|      display_description     |                 test                 |
|          display_name        |                 test                 |
|              id              | 6088f80a-f116-4331-ad48-9afb0dfb196c |
|           metadata           |                  {}                  |
|     os-vol-host-attr:host    | server1@lvmstorage-1#lvmstorage-1    |
| os-vol-mig-status-attr:migstat |                None                |
| os-vol-mig-status-attr:name_id |                None                |
|  os-vol-tenant-attr:tenant_id  |   6bdd8f41203e4149b5d559769307365e   |
|             size             |                  2                   |
|          snapshot_id         |                None                  |
|          source_volid        |                None                  |
|             status           |               in-use                 |
|          volume_type         |                None                  |
+------------------------------+--------------------------------------+
```

Note these attributes:

- `os-vol-host-attr:host` - the volume's current back-end.
- `os-vol-mig-status-attr:migstat` - the status of this volume's migration (None means that a migration is not currently in progress).
- `os-vol-mig-status-attr:name_id` - the volume ID that this volume's name on the back-end is based on. Before a volume is ever migrated, its name on the back-end storage may be based on the volume's ID (see the `volume_name_template` configuration parameter). For example, if `volume_name_template` is kept as the default value (`volume-%s`), your first LVM back-end has a logical volume named `volume-6088f80a-f116-4331-ad48-9afb0dfb196c`. During the course of a migration, if you create a volume and copy over the data, the volume get the new name but keeps its original ID. This is exposed by the `name_id` attribute.

> **Note**
>
> If you plan to decommission a block storage node, you must stop the `cinder` volume service on the node after performing the migration.
>
> On nodes that run CentOS, Fedora, openSUSE, Red Hat Enterprise Linux, or SUSE Linux Enterprise, run:
>
> # service openstack-cinder-volume stop # chkconfig openstack-cinder-volume off
>
> On nodes that run Ubuntu or Debian, run:
>
> ```
> # service cinder-volume stop
> # chkconfig cinder-volume off
> ```
>
> Stopping the cinder volume service will prevent volumes from being allocated to the node.

Migrate this volume to the second LVM back-end:

```
$ cinder migrate 6088f80a-f116-4331-ad48-9afb0dfb196c \
  server2@lvmstorage-2#lvmstorage-2
```

You can use the **cinder show** command to see the status of the migration. While migrating, the `migstat` attribute shows states such as `migrating` or `completing`. On error, `migstat` is set to None and the host attribute shows the original `host`. On success, in this example, the output looks like:

```
+------------------------------+--------------------------------------+
|            Property          |                Value                 |
+------------------------------+--------------------------------------+
|          attachments         |                [...]                 |
|       availability_zone      |                zone1                 |
|            bootable          |                False                 |
|           created_at         |      2013-09-01T14:53:22.000000      |
|      display_description     |                test                  |
|          display_name        |                test                  |
|               id             | 6088f80a-f116-4331-ad48-9afb0dfb196c |
|           metadata           |                 {}                   |
|     os-vol-host-attr:host    |  server2@lvmstorage-2#lvmstorage-2   |
| os-vol-mig-status-attr:migstat |                None                  |
| os-vol-mig-status-attr:name_id | 133d1f56-9ffc-4f57-8798-d5217d851862 |
|  os-vol-tenant-attr:tenant_id  |   6bdd8f41203e4149b5d559769307365e   |
|              size            |                 2                    |
|          snapshot_id         |                None                  |
|          source_volid        |                None                  |
|             status           |               in-use                 |
|          volume_type         |                None                  |
+------------------------------+--------------------------------------+
```

Note that `migstat` is None, host is the new host, and `name_id` holds the ID of the volume created by the migration. If you look at the second LVM back end, you find the logical volume `volume-133d1f56-9ffc-4f57-8798-d5217d851862`.

> **Note**
>
> The migration is not visible to non-admin users (for example, through the volume `status`). However, some operations are not allowed while a migration is taking place, such as attaching/detaching a volume and deleting a volume. If a user performs such an action during a migration, an error is returned.

> **Note**
>
> Migrating volumes that have snapshots are currently not allowed.

Gracefully remove a GlusterFS volume from usage

Configuring the `cinder` volume service to use GlusterFS involves creating a shares file (for example, `/etc/cinder/glusterfs`). This shares file lists each GlusterFS volume (with its corresponding storage server) that the `cinder` volume service can use for back end storage.

To remove a GlusterFS volume from usage as a back end, delete the volume's corresponding entry from the shares file. After doing so, restart the Block Storage services.

Restarting the Block Storage services will prevent the `cinder` volume service from exporting the deleted GlusterFS volume. This will prevent any instances from mounting the volume from that point onwards.

However, the removed GlusterFS volume might still be mounted on an instance at this point. Typically, this is the case when the volume was already mounted while its entry was deleted from the shares file. Whenever this occurs, you will have to unmount the volume as normal after the Block Storage services are restarted.

Back up and restore volumes and snapshots

The `cinder` command-line interface provides the tools for creating a volume backup. You can restore a volume from a backup as long as the backup's associated database information (or backup metadata) is intact in the Block Storage database.

Run this command to create a backup of a volume:

```
$ cinder backup-create [--incremental] [--force] VOLUME
```

Where `VOLUME` is the name or ID of the volume, `incremental` is a flag that indicates whether an incremental backup should be performed, and `force` is a flag that allows or disallows backup of a volume when the volume is attached to an instance.

Without the `incremental` flag, a full backup is created by default. With the `incremental` flag, an incremental backup is created.

Without the `force` flag, the volume will be backed up only if its status is `available`. With the `force` flag, the volume will be backed up whether its status is `available` or `in-use`. A volume is `in-use` when it is attached to an instance. The backup of an `in-use` volume means your data is crash consistent. The `force` flag is False by default.

> **Note**
>
> The `incremental` and `force` flags are only available for block storage API v2. You have to specify `[--os-volume-api-version 2]` in the `cinder` command-line interface to use this parameter.

> **Note**
>
> The `force` flag is new in OpenStack Liberty.

The incremental backup is based on a parent backup which is an existing backup with the latest timestamp. The parent backup can be a full backup or an incremental backup depending on the timestamp.

> **Note**
>
> The first backup of a volume has to be a full backup. Attempting to do an incremental backup without any existing backups will fail. There is an `is_incremental` flag that indicates whether a backup is incremental when showing details on the backup. Another flag, `has_dependent_backups`, returned when showing backup details, will indicate whether the backup has dependent backups. If it is `true`, attempting to delete this backup will fail.

A new configure option `backup_swift_block_size` is introduced into `cinder.conf` for the default Swift backup driver. This is the size in bytes that changes are tracked for incremental backups. The existing `backup_swift_object_size` option, the size in bytes of Swift backup objects, has to be a multiple of `backup_swift_block_size`. The default is 32768 for `backup_swift_block_size`, and the default is 52428800 for `backup_swift_object_size`.

The configuration option `backup_swift_enable_progress_timer` in `cinder.conf` is used when backing up the volume to Object Storage back end. This option enables or disables the timer. It is enabled by default to send the periodic progress notifications to the Telemetry service.

This command also returns a backup ID. Use this backup ID when restoring the volume:

```
$ cinder backup-restore BACKUP_ID
```

When restoring from a full backup, it is a full restore.

When restoring from an incremental backup, a list of backups is built based on the IDs of the parent backups. A full restore is performed based on the full backup first, then restore is done based on the incremental backup, laying on top of it in order.

You can view a backup list with the **cinder backup-list** command. Optional arguments to clarify the status of your backups include: running *--name*, *--status*, and *--volume-id* to filter through backups by the specified name, status, or volume-id. Search with *--all-tenants* for details of the tenants associated with the listed backups.

Because volume backups are dependent on the Block Storage database, you must also back up your Block Storage database regularly to ensure data recovery.

> **Note**
>
> Alternatively, you can export and save the metadata of selected volume backups. Doing so precludes the need to back up the entire Block Storage database. This is useful if you need only a small subset of volumes to survive a catastrophic database failure.
>
> If you specify a UUID encryption key when setting up the volume specifications, the backup metadata ensures that the key will remain valid when you back up and restore the volume.
>
> For more information about how to export and import volume backup metadata, see the section called *Export and import backup metadata*.

By default, the swift object store is used for the backup repository.

If instead you want to use an NFS export as the backup repository, add the following configuration options to the `[DEFAULT]` section of the `cinder.conf` file and restart the Block Storage services:

```
backup_driver = cinder.backup.drivers.nfs
backup_share = HOST:EXPORT_PATH
```

For the `backup_share` option, replace `HOST` with the DNS resolvable host name or the IP address of the storage server for the NFS share, and `EXPORT_PATH` with the path to that share. If your environment requires that non-default mount options be specified for the share, set these as follows:

```
backup_mount_options = MOUNT_OPTIONS
```

`MOUNT_OPTIONS` is a comma-separated string of NFS mount options as detailed in the NFS man page.

There are several other options whose default values may be overridden as appropriate for your environment:

```
backup_compression_algorithm = zlib
backup_sha_block_size_bytes = 32768
backup_file_size = 1999994880
```

The option `backup_compression_algorithm` can be set to `bz2` or `None`. The latter can be a useful setting when the server providing the share for the backup repository itself performs deduplication or compression on the backup data.

Volumes larger than this will be stored in multiple files in the backup repository. The `backup_sha_block_size_bytes` option determines the size of blocks from the cinder volume being backed up on which digital signatures are calculated in order to enable incremental backup capability.

You also have the option of resetting the state of a backup. When creating or restoring a backup, sometimes it may get stuck in the creating or restoring states due to problems like the database or rabbitmq being down. In situations like these resetting the state of the backup can restore it to a functional status.

Run this command to restore the state of a backup:

```
$ cinder backup-reset-state [--state STATE] BACKUP_ID-1 BACKUP_ID-2 ...
```

Run this command to create a backup of a snapshot:

```
$ cinder backup-create [--incremental] [--force] SNAPSHOT
```

Where `SNAPSHOT` is the name or ID of the snapshot.

Export and import backup metadata

A volume backup can only be restored on the same Block Storage service. This is because restoring a volume from a backup requires metadata available on the database used by the Block Storage service.

> **Note**
>
> For information about how to back up and restore a volume, see the section called *Back up and restore volumes and snapshots*.

You can, however, export the metadata of a volume backup. To do so, run this command as an OpenStack `admin` user (presumably, after creating a volume backup):

```
$ cinder backup-export BACKUP_ID
```

Where `BACKUP_ID` is the volume backup's ID. This command should return the backup's corresponding database information as encoded string metadata.

Exporting and storing this encoded string metadata allows you to completely restore the backup, even in the event of a catastrophic database failure. This will preclude the need to back up the entire Block Storage database, particularly if you only need to keep complete backups of a small subset of volumes.

If you have placed encryption on your volumes, the encryption will still be in place when you restore the volume if a UUID encryption key is specified when creating volumes. Using

backup metadata support, UUID keys set up for a volume (or volumes) will remain valid when you restore a backed-up volume. The restored volume will remain encrypted, and will be accessible with your credentials.

In addition, having a volume backup and its backup metadata also provides volume portability. Specifically, backing up a volume and exporting its metadata will allow you to restore the volume on a completely different Block Storage database, or even on a different cloud service. To do so, first import the backup metadata to the Block Storage database and then restore the backup.

To import backup metadata, run the following command as an OpenStack `admin`:

```
$ cinder backup-import METADATA
```

Where `METADATA` is the backup metadata exported earlier.

Once you have imported the backup metadata into a Block Storage database, restore the volume (see the section called *Back up and restore volumes and snapshots*).

Use LIO iSCSI support

The default mode for the `iscsi_helper` tool is `tgtadm`. To use LIO iSCSI, install the `python-rtslib` package, and set `iscsi_helper=lioadm` in the `cinder.conf` file.

Once configured, you can use the **cinder-rtstool** command to manage the volumes. This command enables you to create, delete, and verify volumes and determine targets and add iSCSI initiators to the system.

Configure and use volume number weigher

OpenStack Block Storage enables you to choose a volume back end according to `free_capacity` and `allocated_capacity`. The volume number weigher feature lets the scheduler choose a volume back end based on its volume number in the volume back end. This can provide another means to improve the volume back ends' I/O balance and the volumes' I/O performance.

Enable volume number weigher

To enable a volume number weigher, set the `scheduler_default_weighers` to `VolumeNumberWeigher` flag in the `cinder.conf` file to define `VolumeNumberWeigher` as the selected weigher.

Configure multiple-storage back ends

To configure `VolumeNumberWeigher`, use `LVMVolumeDriver` as the volume driver.

This configuration defines two LVM volume groups: `stack-volumes` with 10 GB capacity and `stack-volumes-1` with 60 GB capacity. This example configuration defines two back ends:

```
scheduler_default_weighers=VolumeNumberWeigher
enabled_backends=lvmdriver-1,lvmdriver-2
[lvmdriver-1]
volume_group=stack-volumes
volume_driver=cinder.volume.drivers.lvm.LVMVolumeDriver
volume_backend_name=LVM

[lvmdriver-2]
volume_group=stack-volumes-1
volume_driver=cinder.volume.drivers.lvm.LVMVolumeDriver
volume_backend_name=LVM
```

Volume type

Define a volume type in Block Storage:

```
$ cinder type-create lvm
```

Create an extra specification that links the volume type to a back-end name:

```
$ cinder type-key lvm set volume_backend_name=LVM
```

This example creates a lvm volume type with `volume_backend_name=LVM` as extra specifications.

Usage

To create six 1-GB volumes, run the **cinder create –volume-type lvm 1** command six times:

```
$ cinder create --volume-type lvm 1
```

This command creates three volumes in `stack-volumes` and three volumes in `stack-volumes-1`.

List the available volumes:

```
# lvs
LV                                          VG             Attr      LSize
Pool Origin Data%  Move Log Copy%  Convert
volume-3814f055-5294-4796-b5e6-1b7816806e5d stack-volumes   -wi-a----  1.00g
volume-72cf5e79-99d2-4d23-b84e-1c35d3a293be stack-volumes   -wi-a----  1.00g
volume-96832554-0273-4e9d-902b-ad421dfb39d1 stack-volumes   -wi-a----  1.00g
```

```
volume-169386ef-3d3e-4a90-8439-58ceb46889d9 stack-volumes-1 -wi-a----  1.00g
volume-460b0bbb-d8a0-4bc3-9882-a129a5fe8652 stack-volumes-1 -wi-a----  1.00g
volume-9a08413b-0dbc-47c9-afb8-41032ab05a41 stack-volumes-1 -wi-a----  1.00g
```

Consistency groups

Consistency group support is available in OpenStack Block Storage. The support is added for creating snapshots of consistency groups. This feature leverages the storage level consistency technology. It allows snapshots of multiple volumes in the same consistency group to be taken at the same point-in-time to ensure data consistency. The consistency group operations can be performed using the Block Storage command line.

> **Note**
>
> Only Block Storage V2 API supports consistency groups. You can specify *--os-volume-api-version 2* when using Block Storage command line for consistency group operations.

Before using consistency groups, make sure the Block Storage driver that you are running has consistency group support by reading the Block Storage manual or consulting the driver maintainer. There are a small number of drivers that have implemented this feature. The default LVM driver does not support consistency groups yet because the consistency technology is not available at the storage level.

Before using consistency groups, you must change policies for the consistency group APIs in the `/etc/cinder/policy.json` file. By default, the consistency group APIs are disabled. Enable them before running consistency group operations.

Here are existing policy entries for consistency groups:

```
"consistencygroup:create": "group:nobody",
"consistencygroup:delete": "group:nobody",
"consistencygroup:update": "group:nobody",
"consistencygroup:get": "group:nobody",
"consistencygroup:get_all": "group:nobody",
"consistencygroup:create_cgsnapshot" : "group:nobody",
"consistencygroup:delete_cgsnapshot": "group:nobody",
"consistencygroup:get_cgsnapshot": "group:nobody",
"consistencygroup:get_all_cgsnapshots": "group:nobody",
```

Remove `group:nobody` to enable these APIs:

```
"consistencygroup:create": "",
"consistencygroup:delete": "",
"consistencygroup:update": "",
"consistencygroup:get": "",
```

```
"consistencygroup:get_all": "",
"consistencygroup:create_cgsnapshot" : "",
"consistencygroup:delete_cgsnapshot": "",
"consistencygroup:get_cgsnapshot": "",
"consistencygroup:get_all_cgsnapshots": "",
```

Restart Block Storage API service after changing policies.

The following consistency group operations are supported:

- Create a consistency group, given volume types.

> **Note**
>
> A consistency group can support more than one volume type. The scheduler is responsible for finding a back end that can support all given volume types.
>
> A consistency group can only contain volumes hosted by the same back end.
>
> A consistency group is empty upon its creation. Volumes need to be created and added to it later.

- Show a consistency group.
- List consistency groups.
- Create a volume and add it to a consistency group, given volume type and consistency group id.
- Create a snapshot for a consistency group.
- Show a snapshot of a consistency group.
- List consistency group snapshots.
- Delete a snapshot of a consistency group.
- Delete a consistency group.
- Modify a consistency group.
- Create a consistency group from the snapshot of another consistency group.
- Create a consistency group from a source consistency group.

The following operations are not allowed if a volume is in a consistency group:

- Volume migration.
- Volume retype.
- Volume deletion.

> **Note**
>
> A consistency group has to be deleted as a whole with all the volumes.

The following operations are not allowed if a volume snapshot is in a consistency group snapshot:

- Volume snapshot deletion.

> **Note**
>
> A consistency group snapshot has to be deleted as a whole with all the volume snapshots.

The details of consistency group operations are shown in the following.

Create a consistency group:

```
cinder consisgroup-create
[--name name]
[--description description]
[--availability-zone availability-zone]
volume-types
```

> **Note**
>
> The parameter `volume-types` is required. It can be a list of names or UUIDs of volume types separated by commas without spaces in between. For example, `volumetype1,volumetype2,volumetype3..`

```
$ cinder consisgroup-create --name bronzeCG2 volume_type_1
+------------------+--------------------------------------+
|      Property    |                Value                 |
+------------------+--------------------------------------+
| availability_zone|                nova                  |
|     created_at   |      2014-12-29T12:59:08.000000      |
|    description   |                None                  |
|         id       | 1de80c27-3b2f-47a6-91a7-e867cbe36462 |
|        name      |              bronzeCG2               |
|       status     |              creating                |
+------------------+--------------------------------------+
```

Show a consistency group:

```
$ cinder consisgroup-show 1de80c27-3b2f-47a6-91a7-e867cbe36462
```

```
+-------------------+--------------------------------------+
|      Property     |                Value                 |
+-------------------+--------------------------------------+
| availability_zone |                 nova                 |
|     created_at    |      2014-12-29T12:59:08.000000      |
|    description    |                 None                 |
|         id        | 2a6b2bda-1f43-42ce-9de8-249fa5cbae9a |
|        name       |              bronzeCG2               |
|       status      |              available               |
|     volume_types  |              volume_type_1           |
+-------------------+--------------------------------------+
```

List consistency groups:

```
$ cinder consisgroup-list
```

```
+--------------------------------------+-----------+-----------+
|                  ID                  |   Status  |    Name   |
+--------------------------------------+-----------+-----------+
| 1de80c27-3b2f-47a6-91a7-e867cbe36462 | available | bronzeCG2 |
| 3a2b3c42-b612-479a-91eb-1ed45b7f2ad5 |   error   |  bronzeCG |
+--------------------------------------+-----------+-----------+
```

Create a volume and add it to a consistency group:

> **Note**
>
> When creating a volume and adding it to a consistency group, a volume type and a consistency group id must be provided. This is because a consistency group can support more than one volume type.

```
$ cinder create --volume-type volume_type_1 --name cgBronzeVol\
  --consisgroup-id 1de80c27-3b2f-47a6-91a7-e867cbe36462 1
```

```
+---------------------------------------+--------------------------------------+
|                Property               |                Value                 |
+---------------------------------------+--------------------------------------+
|              attachments              |                  []                  |
|           availability_zone           |                 nova                 |
|                bootable               |                false                 |
|          consistencygroup_id          | 1de80c27-3b2f-47a6-91a7-e867cbe36462 |
|               created_at              |      2014-12-29T13:16:47.000000      |
|              description              |                 None                 |
|               encrypted               |                False                 |
|                   id                  | 5e6d1386-4592-489f-a56b-9394a81145fe |
|               metadata                |                  {}                  |
|                  name                 |             cgBronzeVol              |
|         os-vol-host-attr:host         |      server-1@backend-1#pool-1       |
```

```
|     os-vol-mig-status-attr:migstat    |                 None                 |
|     os-vol-mig-status-attr:name_id    |                 None                 |
|      os-vol-tenant-attr:tenant_id     |   1349b21da2a046d8aa5379f0ed447bed   |
|   os-volume-replication:driver_data   |                 None                 |
| os-volume-replication:extended_status |                 None                 |
|           replication_status          |               disabled               |
|                  size                 |                  1                   |
|              snapshot_id              |                 None                 |
|              source_volid             |                 None                 |
|                 status                |               creating               |
|               user_id                 |   93bdea12d3e04c4b86f9a9f172359859   |
|              volume_type              |            volume_type_1             |
+---------------------------------------+--------------------------------------+
```

Create a snapshot for a consistency group:

```
$ cinder cgsnapshot-create 1de80c27-3b2f-47a6-91a7-e867cbe36462
```

```
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
| consistencygroup_id | 1de80c27-3b2f-47a6-91a7-e867cbe36462 |
|      created_at     |      2014-12-29T13:19:44.000000      |
|     description     |                 None                 |
|          id         | d4aff465-f50c-40b3-b088-83feb9b349e9 |
|         name        |                 None                 |
|        status       |               creating               |
+---------------------+--------------------------------------+
```

Show a snapshot of a consistency group:

```
$ cinder cgsnapshot-show d4aff465-f50c-40b3-b088-83feb9b349e9
```

List consistency group snapshots:

```
$ cinder cgsnapshot-list
```

```
+--------------------------------------+--------+----------+
|                  ID                  | Status | Name     |
+--------------------------------------+--------+----------+
| 6d9dfb7d-079a-471e-b75a-6e9185ba0c38 | available  | None |
| aa129f4d-d37c-4b97-9e2d-7efffda29de0 | available  | None |
| bb5b5d82-f380-4a32-b469-3ba2e299712c | available  | None |
| d4aff465-f50c-40b3-b088-83feb9b349e9 | available  | None |
+--------------------------------------+--------+----------+
```

Delete a snapshot of a consistency group:

```
$ cinder cgsnapshot-delete d4aff465-f50c-40b3-b088-83feb9b349e9
```

Delete a consistency group:

> **Note**
>
> The force flag is needed when there are volumes in the consistency group:
>
> $ cinder consisgroup-delete --force 1de80c27-3b2f-47a6-91a7-e867cbe36462

Modify a consistency group:

```
cinder consisgroup-update
[--name NAME]
[--description DESCRIPTION]
[--add-volumes UUID1,UUID2,......]
[--remove-volumes UUID3,UUID4,......]
CG
```

The parameter `CG` is required. It can be a name or UUID of a consistency group. UUID1,UUID2,..... are UUIDs of one or more volumes to be added to the consistency group, separated by commas. Default is None. UUID3,UUID4,..... are UUIDs of one or more volumes to be removed from the consistency group, separated by commas. Default is None.

```
$ cinder consisgroup-update --name 'new name' --description 'new descripti\
  on' --add-volumes 0b3923f5-95a4-4596-a536-914c2c84e2db,1c02528b-3781-4e3\
  2-929c-618d81f52cf3 --remove-volumes 8c0f6ae4-efb1-458f-a8fc-9da2afcc5fb\
  1,a245423f-bb99-4f94-8c8c-02806f9246d8 1de80c27-3b2f-47a6-91a7-e867cbe36462
```

Create a consistency group from the snapshot of another consistency group:

```
$ cinder consisgroup-create-from-src
[--cgsnapshot CGSNAPSHOT]
[--name NAME]
[--description DESCRIPTION]
```

The parameter `CGSNAPSHOT` is a name or UUID of a snapshot of a consistency group:

```
$ cinder consisgroup-create-from-src --cgsnapshot 6d9dfb7d-079a-471e-b75a-\
  6e9185ba0c38 --name 'new cg' --description 'new cg from cgsnapshot'
```

Create a consistency group from a source consistency group:

```
$ cinder consisgroup-create-from-src
[--source-cg SOURCECG]
[--name NAME]
[--description DESCRIPTION]
```

The parameter SOURCECG is a name or UUID of a source consistency group:

```
$ cinder consisgroup-create-from-src --source-cg 6d9dfb7d-079a-471e-b75a-\
  6e9185ba0c38 --name 'new cg' --description 'new cloned cg'
```

Configure and use driver filter and weighing for scheduler

OpenStack Block Storage enables you to choose a volume back end based on back-end specific properties by using the DriverFilter and GoodnessWeigher for the scheduler. The driver filter and weigher scheduling can help ensure that the scheduler chooses the best back end based on requested volume properties as well as various back-end specific properties.

What is driver filter and weigher and when to use it

The driver filter and weigher gives you the ability to more finely control how the OpenStack Block Storage scheduler chooses the best back end to use when handling a volume request. One example scenario where using the driver filter and weigher can be if a back end that utilizes thin-provisioning is used. The default filters use the `free capacity` property to determine the best back end, but that is not always perfect. If a back end has the ability to provide a more accurate back-end specific value you can use that as part of the weighing. Another example of when the driver filter and weigher can prove useful is if a back end exists where there is a hard limit of 1000 volumes. The maximum volume size is 500 GB. Once 75% of the total space is occupied the performance of the back end degrades. The driver filter and weigher can provide a way for these limits to be checked for.

Enable driver filter and weighing

To enable the driver filter, set the `scheduler_default_filters` option in the `cinder.conf` file to `DriverFilter` or add it to the list if other filters are already present.

To enable the goodness filter as a weigher, set the `scheduler_default_weighers` option in the `cinder.conf` file to `GoodnessWeigher` or add it to the list if other weighers are already present.

You can choose to use the `DriverFilter` without the `GoodnessWeigher` or vice-versa. The filter and weigher working together, however, create the most benefits when helping the scheduler choose an ideal back end.

> **Important**
>
> The support for the `DriverFilter` and `GoodnessWeigher` is optional for back ends. If you are using a back end that does not support the filter and weigher functionality you may not get the full benefit.

Example `cinder.conf` configuration file:

```
scheduler_default_filters = DriverFilter
scheduler_default_weighers = GoodnessWeigher
```

> **Note**
>
> It is useful to use the other filters and weighers available in OpenStack in combination with these custom ones. For example, the `CapacityFilter` and `CapacityWeigher` can be combined with these.

Defining your own filter and goodness functions

You can define your own filter and goodness functions through the use of various properties that OpenStack Block Storage has exposed. Properties exposed include information about the volume request being made, `volume_type` settings, and back-end specific information about drivers. All of these allow for a lot of control over how the ideal back end for a volume request will be decided.

The `filter_function` option is a string defining an equation that will determine whether a back end should be considered as a potential candidate in the scheduler.

The `goodness_function` option is a string defining an equation that will rate the quality of the potential host (0 to 100, 0 lowest, 100 highest).

> **Important**
>
> The drive filter and weigher will use default values for filter and goodness functions for each back end if you do not define them yourself. If complete control is desired then a filter and goodness function should be defined for each of the back ends in the `cinder.conf` file.

Supported operations in filter and goodness functions

Below is a table of all the operations currently usable in custom filter and goodness functions created by you:

Operations	Type
+, -, *, /, ^	standard math
not, and, or, &, |, !	logic
>, >=, <, <=, ==, <>, !=	equality
+, -	sign
x ? a : b	ternary
abs(x), max(x, y), min(x, y)	math helper functions

> **Caution**
>
> Syntax errors you define in filter or goodness strings are thrown at a volume request time.

Available properties when creating custom functions

There are various properties that can be used in either the `filter_function` or the `goodness_function` strings. The properties allow access to volume info, qos settings, extra specs, and so on.

The following properties and their sub-properties are currently available for use:

Host stats for a back end

host
: The host's name

volume_backend_name
: The volume back end name

vendor_name
: The vendor name

driver_version
: The driver version

storage_protocol
: The storage protocol

QoS_support
: Boolean signifying whether QoS is supported

total_capacity_gb
: The total capacity in GB

allocated_capacity_gb
The allocated capacity in GB
reserved_percentage
The reserved storage percentage

Capabilities specific to a back end

These properties are determined by the specific back end you are creating filter and goodness functions for. Some back ends may not have any properties available here.

Requested volume properties

status
Status for the requested volume
volume_type_id
The volume type ID
display_name
The display name of the volume
volume_metadata
Any metadata the volume has
reservations
Any reservations the volume has
user_id
The volume's user ID
attach_status
The attach status for the volume
display_description
The volume's display description
id
The volume's ID
replication_status
The volume's replication status
snapshot_id
The volume's snapshot ID
encryption_key_id
The volume's encryption key ID
source_volid
The source volume ID
volume_admin_metadata
Any admin metadata for this volume
source_replicaid
The source replication ID
consistencygroup_id

The consistency group ID

size

The size of the volume in GB

metadata

General metadata

The property most used from here will most likely be the `size` sub-property.

Extra specs for the requested volume type

View the available properties for volume types by running:

```
$ cinder extra-specs-list
```

Current QoS specs for the requested volume type

View the available properties for volume types by running:

```
$ cinder qos-list
```

In order to access these properties in a custom string use the following format:

```
<property>.<sub_property>
```

Driver filter and weigher usage examples

Below are examples for using the filter and weigher separately, together, and using driver-specific properties.

Example `cinder.conf` file configuration for customizing the filter function:

```
[default]
scheduler_default_filters = DriverFilter
enabled_backends = lvm-1, lvm-2

[lvm-1]
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_backend_name = sample_LVM
filter_function = "volume.size < 10"

[lvm-2]
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_backend_name = sample_LVM
filter_function = "volume.size >= 10"
```

The above example will filter volumes to different back ends depending on the size of the requested volume. Default OpenStack Block Storage scheduler weighing is done. Volumes

with a size less than 10 GB are sent to lvm-1 and volumes with a size greater than or equal to 10 GB are sent to lvm-2.

Example `cinder.conf` file configuration for customizing the goodness function:

```
[default]
scheduler_default_weighers = GoodnessWeigher
enabled_backends = lvm-1, lvm-2

[lvm-1]
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_backend_name = sample_LVM
goodness_function = "(volume.size < 5) ? 100 : 50"

[lvm-2]
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_backend_name = sample_LVM
goodness_function = "(volume.size >= 5) ? 100 : 25"
```

The above example will determine the goodness rating of a back end based off of the requested volume's size. Default OpenStack Block Storage scheduler filtering is done. The example shows how the ternary if statement can be used in a filter or goodness function. If a requested volume is of size 10 GB then lvm-1 is rated as 50 and lvm-2 is rated as 100. In this case lvm-2 wins. If a requested volume is of size 3 GB then lvm-1 is rated 100 and lvm-2 is rated 25. In this case lvm-1 would win.

Example `cinder.conf` file configuration for customizing both the filter and goodness functions:

```
[default]
scheduler_default_filters = DriverFilter
scheduler_default_weighers = GoodnessWeigher
enabled_backends = lvm-1, lvm-2

[lvm-1]
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_backend_name = sample_LVM
filter_function = "stats.total_capacity_gb < 500"
goodness_function = "(volume.size < 25) ? 100 : 50"

[lvm-2]
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_backend_name = sample_LVM
filter_function = "stats.total_capacity_gb >= 500"
goodness_function = "(volume.size >= 25) ? 100 : 75"
```

The above example combines the techniques from the first two examples. The best back end is now decided based off of the total capacity of the back end and the requested volume's size.

Example `cinder.conf` file configuration for accessing driver specific properties:

```
[default]
scheduler_default_filters = DriverFilter
scheduler_default_weighers = GoodnessWeigher
enabled_backends = lvm-1,lvm-2,lvm-3

[lvm-1]
volume_group = stack-volumes-lvmdriver-1
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_backend_name = lvmdriver-1
filter_function = "volume.size < 5"
goodness_function = "(capabilities.total_volumes < 3) ? 100 : 50"

[lvm-2]
volume_group = stack-volumes-lvmdriver-2
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_backend_name = lvmdriver-2
filter_function = "volumes.size < 5"
goodness_function = "(capabilities.total_volumes < 8) ? 100 : 50"

[lvm-3]
volume_group = stack-volumes-lvmdriver-3
volume_driver = cinder.volume.drivers.LVMVolumeDriver
volume_backend_name = lvmdriver-3
goodness_function = "55"
```

The above is an example of how back-end specific properties can be used in the filter and goodness functions. In this example the LVM driver's `total_volumes` capability is being used to determine which host gets used during a volume request. In the above example, lvm-1 and lvm-2 will handle volume requests for all volumes with a size less than 5 GB. The lvm-1 host will have priority until it contains three or more volumes. After than lvm-2 will have priority until it contains eight or more volumes. The lvm-3 will collect all volumes greater or equal to 5 GB as well as all volumes once lvm-1 and lvm-2 lose priority.

Rate-limit volume copy bandwidth

When you create a new volume from an image or an existing volume, or when you upload a volume image to the Image service, large data copy may stress disk and network bandwidth. To mitigate slow down of data access from the instances, OpenStack Block

Storage supports rate-limiting of volume data copy bandwidth.

Configure volume copy bandwidth limit

To configure the volume copy bandwidth limit, set the `volume_copy_bps_limit` option in the configuration groups for each back end in the `cinder.conf` file. This option takes the integer of maximum bandwidth allowed for volume data copy in byte per second. If this option is set to `0`, the rate-limit is disabled.

While multiple volume data copy operations are running in the same back end, the specified bandwidth is divided to each copy.

Example `cinder.conf` configuration file to limit volume copy bandwidth of `lvmdriver-1` up to 100 MiB/s:

```
[lvmdriver-1]
volume_group=cinder-volumes-1
volume_driver=cinder.volume.drivers.lvm.LVMVolumeDriver
volume_backend_name=LVM
volume_copy_bps_limit=104857600
```

> **Note**
>
> This feature requires libcgroup to set up blkio cgroup for disk I/O bandwidth limit. The libcgroup is provided by the cgroup-bin package in Debian and Ubuntu, or by the libcgroup-tools package in Fedora, Red Hat Enterprise Linux, CentOS, openSUSE, and SUSE Linux Enterprise.
>
> **Note**
>
> Some back ends which use remote file systems such as NFS are not supported by this feature.

Oversubscription in thin provisioning

OpenStack Block Storage enables you to choose a volume back end based on virtual capacities for thin provisioning using the oversubscription ratio.

A reference implementation is provided for the default LVM driver. The illustration below uses the LVM driver as an example.

Configure oversubscription settings

To support oversubscription in thin provisioning, a flag `max_over_subscription_ratio` is

introduced into `cinder.conf`. This is a float representation of the oversubscription ratio when thin provisioning is involved. Default ratio is 20.0, meaning provisioned capacity can be 20 times of the total physical capacity. A ratio of 10.5 means provisioned capacity can be 10.5 times of the total physical capacity. A ratio of 1.0 means provisioned capacity cannot exceed the total physical capacity. A ratio lower than 1.0 is ignored and the default value is used instead.

> **Note**
>
> `max_over_subscription_ratio` can be configured for each back end when multiple-storage back ends are enabled. It is provided as a reference implementation and is used by the LVM driver. However, it is not a requirement for a driver to use this option from `cinder.conf`.
>
> `max_over_subscription_ratio` is for configuring a back end. For a driver that supports multiple pools per back end, it can report this ratio for each pool. The LVM driver does not support multiple pools.

The existing `reserved_percentage` flag is used to prevent over provisioning. This flag represents the percentage of the back-end capacity that is reserved.

> **Note**
>
> There is a change on how `reserved_percentage` is used. It was measured against the free capacity in the past. Now it is measured against the total capacity.

Capabilities

Drivers can report the following capabilities for a back end or a pool:

```
thin_provisioning_support = True(or False)
thick_provisioning_support = True(or False)
provisioned_capacity_gb = PROVISIONED_CAPACITY
max_over_subscription_ratio = MAX_RATIO
```

Where `PROVISIONED_CAPACITY` is the apparent allocated space indicating how much capacity has been provisioned and `MAX_RATIO` is the maximum oversubscription ratio. For the LVM driver, it is `max_over_subscription_ratio` in `cinder.conf`.

Two capabilities are added here to allow a back end or pool to claim support for thin provisioning, or thick provisioning, or both.

The LVM driver reports `thin_provisioning_support=True` and

`thick_provisioning_support=False` if the `lvm_type` flag in `cinder.conf` is `thin`. Otherwise it reports `thin_provisioning_support=False` and `thick_provisioning_support=True`.

Volume type extra specs

If volume type is provided as part of the volume creation request, it can have the following extra specs defined:

```
'capabilities:thin_provisioning_support': '<is> True' or '<is> False'
'capabilities:thick_provisioning_support': '<is> True' or '<is> False'
```

> **Note**
>
> `capabilities` scope key before `thin_provisioning_support` and `thick_provisioning_support` is not required. So the following works too:

```
'thin_provisioning_support': '<is> True' or '<is> False'
'thick_provisioning_support': '<is> True' or '<is> False'
```

The above extra specs are used by the scheduler to find a back end that supports thin provisioning, thick provisioning, or both to match the needs of a specific volume type.

Volume replication extra specs

OpenStack Block Storage has the ability to create volume replicas. Administrators can define a storage policy that includes replication by adjusting the cinder volume driver. Volume replication for OpenStack Block Storage helps safeguard OpenStack environments from data loss during disaster recovery.

To enable replication when creating volume types, configure the cinder volume with `capabilities:replication="<is> True"`.

Each volume created with the replication capability set to `True` generates a copy of the volume on a storage back end.

One use case for replication involves an OpenStack cloud environment installed across two data centers located nearby each other. The distance between the two data centers in this use case is the length of a city.

At each data center, a cinder host supports the Block Storage service. Both data centers include storage back ends.

Depending on the storage requirements, there can be one or two cinder hosts. The administrator accesses the `/etc/cinder/cinder.conf` configuration file and sets `capabilities:replication="<is> True"`.

If one data center experiences a service failure, administrators can redeploy the VM. The VM will run using a replicated, backed up volume on a host in the second data center.

Capacity filter

In the capacity filter, `max_over_subscription_ratio` is used when choosing a back end if `thin_provisioning_support` is True and `max_over_subscription_ratio` is greater than 1.0.

Capacity weigher

In the capacity weigher, virtual free capacity is used for ranking if `thin_provisioning_support` is True. Otherwise, real free capacity will be used as before.

Image-Volume cache

OpenStack Block Storage has an optional Image cache which can dramatically improve the performance of creating a volume from an image. The improvement depends on many factors, primarily how quickly the configured back end can clone a volume.

When a volume is first created from an image, a new cached image-volume will be created that is owned by the Block Storage Internal Tenant. Subsequent requests to create volumes from that image will clone the cached version instead of downloading the image contents and copying data to the volume.

The cache itself is configurable per back end and will contain the most recently used images.

Configure the Internal Tenant

The Image-Volume cache requires that the Internal Tenant be configured for the Block Storage services. This tenant will own the cached image-volumes so they can be managed like normal users including tools like volume quotas. This protects normal users from having to see the cached image-volumes, but does not make them globally hidden.

To enable the Block Storage services to have access to an Internal Tenant, set the following options in the `cinder.conf` file:

```
cinder_internal_tenant_project_id = PROJECT_ID
cinder_internal_tenant_user_id = USER_ID
```

Example `cinder.conf` configuration file:

```
cinder_internal_tenant_project_id = b7455b8974bb4064ad247c8f375eae6c
cinder_internal_tenant_user_id = f46924c112a14c80ab0a24a613d95eef
```

> **Note**
>
> The actual user and project that are configured for the Internal Tenant do not require any special privileges. They can be the Block Storage service tenant or can be any normal project and user.

Configure the Image-Volume cache

To enable the Image-Volume cache, set the following configuration option in `cinder.conf`:

```
image_volume_cache_enabled = True
```

This can be scoped per back end definition or in the default options.

There are optional configuration settings that can limit the size of the cache. These can also be scoped per back end or in the default options in `cinder.conf`:

```
image_volume_cache_max_size_gb = SIZE_GB
image_volume_cache_max_count = MAX_COUNT
```

By default they will be set to 0, which means unlimited.

For example, a configuration which would limit the max size to 200 GB and 50 cache entries will be configured as:

```
image_volume_cache_max_size_gb = 200
image_volume_cache_max_count = 50
```

Notifications

Cache actions will trigger Telemetry messages. There are several that will be sent.

- `image_volume_cache.miss` - A volume is being created from an image which was not found in the cache. Typically this will mean a new cache entry would be created for it.
- `image_volume_cache.hit` - A volume is being created from an image which was found in the cache and the fast path can be taken.
- `image_volume_cache.evict` - A cached image-volume has been deleted from the cache.

Managing cached Image-Volumes

In normal usage there should be no need for manual intervention with the cache. The entries and their backing Image-Volumes are managed automatically.

If needed, you can delete these volumes manually to clear the cache. By using the standard volume deletion APIs, the Block Storage service will clean up correctly.

Volume-backed image

OpenStack Block Storage can quickly create a volume from an image that refers to a volume storing image data (Image-Volume). Compared to the other stores such as file and swift, creating a volume from a Volume-backed image performs better when the block storage driver supports efficient volume cloning.

If the image is set to public in the Image service, the volume data can be shared among tenants.

Configure the Volume-backed image

Volume-backed image feature requires locations information from the cinder store of the Image service. To enable the Image service to use the cinder store, add `cinder` to the `stores` option in the `glance_store` section of the `glance-api.conf` file:

```
stores = file, http, swift, cinder
```

To expose locations information, set the following options in the `DEFAULT` section of the `glance-api.conf` file:

```
show_multiple_locations = True
```

To enable the Block Storage services to create a new volume by cloning Image- Volume, set the following options in the `DEFAULT` section of the `cinder.conf` file. For example:

```
glance_api_version = 2
allowed_direct_url_schemes = cinder
```

To enable the **openstack image create –volume <volume>** command to create an image that refers an `Image-Volume`, set the following options in each back-end section of the `cinder.conf` file:

```
image_upload_use_cinder_backend = True
```

By default, the **openstack image create –volume <volume>** command creates the Image-Volume in the current tenant. To store the Image-Volume into the internal tenant, set the following options in each back-end section of the `cinder.conf` file:

```
image_upload_use_internal_tenant = True
```

To make the Image-Volume in the internal tenant accessible from the Image service, set the following options in the `glance_store` section of the `glance-api.conf` file:

- `cinder_store_auth_address`
- `cinder_store_user_name`
- `cinder_store_password`

- `cinder_store_project_name`

Creating a Volume-backed image

To register an existing volume as a new Volume-backed image, use the following commands:

```
$ glance image-create --disk-format raw --container-format bare --name <name>

$ glance location-add <image-uuid> --url cinder://<volume-uuid>
```

If the `image_upload_use_cinder_backend` option is enabled, the following command creates a new Image-Volume by cloning the specified volume and then registers its location to a new image. The disk format and the container format must be raw and bare (default). Otherwise, the image is uploaded to the default store of the Image service.

```
$ cinder upload-to-image <volume> <image-name>
```

Get capabilities

When an administrator configures `volume type` and `extra specs` of storage on the back end, the administrator has to read the right documentation that corresponds to the version of the storage back end. Deep knowledge of storage is also required.

OpenStack Block Storage enables administrators to configure `volume type` and `extra specs` without specific knowledge of the storage back end.

> **Note**
>
> - `Volume Type`: A group of volume policies.
> - `Extra Specs`: The definition of a volume type. This is a group of policies. For example, provision type, QOS that will be used to define a volume at creation time.
> - `Capabilities`: What the current deployed back end in Cinder is able to do. These correspond to extra specs.

Usage of cinder client

When an administrator wants to define new volume types for their OpenStack cloud, the administrator would fetch a list of `capabilities` for a particular back end using the cinder client.

First, get a list of the services:

```
$ cinder service-list
+------------------+--------------------+------+---------+-------+------+
|      Binary      |        Host        | Zone |  Status | State | ...  |
+------------------+--------------------+------+---------+-------+------+
| cinder-scheduler | controller         | nova | enabled |   up  | ...  |
| cinder-volume    | block1@ABC-driver | nova | enabled |   up  | ...  |
+------------------+--------------------+------+---------+-------+------+
```

With one of the listed hosts, pass that to `get-capabilities`, then the administrator can obtain volume stats and also back end `capabilities` as listed below.

```
$ cinder get-capabilities block1@ABC-driver
+---------------------+-----------------------------------------------+
|     Volume stats    |                     Value                     |
+---------------------+-----------------------------------------------+
|     description     |                      None                     |
|     display_name    |   Capabilities of Cinder Vendor ABC driver    |
|    driver_version   |                     2.0.0                     |
|      namespace      | OS::Storage::Capabilities::block1@ABC-driver |
|      pool_name      |                      None                     |
|   storage_protocol  |                     iSCSI                     |
|     vendor_name     |                   Vendor ABC                  |
|      visibility     |                      pool                     |
| volume_backend_name |                   ABC-driver                  |
+---------------------+-----------------------------------------------+
+----------------------+-------------------------------------------------------+
|  Backend properties  |                      Value                            |
+----------------------+-------------------------------------------------------+
|      compression     | {u'type':u'boolean', u'title':u'Compression',  ...} |
| ABC:compression_type | {u'enum':u'['lossy', 'lossless', 'special']',  ...} |
|         qos          | {u'type':u'boolean', u'title':u'QoS',          ...} |
|     replication      | {u'type':u'boolean', u'title':u'Replication',  ...} |
|  thin_provisioning   | {u'type':u'boolean', u'title':u'Thin Provisioning'} |
|     ABC:minIOPS      | {u'type':u'integer', u'title':u'Minimum IOPS QoS',} |
|     ABC:maxIOPS      | {u'type':u'integer', u'title':u'Maximum IOPS QoS',} |
|    ABC:burstIOPS     | {u'type':u'integer', u'title':u'Burst IOPS QoS',..} |
+----------------------+-------------------------------------------------------+
```

Usage of REST API

New endpoint to `get capabilities` list for specific storage back end is also available. For more details, refer to the Block Storage API reference.

API request:

```
GET /v2/{tenant_id}/capabilities/{hostname}
```

Example of return value:

```
{
  "namespace": "OS::Storage::Capabilities::block1@ABC-driver",
  "volume_backend_name": "ABC-driver",
  "pool_name": "pool",
  "driver_version": "2.0.0",
  "storage_protocol": "iSCSI",
  "display_name": "Capabilities of Cinder Vendor ABC driver",
  "description": "None",
  "visibility": "public",
  "properties": {
   "thin_provisioning": {
      "title": "Thin Provisioning",
      "description": "Sets thin provisioning.",
      "type": "boolean"
    },
    "compression": {
      "title": "Compression",
      "description": "Enables compression.",
      "type": "boolean"
    },
    "ABC:compression_type": {
      "title": "Compression type",
      "description": "Specifies compression type.",
      "type": "string",
      "enum": [
        "lossy", "lossless", "special"
      ]
    },
    "replication": {
      "title": "Replication",
      "description": "Enables replication.",
      "type": "boolean"
    },
    "qos": {
```

```
      "title": "QoS",
      "description": "Enables QoS.",
      "type": "boolean"
    },
    "ABC:minIOPS": {
      "title": "Minimum IOPS QoS",
      "description": "Sets minimum IOPS if QoS is enabled.",
      "type": "integer"
    },
    "ABC:maxIOPS": {
      "title": "Maximum IOPS QoS",
      "description": "Sets maximum IOPS if QoS is enabled.",
      "type": "integer"
    },
    "ABC:burstIOPS": {
      "title": "Burst IOPS QoS",
      "description": "Sets burst IOPS if QoS is enabled.",
      "type": "integer"
    },
  }
}
```

Usage of volume type access extension

Some volume types should be restricted only. For example, test volume types where you are testing a new technology or ultra high performance volumes (for special cases) where you do not want most users to be able to select these volumes. An administrator/operator can then define private volume types using cinder client. Volume type access extension adds the ability to manage volume type access. Volume types are public by default. Private volume types can be created by setting the `is_public` Boolean field to `False` at creation time. Access to a private volume type can be controlled by adding or removing a project from it. Private volume types without projects are only visible by users with the admin role/context.

Create a public volume type by setting `is_public` field to `True`:

```
$ cinder type-create --description test1 --is-public True vol_Type1
+--------------------------------------+-----------+-------------+-----------+
|                  ID                  |    Name   | Description | Is_Public |
+--------------------------------------+-----------+-------------+-----------+
| 0a948c84-bad5-4fba-88a2-c062006e4f6b | vol_Type1 |    test1    |    True   |
+--------------------------------------+-----------+-------------+-----------+
```

Create a private volume type by setting `is_public` field to `False`:

```
$ cinder type-create --description test2 --is-public False vol_Type2
+--------------------------------------+-----------+-------------+-----------+
|                  ID                  |    Name   | Description | Is_Public |
+--------------------------------------+-----------+-------------+-----------+
| fd508846-213f-4a07-aaf2-40518fb9a23f | vol_Type2 |    test2    |    False  |
+--------------------------------------+-----------+-------------+-----------+
```

Get a list of the volume types:

```
$ cinder type-list
+--------------------------------------+-------------+-------------+-----------+
|                  ID                  |     Name    | Description | Is_Public |
+--------------------------------------+-------------+-------------+-----------+
| 0a948c84-bad5-4fba-88a2-c062006e4f6b | vol_Type1   |    test1    |    True   |
| 87e5be6f-9491-4ea5-9906-9ac56494bb91 | lvmdriver-1 |      -      |    True   |
| fd508846-213f-4a07-aaf2-40518fb9a23f | vol_Type2   |    test2    |   False   |
+--------------------------------------+-------------+-------------+-----------+
```

Get a list of the projects:

```
$ openstack project list
+----------------------------------+--------------------+
| ID                               | Name               |
+----------------------------------+--------------------+
| 4105ead90a854100ab6b121266707f2b | alt_demo           |
| 4a22a545cedd4fcfa9836eb75e558277 | admin              |
| 71f9cdb1a3ab4b8e8d07d347a2e146bb | service            |
| c4860af62ffe465e99ed1bc08ef6082e | demo               |
| e4b648ba5108415cb9e75bff65fa8068 | invisible_to_admin |
+----------------------------------+--------------------+
```

Add volume type access for the given demo project, using its project-id:

```
$ cinder type-access-add --volume-type vol_Type2 --project-id \
  c4860af62ffe465e99ed1bc08ef6082e
```

List the access information about the given volume type:

```
$ cinder type-access-list --volume-type vol_Type2
+--------------------------------------+----------------------------------+
|            Volume_type_ID            |            Project_ID            |
+--------------------------------------+----------------------------------+
| fd508846-213f-4a07-aaf2-40518fb9a23f | c4860af62ffe465e99ed1bc08ef6082e |
+--------------------------------------+----------------------------------+
```

Remove volume type access for the given project:

```
$ cinder type-access-remove --volume-type vol_Type2 --project-id \
  c4860af62ffe465e99ed1bc08ef6082e
$ cinder type-access-list --volume-type vol_Type2
+----------------+------------+
| Volume_type_ID | Project_ID |
+----------------+------------+
+----------------+------------+
```

Troubleshoot your installation

This section provides useful tips to help you troubleshoot your Block Storage installation.

Troubleshoot the Block Storage configuration

Most Block Storage errors are caused by incorrect volume configurations that result in volume creation failures. To resolve these failures, review these logs:

- `cinder-api` log (`/var/log/cinder/api.log`)
- `cinder-volume` log (`/var/log/cinder/volume.log`)

The `cinder-api` log is useful for determining if you have endpoint or connectivity issues. If you send a request to create a volume and it fails, review the `cinder-api` log to determine whether the request made it to the Block Storage service. If the request is logged and you see no errors or tracebacks, check the `cinder-volume` log for errors or tracebacks.

> **Note**
>
> Create commands are listed in the `cinder-api` log.

These entries in the `cinder.openstack.common.log` file can be used to assist in troubleshooting your Block Storage configuration.

```
# Print debugging output (set logging level to DEBUG instead
# of default WARNING level). (boolean value)
# debug=false

# Print more verbose output (set logging level to INFO instead
# of default WARNING level). (boolean value)
# verbose=false

# Log output to standard error (boolean value)
# use_stderr=true

# Default file mode used when creating log files (string
# value)
# logfile_mode=0644

# format string to use for log messages with context (string
# value)
# logging_context_format_string=%(asctime)s.%(msecs)03d %(levelname)s
# %(name)s [%(request_id)s %(user)s %(tenant)s] %(instance)s%(message)s

# format string to use for log mes #logging_default_format_string=%(asctime)s.
# %(msecs)03d %(process)d %(levelname)s %(name)s [-] %(instance)s%(message)s

# data to append to log format when level is DEBUG (string
# value)
# logging_debug_format_suffix=%(funcName)s %(pathname)s:%(lineno)d

# prefix each line of exception output with this format
# (string value)
# logging_exception_prefix=%(asctime)s.%(msecs)03d %(process)d TRACE %(name)s
# %(instance)s

# list of logger=LEVEL pairs (list value)
# default_log_levels=amqplib=WARN,sqlalchemy=WARN,boto=WARN,suds=INFO,
# keystone=INFO,eventlet.wsgi.server=WARNsages without context
# (string value)
```

```
# If an instance is passed with the log message, format it
# like this (string value)
# instance_format="[instance: %(uuid)s]"

# If an instance UUID is passed with the log message, format
# it like this (string value)
#instance_uuid_format="[instance: %(uuid)s] "

# Format string for %%(asctime)s in log records. Default:
# %(default)s (string value)
# log_date_format=%Y-%m-%d %H:%M:%S

# (Optional) Name of log file to output to. If not set,
# logging will go to stdout. (string value)
# log_file=<None>

# (Optional) The directory to keep log files in (will be
# prepended to --log-file) (string value)
# log_dir=<None>
# instance_uuid_format="[instance: %(uuid)s]"

# If this option is specified, the logging configuration file
# specified is used and overrides any other logging options
# specified. Please see the Python logging module
# documentation for details on logging configuration files.
# (string value)
# Use syslog for logging. (boolean value)
# use_syslog=false

# syslog facility to receive log lines (string value)
# syslog_log_facility=LOG_USER
# log_config=<None>
```

These common issues might occur during configuration, and the following potential solutions describe how to address the issues.

Issues with `state_path` and `volumes_dir` settings

Problem

The OpenStack Block Storage uses `tgtd` as the default iSCSI helper and implements persistent targets. This means that in the case of a `tgt` restart, or even a node reboot, your existing volumes on that node will be restored automatically with their original *IQN*.

By default, Block Storage uses a `state_path` variable, which if installing with Yum or APT

should be set to `/var/lib/cinder/`. The next part is the `volumes_dir` variable, by default this appends a `volumes` directory to the `state_path`. The result is a file-tree: `/var/lib/cinder/volumes/`.

Solution

In order to ensure nodes are restored to their original *IQN*, the iSCSI target information needs to be stored in a file on creation that can be queried in case of restart of the `tgt daemon`. While the installer should handle all this, it can go wrong.

If you have trouble creating volumes and this directory does not exist you should see an error message in the `cinder-volume` log indicating that the `volumes_dir` does not exist, and it should provide information about which path it was looking for.

The persistent tgt include file

Problem

The Block Storage service may have issues locating the persistent `tgt include` file. Along with the `volumes_dir` option, the iSCSI target driver also needs to be configured to look in the correct place for the persistent `tgt include `` file. This is an entry in the ``/etc/tgt/conf.d` file that should have been set during the OpenStack installation.

Solution

If issues occur, verify that you have a `/etc/tgt/conf.d/cinder.conf` file. If the file is not present, create it with:

```
# echo 'include /var/lib/cinder/volumes/ *' >> /etc/tgt/conf.d/cinder.conf
```

No sign of attach call in the `cinder-api` log

Problem

The attach call is unavailable, or not appearing in the `cinder-api` log.

Solution

Adjust the `nova.conf` file, and make sure that your `nova.conf` has this entry:

```
volume_api_class=nova.volume.cinder.API
```

Failed to create iscsi target error in the `cinder-volume.log` file

Problem

```
2013-03-12 01:35:43 1248 TRACE cinder.openstack.common.rpc.amqp \
ISCSITargetCreateFailed: \
Failed to create iscsi target for volume \
```

```
volume-137641b2-af72-4a2f-b243-65fdccd38780.
```

You might see this error in `cinder-volume.log` after trying to create a volume that is 1 GB.

Solution

To fix this issue, change the content of the `/etc/tgt/targets.conf` file from `include /etc/tgt/conf.d/*.conf` to `include /etc/tgt/conf.d/cinder_tgt.conf`, as follows:

```
include /etc/tgt/conf.d/cinder_tgt.conf
include /etc/tgt/conf.d/cinder.conf
default-driver iscsi
```

Restart `tgt` and `cinder-*` services, so they pick up the new configuration.

Multipath call failed exit

Problem

Multipath call failed exit. This warning occurs in the Compute log if you do not have the optional `multipath-tools` package installed on the compute node. This is an optional package and the volume attachment does work without the multipath tools installed. If the `multipath-tools` package is installed on the compute node, it is used to perform the volume attachment. The IDs in your message are unique to your system.

```
WARNING nova.storage.linuxscsi [req-cac861e3-8b29-4143-8f1b-705d0084e571
    admin admin|req-cac861e3-8b29-4143-8f1b-705d0084e571 admin admin]
    Multipath call failed exit (96)
```

Solution

Run the following command on the compute node to install the `multipath-tools` packages.

```
# apt-get install multipath-tools
```

Addressing discrepancies in reported volume sizes for EqualLogic storage

Problem

There is a discrepancy between both the actual volume size in EqualLogic (EQL) storage and the image size in the Image service, with what is reported to OpenStack database. This could lead to confusion if a user is creating volumes from an image that was uploaded

from an EQL volume (through the Image service). The image size is slightly larger than the target volume size; this is because EQL size reporting accounts for additional storage used by EQL for internal volume metadata.

To reproduce the issue follow the steps in the following procedure.

This procedure assumes that the EQL array is provisioned, and that appropriate configuration settings have been included in `/etc/cinder/cinder.conf` to connect to the EQL array.

Create a new volume. Note the ID and size of the volume. In the following example, the ID and size are `74cf9c04-4543-47ae-a937-a9b7c6c921e7` and `1`, respectively:

```
$ cinder create --display-name volume1 1

+---------------------+--------------------------------------+
|   Property          |   Value                              |
+---------------------+--------------------------------------+
|   attachments       |   []                                 |
|   availability zone |   nova                               |
|   bootable          |   false                              |
|   created_at        |   2014-03-21T18:31:54.248775         |
|   display_description |   None                             |
|   display_name      |   volume1                            |
|   id                |   74cf9c04-4543-47ae-a937-a9b7c6c921e7    |
|   metadata          |   {}                                 |
|   size              |   1                                  |
|   snapshot_id       |   None                               |
|   source volid      |   None                               |
|   status            |   creating                           |
|   volume type       |   None                               |
+---------------------+--------------------------------------+
```

Verify the volume size on the EQL array by using its command-line interface.

The actual size (`VolReserve`) is 1.01 GB. The EQL Group Manager should also report a volume size of 1.01 GB:

```
eql> volume select volume-74cf9c04-4543-47ae-a937-a9b7c6c921e7
eql (volume_volume-74cf9c04-4543-47ae-a937-a9b7c6c921e7)> show
_____________________________ Volume Information
_____________________________

Name: volume-74cf9c04-4543-47ae-a937-a9b7c6c921e7
Size: 1GB
VolReserve: 1.01GB
VolReservelnUse: 0MB
ReplReservelnUse: 0MB
```

```
iSCSI Alias: volume-74cf9c04-4543-47ae-a937-a9b7c6c921e7
iSCSI Name: iqn.2001-05.com.equallogic:0-8a0906-19f91850c-067000000b4532cl-
volume-74cf9c04-4543-47ae-a937-a9b7c6c921e7
ActualMembers: 1
Snap-Warn: 10%
Snap-Depletion: delete-oldest
Description:
Snap-Reserve: 100%
Snap-Reserve-Avail: 100% (1.01GB)
Permission: read-write
DesiredStatus: online
Status: online
Connections: 0
Snapshots: 0
Bind:
Type: not-replicated
ReplicationReserveSpace: 0MB
```

Create a new image from this volume:

```
$ cinder upload-to-image --disk-format raw \
  --container-format bare volume1 image_from_volume1

+---------------------+----------------------------------------+
|       Property      |                 Value                  |
+---------------------+----------------------------------------+
|   container_format  |                  bare                  |
|     disk_format     |                  raw                   |
| display_description |                  None                  |
|          id         | 74cf9c04-4543-47ae-a937-a9b7c6c921e7   |
|       image_id      | 3020a21d-ba37-4495-8899-07fc201161b9   |
|      image_name     |          image_from_volume1            |
|         size        |                  1                     |
|        status       |              uploading                 |
|      updated_at     |      2014-03-21T18:31:55.000000        |
|     volume_type     |                  None                  |
+---------------------+----------------------------------------+
```

When you uploaded the volume in the previous step, the Image service reported the volume's size as 1 (GB). However, when using **glance image-list** to list the image, the displayed size is 1085276160 bytes, or roughly 1.01 GB:

Name	Disk Format	Container Format	Size	Status
image_from_volume1	raw	bare	*1085276160*	active

Create a new volume using the previous image (`image_id 3020a21d-ba37-4495 -8899-07fc201161b9` in this example) as the source. Set the target volume size to 1 GB; this is the size reported by the `cinder` tool when you uploaded the volume to the Image service:

```
$ cinder create --display-name volume2 \
  --image-id 3020a21d-ba37-4495-8899-07fc201161b9 1
ERROR: Invalid input received: Size of specified image 2 is larger
than volume size 1. (HTTP 400) (Request-ID: req-4b9369c0-dec5-4e16-a114-
c0cdl6bSd210)
```

The attempt to create a new volume based on the size reported by the `cinder` tool will then fail.

Solution

To work around this problem, increase the target size of the new image to the next whole number. In the problem example, you created a 1 GB volume to be used as volume-backed image, so a new volume using this volume-backed image should use a size of 2 GB:

```
$ cinder create --display-name volume2 \
  --image-id 3020a21d-ba37-4495-8899-07fc201161b9 1

+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|      bootable       |                false                 |
|     created_at      |      2014-03-21T19:25:31.564482      |
| display_description |                None                  |
|    display_name     |              volume2                 |
|          id         | 64e8eb18-d23f-437b-bcac-b3S2afa6843a |
|      image_id       | 3020a21d-ba37-4495-8899-07fc20116lb9 |
|      metadata       |                  []                  |
|         size        |                  2                   |
|    snapshot_id      |                None                  |
|    source_volid     |                None                  |
|       status        |              creating                |
|    volume_type      |                None                  |
+---------------------+--------------------------------------+
```

> **Note**
>
> The dashboard suggests a suitable size when you create a new volume based on a volume-backed image.

You can then check this new volume into the EQL array:

```
eql> volume select volume-64e8eb18-d23f-437b-bcac-b352afa6843a
eql (volume_volume-61e8eb18-d23f-437b-bcac-b352afa6843a)> show
_____________________________ Volume Information
_____________________________
Name: volume-64e8eb18-d23f-437b-bcac-b352afa6843a
Size: 2GB
VolReserve: 2.01GB
VolReserveInUse: 1.01GB
ReplReserveInUse: 0MB
iSCSI Alias: volume-64e8eb18-d23f-437b-bcac-b352afa6843a
iSCSI Name: iqn.2001-05.com.equallogic:0-8a0906-e3091850e-eae000000b7S32cl-
volume-64e8eb18-d23f-437b-bcac-b3S2afa6Bl3a
ActualMembers: 1
Snap-Warn: 10%
Snap-Depletion: delete-oldest
Description:
Snap-Reserve: 100%
Snap-Reserve-Avail: 100% (2GB)
Permission: read-write
DesiredStatus: online
Status: online
Connections: 1
Snapshots: O
Bind:
Type: not-replicated
ReplicationReserveSpace: 0MB
```

Failed to Attach Volume, Missing sg_scan

Problem

Failed to attach volume to an instance, `sg_scan` file not found. This error occurs when the sg3-utils package is not installed on the compute node. The IDs in your message are unique to your system:

```
ERROR nova.compute.manager [req-cf2679fd-dd9e-4909-807f-48fe9bda3642 admin
admin|req-cf2679fd-dd9e-4909-807f-48fe9bda3642 admin admin]
```

```
[instance: 7d7c92e0-49fa-4a8e-87c7-73f22a9585d5|instance:  7d7c92e0-49fa-4a8e-
87c7-73f22a9585d5]
Failed to attach volume  4cc104c4-ac92-4bd6-9b95-c6686746414a at /dev/vdcTRACE
nova.compute.manager
[instance:  7d7c92e0-49fa-4a8e-87c7-73f22a9585d5|instance: 7d7c92e0-49fa-4a8e-
87c7-73f22a9585d5]
Stdout: '/usr/local/bin/nova-rootwrap: Executable not found: /usr/bin/sg_scan'
```

Solution

Run this command on the compute node to install the `sg3-utils` package:

```
# apt-get install sg3-utils
```

HTTP bad request in cinder volume log

Problem

These errors appear in the `cinder-volume.log` file:

```
2013-05-03 15:16:33 INFO [cinder.volume.manager] Updating volume status
2013-05-03 15:16:33 DEBUG [hp3parclient.http]
REQ: curl -i https://10.10.22.241:8080/api/v1/cpgs -X GET -H "X-Hp3Par-Wsapi-Sessionkey:
48dc-b69ed2e5
f259c58e26df9a4c85df110c-8d1e8451" -H "Accept: application/json" -H "User-Agent: python-
3parclient"

2013-05-03 15:16:33 DEBUG [hp3parclient.http] RESP:{'content-length': 311, 'content-
type': 'text/plain',
'status': '400'}

2013-05-03 15:16:33 DEBUG [hp3parclient.http] RESP BODY:Second simultaneous read on
fileno 13 detected.
Unless you really know what you're doing, make sure that only one greenthread can read
any particular socket.
Consider using a pools.Pool. If you do know what you're doing and want to disable this
error,
call eventlet.debug.hub_multiple_reader_prevention(False)

2013-05-03 15:16:33 ERROR [cinder.manager] Error during
VolumeManager._report_driver_status: Bad request (HTTP 400)
Traceback (most recent call last):
File "/usr/lib/python2.7/dist-packages/cinder/manager.py", line 167, in periodic_tasks
task(self, context)
File "/usr/lib/python2.7/dist-packages/cinder/volume/manager.py", line 690, in
_report_driver_status volume_stats =
self.driver.get_volume_stats(refresh=True)
```

```
File "/usr/lib/python2.7/dist-packages/cinder/volume/drivers/san/hp/hp_3par_fc.py", line
77, in get_volume_stats stats =
self.common.get_volume_stats(refresh, self.client)
File "/usr/lib/python2.7/dist-packages/cinder/volume/drivers/san/hp/hp_3par_common.py",
line 421, in get_volume_stats cpg =
client.getCPG(self.config.hp3par_cpg)
File "/usr/lib/python2.7/dist-packages/hp3parclient/client.py", line 231, in getCPG cpgs
= self.getCPGs()
File "/usr/lib/python2.7/dist-packages/hp3parclient/client.py", line 217, in getCPGs
response, body = self.http.get('/cpgs')
File "/usr/lib/python2.7/dist-packages/hp3parclient/http.py", line 255, in get return
self._cs_request(url, 'GET', **kwargs)
File "/usr/lib/python2.7/dist-packages/hp3parclient/http.py", line 224, in _cs_request
**kwargs)
File "/usr/lib/python2.7/dist-packages/hp3parclient/http.py", line 198, in _time_request
resp, body = self.request(url, method, **kwargs)
File "/usr/lib/python2.7/dist-packages/hp3parclient/http.py", line 192, in request raise
exceptions.from_response(resp, body)
HTTPBadRequest: Bad request (HTTP 400)
```

Solution

You need to update your copy of the `hp_3par_fc.py` driver which contains the synchronization code.

Duplicate 3PAR host

Problem

This error may be caused by a volume being exported outside of OpenStack using a host name different from the system name that OpenStack expects. This error could be displayed with the *IQN* if the host was exported using iSCSI:

```
Duplicate3PARHost: 3PAR Host already exists: Host wwn 50014380242B9750 \
already used by host cld4b5ubuntuW(id = 68. The hostname must be called\
'cld4b5ubuntu'.
```

Solution

Change the 3PAR host name to match the one that OpenStack expects. The 3PAR host constructed by the driver uses just the local host name, not the fully qualified domain name (FQDN) of the compute host. For example, if the FQDN was *myhost.example.com*, just *myhost* would be used as the 3PAR host name. IP addresses are not allowed as host names on the 3PAR storage server.

Failed to attach volume after detaching

Problem

Failed to attach a volume after detaching the same volume.

Solution

You must change the device name on the **nova-attach** command. The VM might not clean up after a **nova-detach** command runs. This example shows how the **nova-attach** command fails when you use the vdb, vdc, or vdd device names:

```
# ls -al /dev/disk/by-path/
total 0
drwxr-xr-x 2 root root 200 2012-08-29 17:33 .
drwxr-xr-x 5 root root 100 2012-08-29 17:33 ..
lrwxrwxrwx 1 root root 9 2012-08-29 17:33 pci-0000:00:04.0-virtio-pci-virtio0 ->
../../vda
lrwxrwxrwx 1 root root 10 2012-08-29 17:33 pci-0000:00:04.0-virtio-pci-virtio0-part1
-> ../../vda1
lrwxrwxrwx 1 root root 10 2012-08-29 17:33 pci-0000:00:04.0-virtio-pci-virtio0-part2
-> ../../vda2
lrwxrwxrwx 1 root root 10 2012-08-29 17:33 pci-0000:00:04.0-virtio-pci-virtio0-part5
-> ../../vda5
lrwxrwxrwx 1 root root 9 2012-08-29 17:33 pci-0000:00:06.0-virtio-pci-virtio2 ->
../../vdb
lrwxrwxrwx 1 root root 9 2012-08-29 17:33 pci-0000:00:08.0-virtio-pci-virtio3 ->
../../vdc
lrwxrwxrwx 1 root root 9 2012-08-29 17:33 pci-0000:00:09.0-virtio-pci-virtio4 ->
../../vdd
lrwxrwxrwx 1 root root 10 2012-08-29 17:33 pci-0000:00:09.0-virtio-pci-virtio4-part1
-> ../../vdd1
```

You might also have this problem after attaching and detaching the same volume from the same VM with the same mount point multiple times. In this case, restart the KVM host.

Failed to attach volume, systool is not installed

Problem

This warning and error occurs if you do not have the required `sysfsutils` package installed on the compute node:

```
WARNING nova.virt.libvirt.utils [req-1200f887-c82b-4e7c-a891-fac2e3735dbb\
admin admin|req-1200f887-c82b-4e7c-a891-fac2e3735dbb admin admin] systool\
is not installed
```

```
ERROR nova.compute.manager [req-1200f887-c82b-4e7c-a891-fac2e3735dbb admin\
admin|req-1200f887-c82b-4e7c-a891-fac2e3735dbb admin admin]
[instance: df834b5a-8c3f-477a-be9b-47c97626555c|instance: df834b5a-8c3f-47\
7a-be9b-47c97626555c]
Failed to attach volume 13d5c633-903a-4764-a5a0-3336945b1db1 at /dev/vdk.
```

Solution

Run the following command on the compute node to install the `sysfsutils` packages:

```
# apt-get install sysfsutils
```

Failed to connect volume in FC SAN

Problem

The compute node failed to connect to a volume in a Fibre Channel (FC) SAN configuration. The WWN may not be zoned correctly in your FC SAN that links the compute host to the storage array:

```
ERROR nova.compute.manager [req-2ddd5297-e405-44ab-aed3-152cd2cfb8c2 admin\
demo|req-2ddd5297-e405-44ab-aed3-152cd2cfb8c2 admin demo] [instance: 60ebd\
6c7-c1e3-4bf0-8ef0-f07aa4c3d5f3|instance: 60ebd6c7-c1e3-4bf0-8ef0-f07aa4c3\
d5f3]
Failed to connect to volume 6f6a6a9c-dfcf-4c8d-b1a8-4445ff883200 while\
attaching at /dev/vdjTRACE nova.compute.manager [instance: 60ebd6c7-c1e3-4\
bf0-8ef0-f07aa4c3d5f3|instance: 60ebd6c7-c1e3-4bf0-8ef0-f07aa4c3d5f3]
Traceback (most recent call last):…f07aa4c3d5f3\] ClientException: The\
server has either erred or is incapable of performing the requested\
operation.(HTTP 500)(Request-ID: req-71e5132b-21aa-46ee-b3cc-19b5b4ab2f00)
```

Solution

The network administrator must configure the FC SAN fabric by correctly zoning the WWN (port names) from your compute node HBAs.

Cannot find suitable emulator for x86_64

Problem

When you attempt to create a VM, the error shows the VM is in the `BUILD` then `ERROR` state.

Solution

On the KVM host, run **cat /proc/cpuinfo**. Make sure the `vmx` or `svm` flags are set.

Follow the instructions in the Enable KVM section in the OpenStack Configuration Reference to enable hardware virtualization support in your BIOS.

Non-existent host

Problem

This error could be caused by a volume being exported outside of OpenStack using a host name different from the system name that OpenStack expects. This error could be displayed with the *IQN* if the host was exported using iSCSI.

```
2013-04-19 04:02:02.336 2814 ERROR cinder.openstack.common.rpc.common [-]
Returning exception Not found (HTTP 404)
NON_EXISTENT_HOST - HOST '10' was not found to caller.
```

Solution

Host names constructed by the driver use just the local host name, not the fully qualified domain name (FQDN) of the Compute host. For example, if the FQDN was **myhost.example.com**, just **myhost** would be used as the 3PAR host name. IP addresses are not allowed as host names on the 3PAR storage server.

Non-existent VLUN

Problem

This error occurs if the 3PAR host exists with the correct host name that the OpenStack Block Storage drivers expect but the volume was created in a different domain.

```
HTTPNotFound: Not found (HTTP 404) NON_EXISTENT_VLUN - VLUN 'osv-
DqT7CE3mSrWi4gZJmHAP-Q' was not found.
```

Solution

The `hpe3par_domain` configuration items either need to be updated to use the domain the 3PAR host currently resides in, or the 3PAR host needs to be moved to the domain that the volume was created in.

Shared File Systems

Shared File Systems service provides a set of services for management of shared file systems in a multi-tenant cloud environment. The service resembles OpenStack block-based storage management from the OpenStack Block Storage service project. With the Shared File Systems service, you can create a remote file system, mount the file system on your instances, and then read and write data from your instances to and from your file system.

The Shared File Systems service serves same purpose as the Amazon Elastic File System (EFS) does.

Introduction

The OpenStack File Share service allows you to offer shared file systems service to OpenStack users in your installation. The Shared File Systems service can run in a single-node or multiple node configuration. The Shared File Systems service can be configured to provision shares from one or more back ends, so it is required to declare at least one back end. Shared File System service contains several configurable components.

It is important to understand these components:

- Share networks
- Shares
- Multi-tenancy
- Back ends

The Shared File Systems service consists of four types of services, most of which are

similar to those of the Block Storage service:

- `manila-api`
- `manila-data`
- `manila-scheduler`
- `manila-share`

Installation of first three - `manila-api`, `manila-data`, and `manila-scheduler` is common for almost all deployments. But configuration of `manila-share` is backend-specific and can differ from deployment to deployment.

Key concepts

Share

In the Shared File Systems service `share` is the fundamental resource unit allocated by the Shared File System service. It represents an allocation of a persistent, readable, and writable filesystems. Compute instances access these filesystems. Depending on the deployment configuration, clients outside of OpenStack can also access the filesystem.

> **Note**
>
> A `share` is an abstract storage object that may or may not directly map to a "share" concept from the underlying storage provider. See the description of `share instance` for more details.

Share instance

This concept is tied with `share` and represents created resource on specific back end, when `share` represents abstraction between end user and back-end storages. In common cases, it is one-to-one relation. One single `share` has more than one `share instance` in two cases:

- When `share migration` is being applied
- When `share replication` is enabled

Therefore, each `share instance` stores information specific to real allocated resource on storage. And `share` represents the information that is common for `share instances`. A user with `member` role will not be able to work with it directly. Only a user with `admin` role has rights to perform actions against specific share instances.

Snapshot

A snapshot is a point-in-time, read-only copy of a share. You can create Snapshots from an existing, operational share regardless of whether a client has mounted the file system. A snapshot can serve as the content source for a new share. Specify the **Create from snapshot** option when creating a new share on the dashboard.

Storage Pools

With the Kilo release of OpenStack, Shared File Systems can use storage pools. The storage may present one or more logical storage resource pools that the Shared File Systems service will select as a storage location when provisioning shares.

Share Type

Share type is an abstract collection of criteria used to characterize shares. They are most commonly used to create a hierarchy of functional capabilities. This hierarchy represents tiered storage services levels. For example, an administrator might define a premium share type that indicates a greater level of performance than a basic share type. Premium represents the best performance level.

Share Access Rules

Share access rules define which users can access a particular share. For example, administrators can declare rules for NFS shares by listing the valid IP networks which will access the share. List the IP networks in CIDR notation.

Security Services

Security services``allow granular client access rules for administrators. They can declare rules for authentication or authorization to access ``share content. External services including LDAP, Active Directory, and Kerberos can be declared as resources. Examine and consult these resources when making an access decision for a particular share. You can associate Shares with multiple security services, but only one service per one type.

Share Networks

A share network is an object that defines a relationship between a tenant network and subnet, as defined in an OpenStack Networking service or Compute service. The share network is also defined in shares created by the same tenant. A tenant may find it

desirable to provision shares such that only instances connected to a particular OpenStack-defined network have access to the share. Also, security services can be attached to share networks, because most of auth protocols require some interaction with network services.

The Shared File Systems service has the ability to work outside of OpenStack. That is due to the StandaloneNetworkPlugin. The plugin is compatible with any network platform, and does not require specific network services in OpenStack like Compute or Networking service. You can set the network parameters in the manila.conf file.

Share Servers

A share server is a logical entity that hosts the shares created on a specific share network. A share server may be a configuration object within the storage controller, or it may represent logical resources provisioned within an OpenStack deployment used to support the data path used to access shares.

Share servers interact with network services to determine the appropriate IP addresses on which to export shares according to the related share network. The Shared File Systems service has a pluggable network model that allows share servers to work with different implementations of the Networking service.

Share management

A share is a remote, mountable file system. You can mount a share to and access a share from several hosts by several users at a time.

You can create a share and associate it with a network, list shares, and show information for, update, and delete a specified share. You can also create snapshots of shares. To create a snapshot, you specify the ID of the share that you want to snapshot.

The shares are based on of the supported Shared File Systems protocols:

- *NFS*. Network File System (NFS).
- *CIFS*. Common Internet File System (CIFS).
- *GLUSTERFS*. Gluster file system (GlusterFS).
- *HDFS*. Hadoop Distributed File System (HDFS).
- *CEPHFS*. Ceph File System (CephFS).

The Shared File Systems service provides set of drivers that enable you to use various network file storage devices, instead of the base implementation. That is the real purpose of the Shared File Systems service in production.

Share basic operations

General concepts

To create a file share, and access it, the following general concepts are prerequisite knowledge:

1. To create a share, use **manila create** command and specify the required arguments: the size of the share and the shared file system protocol. `NFS`, `CIFS`, `GlusterFS`, `HDFS`, or `CephFS` share file system protocols are supported.
2. You can also optionally specify the share network and the share type.
3. After the share becomes available, use the **manila show** command to get the share export locations.
4. After getting the share export locations, you can create an *access rule* for the share, mount it and work with files on the remote file system.

There are big number of the share drivers created by different vendors in the Shared File Systems service. As a Python class, each share driver can be set for the *back end* and run in the back end to manage the share operations.

Initially there are two driver modes for the back ends:

- no share servers mode
- share servers mode

Each share driver supports one or two of possible back end modes that can be configured

in the `manila.conf` file. The configuration option `driver_handles_share_servers` in the `manila.conf` file sets the share servers mode or no share servers mode, and defines the driver mode for share storage lifecycle management:

Mode	Config option	Description
no share servers	driver_handles_share_servers = False	An administrator rather than a share driver manages the bare metal storage with some net interface instead of the presence of the share servers.
share servers	driver_handles_share_servers = True	The share driver creates the share server and manages, or handles, the share server life cycle.

It is *the share types* which have the extra specifications that help scheduler to filter back ends and choose the appropriate back end for the user that requested to create a share. The required extra boolean specification for each share type is `driver_handles_share_servers`. As an administrator, you can create the share types with the specifications you need. For details of managing the share types and configuration the back ends, see *Share types* and *Multi-storage configuration* documentation.

You can create a share in two described above modes:

- in a no share servers mode without specifying the share network and specifying the share type with `driver_handles_share_servers = False` parameter. See subsection *Create a share in no share servers mode*.
- in a share servers mode with specifying the share network and the share type with `driver_handles_share_servers = True` parameter. See subsection *Create a share in share servers mode*.

Create a share in no share servers mode

To create a file share in no share servers mode, you need to:

1. To create a share, use **manila create** command and specify the required arguments: the size of the share and the shared file system protocol. `NFS`, `CIFS`, `GlusterFS`, `HDFS`, or `CephFS` share file system protocols are supported.
2. You should specify the *share type* with `driver_handles_share_servers = False` extra specification.
3. You must not specify the `share network` because no share servers are created. In this mode the Shared File Systems service expects that administrator has some bare metal storage with some net interface.

4. The **manila create** command creates a share. This command does the following things:
   - The *manila-scheduler* service will find the back end with `driver_handles_share_servers = False` mode due to filtering the extra specifications of the share type.
   - The share is created using the storage that is specified in the found back end.
5. After the share becomes available, use the **manila show** command to get the share export locations.

In the example to create a share, the created already share type named `my_type` with `driver_handles_share_servers = False` extra specification is used.

Check share types that exist, run:

```
$ manila type-list
+------+---------+------------+------------+-------------------------------------+-------------------------+
| ID   | Name    | visibility | is_default | required_extra_specs                | optional_extra_specs    |
+------+---------+------------+------------+-------------------------------------+-------------------------+
| %ID% | my_type | public     | -          | driver_handles_share_servers : False | snapshot_support : True |
+------+---------+------------+------------+-------------------------------------+-------------------------+
```

Create a private share with `my_type` share type, NFS shared file system protocol, and size 1 GB:

```
$ manila create nfs 1 --name Share1 --description "My share" --share-type
my_type
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| status                      | creating                             |
| share_type_name             | my_type                              |
| description                 | My share                             |
| availability_zone           | None                                 |
| share_network_id            | None                                 |
| share_server_id             | None                                 |
| host                        |                                      |
| access_rules_status         | active                               |
| snapshot_id                 | None                                 |
| is_public                   | False                                |
| task_state                  | None                                 |
| snapshot_support            | True                                 |
| id                          | 10f5a2a1-36f5-45aa-a8e6-00e94e592e88 |
| size                        | 1                                    |
| name                        | Share1                               |
| share_type                  | 14ee8575-aac2-44af-8392-d9c9d344f392 |
| has_replicas                | False                                |
| replication_type            | None                                 |
| created_at                  | 2016-03-25T12:02:46.000000           |
| share_proto                 | NFS                                  |
| consistency_group_id        | None                                 |
| source_cgsnapshot_member_id | None                                 |
| project_id                  | 907004508ef4447397ce6741a8f037c1     |
| metadata                    | {}                                   |
+-----------------------------+--------------------------------------+
```

New share `Share2` should have a status `available`:

```
$ manila show Share2
+----------------------------+------------------------------------------------------------+
| Property                   | Value                                                      |
+----------------------------+------------------------------------------------------------+
| status                     | available                                                  |
| share_type_name            | my_type                                                    |
| description                | My share                                                   |
| availability_zone          | nova                                                       |
| share_network_id           | None                                                       |
| export_locations           |                                                            |
|                            | path = 10.0.0.4:/shares/manila_share_a5fb1ab7_...          |
|                            | preferred = False                                          |
|                            | is_admin_only = False                                      |
|                            | id = 9e078eee-bcad-40b8-b4fe-1c916cf98ed1                  |
|                            | share_instance_id = a5fb1ab7-0bbd-465b-ac14-05706294b6e9   |
|                            | path = 172.18.198.52:/shares/manila_share_a5fb1ab7_...     |
|                            | preferred = False                                          |
|                            | is_admin_only = True                                       |
|                            | id = 44933f59-e0e3-4483-bb88-72ba7c486f41                  |
|                            | share_instance_id = a5fb1ab7-0bbd-465b-ac14-05706294b6e9   |
| share_server_id            | None                                                       |
| host                       | manila@paris#epsilon                                       |
| access_rules_status        | active                                                     |
| snapshot_id                | None                                                       |
| is_public                  | False                                                      |
| task_state                 | None                                                       |
| snapshot_support           | True                                                       |
| id                         | 10f5a2a1-36f5-45aa-a8e6-00e94e592e88                       |
| size                       | 1                                                          |
| name                       | Share1                                                     |
| share_type                 | 14ee8575-aac2-44af-8392-d9c9d344f392                       |
| has_replicas               | False                                                      |
| replication_type           | None                                                       |
| created_at                 | 2016-03-25T12:02:46.000000                                 |
| share_proto                | NFS                                                        |
| consistency_group_id       | None                                                       |
| source_cgsnapshot_member_id | None                                                      |
| project_id                 | 907004508ef4447397ce6741a8f037c1                           |
| metadata                   | {}                                                         |
+----------------------------+------------------------------------------------------------+
```

Create a share in share servers mode

To create a file share in share servers mode, you need to:

1. To create a share, use **manila create** command and specify the required arguments: the size of the share and the shared file system protocol. `NFS`, `CIFS`, `GlusterFS`, `HDFS`, or `CephFS` share file system protocols are supported.
2. You should specify the *share type* with `driver_handles_share_servers = True` extra specification.
3. You should specify the *share network*.
4. The **manila create** command creates a share. This command does the following things:
    - The *manila-scheduler* service will find the back end with

`driver_handles_share_servers = True` mode due to filtering the extra specifications of the share type.

- The share driver will create a share server with the share network. For details of creating the resources, see the documentation of the specific share driver.

5. After the share becomes available, use the **manila show** command to get the share export location.

In the example to create a share, the default share type and the already existing share network are used.

> **Note**
>
> There is no default share type just after you started manila as the administrator. See *Share types* to create the default share type. To create a share network, use *Share networks*.

Check share types that exist, run:

```
$ manila type-list
+------+---------+------------+------------+-------------------------------------+-------------------------+
| ID   | Name    | visibility | is_default | required_extra_specs                | optional_extra_specs    |
+------+---------+------------+------------+-------------------------------------+-------------------------+
| %id% | default | public     | YES        | driver_handles_share_servers : True | snapshot_support : True |
+------+---------+------------+------------+-------------------------------------+-------------------------+
```

Check share networks that exist, run:

```
$ manila share-network-list
+--------------------------------------+--------------+
| id                                   | name         |
+--------------------------------------+--------------+
| c895fe26-92be-4152-9e6c-f2ad230efb13 | my_share_net |
+--------------------------------------+--------------+
```

Create a public share with `my_share_net` network, `default` share type, NFS shared file system protocol, and size 1 GB:

```
$ manila create nfs 1 \
    --name "Share2" \
    --description "My second share" \
    --share-type default \
    --share-network my_share_net \
    --metadata aim=testing \
    --public
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| status                      | creating                             |
| share_type_name             | default                              |
| description                 | My second share                      |
| availability_zone           | None                                 |
```

```
| share_network_id            | c895fe26-92be-4152-9e6c-f2ad230efb13 |
| share_server_id             | None                                 |
| host                        |                                      |
| access_rules_status         | active                               |
| snapshot_id                 | None                                 |
| is_public                   | True                                 |
| task_state                  | None                                 |
| snapshot_support            | True                                 |
| id                          | 195e3ba2-9342-446a-bc93-a584551de0ac |
| size                        | 1                                    |
| name                        | Share2                               |
| share_type                  | bf6ada49-990a-47c3-88bc-c0cb31d5c9bf |
| has_replicas                | False                                |
| replication_type            | None                                 |
| created_at                  | 2016-03-25T12:13:40.000000           |
| share_proto                 | NFS                                  |
| consistency_group_id        | None                                 |
| source_cgsnapshot_member_id | None                                 |
| project_id                  | 907004508ef4447397ce6741a8f037c1     |
| metadata                    | {u'aim': u'testing'}                 |
+-----------------------------+--------------------------------------+
```

The share also can be created from a share snapshot. For details, see *Share snapshots*.

See the share in a share list:

```
$ manila list
+--------------------------------------+--------+------+-------------+-----------+-----------+-----------------+----------------------+-------------------+
| ID                                   | Name   | Size | Share Proto | Status    | Is Public | Share Type Name | Host                 | Availability Zone |
+--------------------------------------+--------+------+-------------+-----------+-----------+-----------------+----------------------+-------------------+
| 10f5a2a1-36f5-45aa-a8e6-00e94e592e88 | Share1 | 1    | NFS         | available | False     | my_type         | manila@paris#epsilon | nova              |
| 195e3ba2-9342-446a-bc93-a584551de0ac | Share2 | 1    | NFS         | available | True      | default         | manila@london#LONDON | nova              |
+--------------------------------------+--------+------+-------------+-----------+-----------+-----------------+----------------------+-------------------+
```

Check the share status and see the share export locations. After `creating` status share should have status `available`:

```
$ manila show Share2
+----------------------+--------------------------------------------------------------------------------+
| Property             | Value                                                                          |
+----------------------+--------------------------------------------------------------------------------+
| status               | available                                                                      |
| share_type_name      | default                                                                        |
| description          | My second share                                                                |
| availability_zone    | nova                                                                           |
| share_network_id     | c895fe26-92be-4152-9e6c-f2ad230efb13                                           |
| export_locations     |                                                                                |
|                      | path = 10.254.0.3:/shares/share-fe874928-39a2-441b-8d24-29e6f0fde965           |
|                      | preferred = False                                                              |
|                      | is_admin_only = False                                                          |
|                      | id = de6d4012-6158-46f0-8b28-4167baca51a7                                      |
|                      | share_instance_id = fe874928-39a2-441b-8d24-29e6f0fde965                       |
|                      | path = 10.0.0.3:/shares/share-fe874928-39a2-441b-8d24-29e6f0fde965             |
|                      | preferred = False                                                              |
|                      | is_admin_only = True                                                           |
|                      | id = 602d0f5c-921b-4e45-bfdb-5eec8a89165a                                      |
|                      | share_instance_id = fe874928-39a2-441b-8d24-29e6f0fde965                       |
```

```
| share_server_id      | 2e9d2d02-883f-47b5-bb98-e053b8d1e683                                 |
| host                 | manila@london#LONDON                                                 |
| access_rules_status  | active                                                               |
| snapshot_id          | None                                                                 |
| is_public            | True                                                                 |
| task_state           | None                                                                 |
| snapshot_support     | True                                                                 |
| id                   | 195e3ba2-9342-446a-bc93-a584551de0ac                                 |
| size                 | 1                                                                    |
| name                 | Share2                                                               |
| share_type           | bf6ada49-990a-47c3-88bc-c0cb31d5c9bf                                 |
| has_replicas         | False                                                                |
| replication_type     | None                                                                 |
| created_at           | 2016-03-25T12:13:40.000000                                           |
| share_proto          | NFS                                                                  |
| consistency_group_id | None                                                                 |
| project_id           | 907004508ef4447397ce6741a8f037c1                                     |
| metadata             | {u'aim': u'testing'}                                                 |
+----------------------+----------------------------------------------------------------------+
```

`is_public` defines the level of visibility for the share: whether other tenants can or cannot see the share. By default, the share is private.

Update share

Update the name, or description, or level of visibility for all tenants for the share if you need:

```
$ manila update Share2 --description "My second share. Updated" --is-public
False
```

```
$ manila show Share2
+----------------------+----------------------------------------------------------------------+
| Property             | Value                                                                |
+----------------------+----------------------------------------------------------------------+
| status               | available                                                            |
| share_type_name      | default                                                              |
| description          | My second share. Updated                                             |
| availability_zone    | nova                                                                 |
| share_network_id     | c895fe26-92be-4152-9e6c-f2ad230efb13                                 |
| export_locations     |                                                                      |
|                      | path = 10.254.0.3:/shares/share-fe874928-39a2-441b-8d24-29e6f0fde965 |
|                      | preferred = False                                                    |
|                      | is_admin_only = False                                                |
|                      | id = de6d4012-6158-46f0-8b28-4167baca51a7                            |
|                      | share_instance_id = fe874928-39a2-441b-8d24-29e6f0fde965             |
|                      | path = 10.0.0.3:/shares/share-fe874928-39a2-441b-8d24-29e6f0fde965   |
|                      | preferred = False                                                    |
|                      | is_admin_only = True                                                 |
|                      | id = 602d0f5c-921b-4e45-bfdb-5eec8a89165a                            |
|                      | share_instance_id = fe874928-39a2-441b-8d24-29e6f0fde965             |
| share_server_id      | 2e9d2d02-883f-47b5-bb98-e053b8d1e683                                 |
| host                 | manila@london#LONDON                                                 |
```

```
| access_rules_status  | active                                                                |
| snapshot_id          | None                                                                  |
| is_public            | False                                                                 |
| task_state           | None                                                                  |
| snapshot_support     | True                                                                  |
| id                   | 195e3ba2-9342-446a-bc93-a584551de0ac                                  |
| size                 | 1                                                                     |
| name                 | Share2                                                                |
| share_type           | bf6ada49-990a-47c3-88bc-c0cb31d5c9bf                                  |
| has_replicas         | False                                                                 |
| replication_type     | None                                                                  |
| created_at           | 2016-03-25T12:13:40.000000                                            |
| share_proto          | NFS                                                                   |
| consistency_group_id | None                                                                  |
| project_id           | 907004508ef4447397ce6741a8f037c1                                      |
| metadata             | {u'aim': u'testing'}                                                  |
+----------------------+-----------------------------------------------------------------------+
```

A share can have one of these status values:

Status	Description
creating	The share is being created.
deleting	The share is being deleted.
error	An error occurred during share creation.
error_deleting	An error occurred during share deletion.
available	The share is ready to use.
manage_starting	Share manage started.
manage_error	Share manage failed.
unmanage_starting	Share unmanage started.
unmanage_error	Share cannot be unmanaged.
unmanaged	Share was unmanaged.
extending	The extend, or increase, share size request was issued successfully.
extending_error	Extend share failed.
shrinking	Share is being shrunk.
shrinking_error	Failed to update quota on share shrinking.
shrinking_possible_data_loss_error	Shrink share failed due to possible data loss.

Status	Description
migrating	Share migration is in progress.

Share metadata

If you want to set the metadata key-value pairs on the share, run:

```
$ manila metadata Share2 set project=my_abc deadline=01/20/16
```

Get all metadata key-value pairs of the share:

```
$ manila metadata-show Share2
+----------+----------+
| Property | Value    |
+----------+----------+
| aim      | testing  |
| project  | my_abc   |
| deadline | 01/20/16 |
+----------+----------+
```

You can update the metadata:

```
$ manila metadata-update-all Share2 deadline=01/30/16
+----------+----------+
| Property | Value    |
+----------+----------+
| deadline | 01/30/16 |
+----------+----------+
```

You also can unset the metadata using **manila metadata <share_name> unset <metadata_key(s)>**.

Reset share state

As administrator, you can reset the state of a share.

Use **manila reset-state [-state <state>] <share>** command to reset share state, where `state` indicates which state to assign the share. Options include `available`, `error`, `creating`, `deleting`, `error_deleting` states.

```
$ manila reset-state Share2 --state deleting
```

```
$ manila show Share2
+---------------------+--------------------------------------------------------------------------+
| Property            | Value                                                                    |
+---------------------+--------------------------------------------------------------------------+
| status              | deleting                                                                 |
| share_type_name     | default                                                                  |
| description         | My second share. Updated                                                 |
```

```
| availability_zone    | nova                                                                    |
| share_network_id     | c895fe26-92be-4152-9e6c-f2ad230efb13                                    |
| export_locations     |                                                                         |
|                      | path = 10.254.0.3:/shares/share-fe874928-39a2-441b-8d24-29e6f0fde965 |
|                      | preferred = False                                                       |
|                      | is_admin_only = False                                                   |
|                      | id = de6d4012-6158-46f0-8b28-4167baca51a7                               |
|                      | share_instance_id = fe874928-39a2-441b-8d24-29e6f0fde965                |
|                      | path = 10.0.0.3:/shares/share-fe874928-39a2-441b-8d24-29e6f0fde965   |
|                      | preferred = False                                                       |
|                      | is_admin_only = True                                                    |
|                      | id = 602d0f5c-921b-4e45-bfdb-5eec8a89165a                               |
|                      | share_instance_id = fe874928-39a2-441b-8d24-29e6f0fde965                |
| share_server_id      | 2e9d2d02-883f-47b5-bb98-e053b8d1e683                                    |
| host                 | manila@london#LONDON                                                    |
| access_rules_status  | active                                                                  |
| snapshot_id          | None                                                                    |
| is_public            | False                                                                   |
| task_state           | None                                                                    |
| snapshot_support     | True                                                                    |
| id                   | 195e3ba2-9342-446a-bc93-a584551de0ac                                    |
| size                 | 1                                                                       |
| name                 | Share2                                                                  |
| share_type           | bf6ada49-990a-47c3-88bc-c0cb31d5c9bf                                    |
| has_replicas         | False                                                                   |
| replication_type     | None                                                                    |
| created_at           | 2016-03-25T12:13:40.000000                                              |
| share_proto          | NFS                                                                     |
| consistency_group_id | None                                                                    |
| project_id           | 907004508ef4447397ce6741a8f037c1                                        |
| metadata             | {u'deadline': u'01/30/16'}                                              |
+----------------------+-------------------------------------------------------------------------+
```

Delete and force-delete share

You also can force-delete a share. The shares cannot be deleted in transitional states. The transitional states are `creating`, `deleting`, `managing`, `unmanaging`, `migrating`, `extending`, and `shrinking` statuses for the shares. Force-deletion deletes an object in any state. Use the `policy.json` file to grant permissions for this action to other roles.

> **Tip**
>
> The configuration file `policy.json` may be used from different places. The path `/etc/manila/policy.json` is one of expected paths by default.

Use **manila delete <share_name_or_ID>** command to delete a specified share:

```
$ manila delete %share_name_or_id%
```

> **Note**
>
> If you specified *the consistency group* while creating a share, you should provide the *--consistency-group* parameter to delete the share:

```
$ manila delete %share_name_or_id% --consistency-group %consistency-group-id%
```

If you try to delete the share in one of the transitional state using soft-deletion you'll get an error:

```
$ manila delete Share2
Delete for share 195e3ba2-9342-446a-bc93-a584551de0ac failed: Invalid share:
Share status must be one of ('available', 'error', 'inactive'). (HTTP 403)
(Request-ID: req-9a77b9a0-17d2-4d97-8a7a-b7e23c27f1fe)
ERROR: Unable to delete any of the specified shares.
```

A share cannot be deleted in a transitional status, that it why an error from `python-manilaclient` appeared.

Print the list of all shares for all tenants:

```
$ manila list --all-tenants
+--------------------------------------+--------+------+-------------+-----------+-----------+-----------------+----------------------+-------------------+
| ID                                   | Name   | Size | Share Proto | Status    | Is Public | Share Type Name | Host                 | Availability Zone |
+--------------------------------------+--------+------+-------------+-----------+-----------+-----------------+----------------------+-------------------+
| 10f5a2a1-36f5-45aa-a8e6-00e94e592e88 | Share1 | 1    | NFS         | available | False     | my_type         | manila@paris#epsilon | nova              |
| 195e3ba2-9342-446a-bc93-a584551de0ac | Share2 | 1    | NFS         | available | False     | default         | manila@london#LONDON | nova              |
+--------------------------------------+--------+------+-------------+-----------+-----------+-----------------+----------------------+-------------------+
```

Force-delete Share2 and check that it is absent in the list of shares, run:

```
$ manila force-delete Share2
```

```
$ manila list
+--------------------------------------+--------+------+-------------+-----------+-----------+-----------------+----------------------+-------------------+
| ID                                   | Name   | Size | Share Proto | Status    | Is Public | Share Type Name | Host                 | Availability Zone |
+--------------------------------------+--------+------+-------------+-----------+-----------+-----------------+----------------------+-------------------+
| 10f5a2a1-36f5-45aa-a8e6-00e94e592e88 | Share1 | 1    | NFS         | available | False     | my_type         | manila@paris#epsilon | nova              |
+--------------------------------------+--------+------+-------------+-----------+-----------+-----------------+----------------------+-------------------+
```

Manage access to share

The Shared File Systems service allows to grant or deny access to a specified share, and list the permissions for a specified share.

To grant or deny access to a share, specify one of these supported share access levels:

- **rw**. Read and write (RW) access. This is the default value.
- **ro**. Read-only (RO) access.

You must also specify one of these supported authentication methods:

- **ip**. Authenticates an instance through its IP address. A valid format is `XX.XX.XX.XX`

or `XX.XX.XX.XX/XX`. For example `0.0.0.0/0`.

- **user**. Authenticates by a specified user or group name. A valid value is an alphanumeric string that can contain some special characters and is from 4 to 32 characters long.
- **cert**. Authenticates an instance through a TLS certificate. Specify the TLS identity as the IDENTKEY. A valid value is any string up to 64 characters long in the common name (CN) of the certificate. The meaning of a string depends on its interpretation.
- **cephx**. Ceph authentication system. Specify the Ceph auth ID that needs to be authenticated and authorized for share access by the Ceph back end. A valid value must be non-empty, consist of ASCII printable characters, and not contain periods.

Try to mount NFS share with export path `10.0.0.4:/shares/manila_share_a5fb1ab7_0bbd_465b_ac14_05706294b6e9` on the node with IP address `10.0.0.13`:

```
$ sudo mount -v -t nfs
10.0.0.4:/shares/manila_share_a5fb1ab7_0bbd_465b_ac14_05706294b6e9 /mnt/
mount.nfs: timeout set for Tue Oct  6 10:37:23 2015
mount.nfs: trying text-based options 'vers=4,addr=10.0.0.4,clientaddr=10.0.0.13'
mount.nfs: mount(2): Permission denied
mount.nfs: access denied by server while mounting
10.0.0.4:/shares/manila_share_a5fb1ab7_0bbd_465b_ac14_05706294b6e9
```

An error message “Permission denied” appeared, so you are not allowed to mount a share without an access rule. Allow access to the share with `ip` access type and `10.0.0.13` IP address:

```
$ manila access-allow Share1 ip 10.0.0.13 --access-level rw
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| share_id     | 10f5a2a1-36f5-45aa-a8e6-00e94e592e88 |
| access_type  | ip                                   |
| access_to    | 10.0.0.13                            |
| access_level | rw                                   |
| state        | new                                  |
| id           | de715226-da00-4cfc-b1ab-c11f3393745e |
+--------------+--------------------------------------+
```

Try to mount a share again. This time it is mounted successfully:

```
$ sudo mount -v -t nfs
10.0.0.4:/shares/manila_share_a5fb1ab7_0bbd_465b_ac14_05706294b6e9 /mnt/
```

Since it is allowed node on 10.0.0.13 read and write access, try to create a file on a mounted share:

```
$ cd /mnt
$ ls
lost+found
$ touch my_file.txt
```

Connect via SSH to the `10.0.0.4` node and check new file my_file.txt in the `/shares/manila_share_a5fb1ab7_0bbd_465b_ac14_05706294b6e9` directory:

```
$ ssh 10.0.0.4
$ cd /shares
$ ls
manila_share_a5fb1ab7_0bbd_465b_ac14_05706294b6e9
$ cd manila_share_a5fb1ab7_0bbd_465b_ac14_05706294b6e9
$ ls
lost+found  my_file.txt
```

You have successfully created a file from instance that was given access by its IP address.

Allow access to the share with `user` access type:

```
$ manila access-allow Share1 user demo --access-level rw
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| share_id     | 10f5a2a1-36f5-45aa-a8e6-00e94e592e88 |
| access_type  | user                                 |
| access_to    | demo                                 |
| access_level | rw                                   |
| state        | new                                  |
| id           | 4f391c6b-fb4f-47f5-8b4b-88c5ec9d568a |
+--------------+--------------------------------------+
```

> **Note**
>
> Different share features are supported by different share drivers. For the example, the Generic driver with the Block Storage service as a back-end doesn't support `user` and `cert` authentications methods. For details of supporting of features by different drivers, see Manila share features support mapping.

To verify that the access rules (ACL) were configured correctly for a share, you list permissions for a share:

```
$ manila access-list Share1
+--------------------------------------+-------------+------------+--------------+--------+
| id                                   | access type | access to  | access level | state  |
+--------------------------------------+-------------+------------+--------------+--------+
| 4f391c6b-fb4f-47f5-8b4b-88c5ec9d568a | user        | demo       | rw           | error  |
| de715226-da00-4cfc-b1ab-c11f3393745e | ip          | 10.0.0.13  | rw           | active |
+--------------------------------------+-------------+------------+--------------+--------+
```

Deny access to the share and check that deleted access rule is absent in the access rule list:

```
$ manila access-deny Share1 de715226-da00-4cfc-b1ab-c11f3393745e

$ manila access-list Share1
+--------------------------------------+-------------+-----------+--------------+-------+
| id                                   | access type | access to | access level | state |
+--------------------------------------+-------------+-----------+--------------+-------+
| 4f391c6b-fb4f-47f5-8b4b-88c5ec9d568a | user        | demo      | rw           | error |
+--------------------------------------+-------------+-----------+--------------+-------+
```

Manage and unmanage share

To `manage` a share means that an administrator, rather than a share driver, manages the storage lifecycle. This approach is appropriate when an administrator already has the custom non-manila share with its size, shared file system protocol, and export path, and an administrator wants to register it in the Shared File System service.

To `unmanage` a share means to unregister a specified share from the Shared File Systems service. Administrators can revert an unmanaged share to managed status if needed.

Unmanage a share

The `unmanage` operation is not supported for shares that were created on top of share servers and created with share networks. The Share service should have the option `driver_handles_share_servers = False` set in the `manila.conf` file. You can unmanage a share that has no dependent snapshots.

To unmanage managed share, run the **manila unmanage <share>** command. Then try to print the information about the share. The returned result should indicate that Shared File Systems service won’t find the share:

```
$ manila unmanage share_for_docs
$ manila show share_for_docs
ERROR: No share with a name or ID of 'share_for_docs' exists.
```

Manage a share

To register the non-managed share in the File System service, run the **manila manage** command:

```
manila manage [--name <name>] [--description <description>]
              [--share_type <share-type>]
              [--driver_options [<key=value> [<key=value> ...]]]
              <service_host> <protocol> <export_path>
```

The positional arguments are:

- service_host. The manage-share service host in `host@backend#POOL` format, which consists of the host name for the back end, the name of the back end, and the pool name for the back end.
- protocol. The Shared File Systems protocol of the share to manage. Valid values are NFS, CIFS, GlusterFS, or HDFS.
- export_path. The share export path in the format appropriate for the protocol:
  - NFS protocol. 10.0.0.1:/foo_path.
  - CIFS protocol. \\10.0.0.1\foo_name_of_cifs_share.
  - HDFS protocol. hdfs://10.0.0.1:foo_port/foo_share_name.
  - GlusterFS. 10.0.0.1:/foo_volume.

The `driver_options` is an optional set of one or more key and value pairs that describe driver options. Note that the share type must have the `driver_handles_share_servers = False` option. As a result, a special share type named `for_managing` was used in example.

To manage share, run:

```
$ manila manage \
     manila@paris#shares \
    nfs \
    1.0.0.4:/shares/manila_share_6d2142d8_2b9b_4405_867f_8a48094c893f \
    --name share_for_docs \
    --description "We manage share." \
    --share_type for_managing
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| status                      | manage_starting                      |
| share_type_name             | for_managing                         |
| description                 | We manage share.                     |
| availability_zone           | None                                 |
| share_network_id            | None                                 |
| share_server_id             | None                                 |
| host                        | manila@paris#shares                  |
| access_rules_status         | active                               |
| snapshot_id                 | None                                 |
| is_public                   | False                                |
```

```
| task_state                  | None                                 |
| snapshot_support            | True                                 |
| id                          | ddfb1240-ed5e-4071-a031-b842035a834a |
| size                        | None                                 |
| name                        | share_for_docs                       |
| share_type                  | 14ee8575-aac2-44af-8392-d9c9d344f392 |
| has_replicas                | False                                |
| replication_type            | None                                 |
| created_at                  | 2016-03-25T15:22:43.000000           |
| share_proto                 | NFS                                  |
| consistency_group_id        | None                                 |
| source_cgsnapshot_member_id | None                                 |
| project_id                  | 907004508ef4447397ce6741a8f037c1     |
| metadata                    | {}                                   |
+-----------------------------+--------------------------------------+
```

Check that the share is available:

```
$ manila show share_for_docs
+---------------------+------------------------------------------------------------------------+
| Property            | Value                                                                  |
+---------------------+------------------------------------------------------------------------+
| status              | available                                                              |
| share_type_name     | for_managing                                                           |
| description         | We manage share.                                                       |
| availability_zone   | None                                                                   |
| share_network_id    | None                                                                   |
| export_locations    |                                                                        |
|                     | path = 1.0.0.4:/shares/manila_share_6d2142d8_2b9b_4405_867f_8a48094c893f |
|                     | preferred = False                                                      |
|                     | is_admin_only = False                                                  |
|                     | id = d4d048bf-4159-4a94-8027-e567192b8d30                              |
|                     | share_instance_id = 4c8e3887-4f9a-4775-bab4-e5840a09c34e               |
|                     | path = 2.0.0.3:/shares/manila_share_6d2142d8_2b9b_4405_867f_8a48094c893f |
|                     | preferred = False                                                      |
|                     | is_admin_only = True                                                   |
|                     | id = 1dd4f0a3-778d-486a-a851-b522f6e7cf5f                              |
|                     | share_instance_id = 4c8e3887-4f9a-4775-bab4-e5840a09c34e               |
| share_server_id     | None                                                                   |
| host                | manila@paris#shares                                                    |
| access_rules_status | active                                                                 |
| snapshot_id         | None                                                                   |
| is_public           | False                                                                  |
| task_state          | None                                                                   |
| snapshot_support    | True                                                                   |
| id                  | ddfb1240-ed5e-4071-a031-b842035a834a                                   |
| size                | 1                                                                      |
| name                | share_for_docs                                                         |
| share_type          | 14ee8575-aac2-44af-8392-d9c9d344f392                                   |
| has_replicas        | False                                                                  |
| replication_type    | None                                                                   |
| created_at          | 2016-03-25T15:22:43.000000                                             |
| share_proto         | NFS                                                                    |
| consistency_group_id | None                                                                  |
| project_id          | 907004508ef4447397ce6741a8f037c1                                       |
| metadata            | {}                                                                     |
+---------------------+------------------------------------------------------------------------+
```

Resize share

To change file share size, use the **manila extend** command and the **manila shrink** command. For most drivers it is safe operation. If you want to be sure that your data is safe, you can make a share back up by creating a snapshot of it.

You can extend and shrink the share with the **manila extend** and **manila shrink** commands respectively, and specify the share with the new size that does not exceed the quota. For details, see *Quotas and Limits*. You also cannot shrink share size to 0 or to a greater value than the current share size.

While extending, the share has an `extending` status. This means that the increase share size request was issued successfully.

To extend the share and check the result, run:

```
$ manila extend docs_resize 2
$ manila show docs_resize
```

```
+---------------------+--------------------------------------------------------------------------+
| Property            | Value                                                                    |
+---------------------+--------------------------------------------------------------------------+
| status              | available                                                                |
| share_type_name     | my_type                                                                  |
| description         | None                                                                     |
| availability_zone   | nova                                                                     |
| share_network_id    | None                                                                     |
| export_locations    |                                                                          |
|                     | path = 1.0.0.4:/shares/manila_share_b8afc508_8487_442b_b170_ea65b07074a8 |
|                     | preferred = False                                                        |
|                     | is_admin_only = False                                                    |
|                     | id = 3ffb76f4-92b9-4639-83fd-025bc3e302ff                                |
|                     | share_instance_id = b8afc508-8487-442b-b170-ea65b07074a8                 |
|                     | path = 2.0.0.3:/shares/manila_share_b8afc508_8487_442b_b170_ea65b07074a8 |
|                     | preferred = False                                                        |
|                     | is_admin_only = True                                                     |
|                     | id = 1f0e263f-370d-47d3-95f6-1be64088b9da                                |
|                     | share_instance_id = b8afc508-8487-442b-b170-ea65b07074a8                 |
| share_server_id     | None                                                                     |
| host                | manila@paris#shares                                                      |
| access_rules_status | active                                                                   |
| snapshot_id         | None                                                                     |
| is_public           | False                                                                    |
| task_state          | None                                                                     |
| snapshot_support    | True                                                                     |
| id                  | b07dbebe-a328-403c-b402-c8871c89e3d1                                     |
| size                | 2                                                                        |
| name                | docs_resize                                                              |
| share_type          | 14ee8575-aac2-44af-8392-d9c9d344f392                                     |
| has_replicas        | False                                                                    |
| replication_type    | None                                                                     |
| created_at          | 2016-03-25T15:33:18.000000                                               |
| share_proto         | NFS                                                                      |
| consistency_group_id | None                                                                    |
| project_id          | 907004508ef4447397ce6741a8f037c1                                         |
| metadata            | {}                                                                       |
+---------------------+--------------------------------------------------------------------------+
```

While shrinking, the share has a `shrinking` status. This means that the decrease share size request was issued successfully. To shrink the share and check the result, run:

```
$ manila shrink docs_resize 1
$ manila show docs_resize
```

```
+---------------------+--------------------------------------------------------------------------+
| Property            | Value                                                                    |
+---------------------+--------------------------------------------------------------------------+
| status              | available                                                                |
| share_type_name     | my_type                                                                  |
| description         | None                                                                     |
| availability_zone   | nova                                                                     |
| share_network_id    | None                                                                     |
| export_locations    |                                                                          |
|                     | path = 1.0.0.4:/shares/manila_share_b8afc508_8487_442b_b170_ea65b07074a8 |
|                     | preferred = False                                                        |
|                     | is_admin_only = False                                                    |
|                     | id = 3ffb76f4-92b9-4639-83fd-025bc3e302ff                                |
|                     | share_instance_id = b8afc508-8487-442b-b170-ea65b07074a8                 |
|                     | path = 2.0.0.3:/shares/manila_share_b8afc508_8487_442b_b170_ea65b07074a8 |
|                     | preferred = False                                                        |
|                     | is_admin_only = True                                                     |
|                     | id = 1f0e263f-370d-47d3-95f6-1be64088b9da                                |
|                     | share_instance_id = b8afc508-8487-442b-b170-ea65b07074a8                 |
| share_server_id     | None                                                                     |
| host                | manila@paris#shares                                                      |
| access_rules_status | active                                                                   |
| snapshot_id         | None                                                                     |
| is_public           | False                                                                    |
| task_state          | None                                                                     |
| snapshot_support    | True                                                                     |
| id                  | b07dbebe-a328-403c-b402-c8871c89e3d1                                     |
| size                | 1                                                                        |
| name                | docs_resize                                                              |
| share_type          | 14ee8575-aac2-44af-8392-d9c9d344f392                                     |
| has_replicas        | False                                                                    |
| replication_type    | None                                                                     |
| created_at          | 2016-03-25T15:33:18.000000                                               |
| share_proto         | NFS                                                                      |
| consistency_group_id | None                                                                    |
| project_id          | 907004508ef4447397ce6741a8f037c1                                         |
| metadata            | {}                                                                       |
+---------------------+--------------------------------------------------------------------------+
```

Quotas and limits

Limits

Limits are the resource limitations that are allowed for each tenant (project). An administrator can configure limits in the `manila.conf` file.

Users can query their rate and absolute limits.

To see the absolute limits, run:

```
$ manila absolute-limits
```

```
+----------------------------+-------+
| Name                       | Value |
+----------------------------+-------+
| maxTotalShareGigabytes     | 1000  |
| maxTotalShareNetworks      | 10    |
| maxTotalShareSnapshots     | 50    |
| maxTotalShares             | 50    |
| maxTotalSnapshotGigabytes  | 1000  |
| totalShareGigabytesUsed    | 1     |
| totalShareNetworksUsed     | 2     |
| totalShareSnapshotsUsed    | 1     |
| totalSharesUsed            | 1     |
| totalSnapshotGigabytesUsed | 1     |
+----------------------------+-------+
```

Rate limits control the frequency at which users can issue specific API requests. Administrators use rate limiting to configure limits on the type and number of API calls that can be made in a specific time interval. For example, a rate limit can control the number of `GET` requests processed during a one-minute period.

To set the API rate limits, modify the `etc/manila/api-paste.ini` file, which is a part of the WSGI pipeline and defines the actual limits. You need to restart `manila-api` service after you edit the `etc/manila/api-paste.ini` file.

```
[filter:ratelimit]
paste.filter_factory = manila.api.v1.limits:RateLimitingMiddleware.factory
limits = (POST, "*/shares", ^/shares, 120, MINUTE);(PUT, "*/shares", .*, 120,
MINUTE);(DELETE, "*", .*, 120, MINUTE)
```

Also, add the `ratelimit` to `noauth`, `keystone`, `keystone_nolimit` parameters in the `[composite:openstack_share_api]` and `[composite:openstack_share_api_v2]` groups.

```
[composite:openstack_share_api]
use = call:manila.api.middleware.auth:pipeline_factory
noauth = cors faultwrap ssl ratelimit sizelimit noauth api
keystone = cors faultwrap ssl ratelimit sizelimit authtoken keystonecontext api
keystone_nolimit = cors faultwrap ssl ratelimit sizelimit authtoken
keystonecontext api

[composite:openstack_share_api_v2]
use = call:manila.api.middleware.auth:pipeline_factory
noauth = cors faultwrap ssl ratelimit sizelimit noauth apiv2
keystone = cors faultwrap ssl ratelimit sizelimit authtoken keystonecontext
apiv2
keystone_nolimit = cors faultwrap ssl ratelimit sizelimit authtoken
keystonecontext apiv2
```

To see the rate limits, run:

```
$ manila rate-limits
+--------+------------+-------+--------+--------+----------------------+
| Verb   | URI        | Value | Remain | Unit   | Next_Available       |
+--------+------------+-------+--------+--------+----------------------+
| DELETE | "*"        | 120   | 120    | MINUTE | 2015-10-20T15:17:20Z |
| POST   | "*/shares" | 120   | 120    | MINUTE | 2015-10-20T15:17:20Z |
| PUT    | "*/shares" | 120   | 120    | MINUTE | 2015-10-20T15:17:20Z |
+--------+------------+-------+--------+--------+----------------------+
```

Quotas

Quota sets provide quota management support.

To list the quotas for a tenant or user, use the **manila quota-show** command. If you specify the optional *--user* parameter, you get the quotas for this user in the specified tenant. If you omit this parameter, you get the quotas for the specified project.

> **Note**
>
> The Shared File Systems service does not perform mapping of usernames and tenant/project names to IDs. Provide only ID values to get correct setup of quotas. Setting it by names you set quota for nonexistent tenant/user. In case quota is not set explicitly by tenant/user ID, The Shared File Systems service just applies default quotas.

```
$ manila quota-show --tenant %tenant_id% --user %user_id%
+--------------------+-------+
| Property           | Value |
+--------------------+-------+
| gigabytes          | 1000  |
| snapshot_gigabytes | 1000  |
| snapshots          | 50    |
| shares             | 50    |
| share_networks     | 10    |
+--------------------+-------+
```

There are default quotas for a project that are set from the `manila.conf` file. To list the default quotas for a project, use the **manila quota-defaults** command:

```
$ manila quota-defaults --tenant %tenant_id%
+--------------------+-------+
| Property           | Value |
+--------------------+-------+
| gigabytes          | 1000  |
| snapshot_gigabytes | 1000  |
| snapshots          | 50    |
| shares             | 50    |
| share_networks     | 10    |
+--------------------+-------+
```

The administrator can update the quotas for a specific tenant, or for a specific user by providing both the `--tenant` and `--user` optional arguments. It is possible to update the `shares`, `snapshots`, `gigabytes`, `snapshot-gigabytes`, and `share-networks` quotas.

```
$ manila quota-update %tenant_id% --user %user_id% --shares 49 --snapshots 49
```

As administrator, you can also permit or deny the force-update of a quota that is already used, or if the requested value exceeds the configured quota limit. To force-update a quota, use `force` optional key.

```
$ manila quota-update %tenant_id% --shares 51 --snapshots 51 --force
```

To revert quotas to default for a project or for a user, delete quotas:

```
$ manila quota-delete --tenant %tenant_id% --user %user_id%
```

Migrate shares

As an administrator, you can migrate a share with its data from one location to another in a manner that is transparent to users and workloads. You can use `manila` client commands to complete a share migration.

Possible use cases for data migration include:

- Bring down a physical storage device for maintenance without disrupting workloads.
- Modify the properties of a share.
- Free up space in a thinly-provisioned back end.

Migrate a share with the **manila migrate** command, as shown in the following example:

```
$ manila migrate shareID destinationHost --force-host-copy True|False
```

In this example, *--force-host-copy True* forces the generic host-based migration mechanism and bypasses any driver optimizations. `destinationHost` is in this format `host#pool` which includes destination host and pool.

> **Note**
>
> If the user is not an administrator, the migration fails.

Share types

A share type enables you to filter or choose back ends before you create a share and to set data for the share driver. A share type behaves in the same way as a Block Storage volume type behaves.

In the Shared File Systems configuration file `manila.conf`, the administrator can set the share type used by default for the share creation and then create a default share type.

To create a share type, use **manila type-create** command as:

```
manila type-create [--snapshot_support <snapshot_support>]
                   [--is_public <is_public>]
                   <name> <spec_driver_handles_share_servers>
```

where the `name` is the share type name, `--is_public` defines the level of the visibility for the share type, `snapshot_support` and `spec_driver_handles_share_servers` are the extra specifications used to filter back ends. Administrators can create share types with these extra specifications for the back ends filtering:

- `driver_handles_share_servers`. Required. Defines the driver mode for share server lifecycle management. Valid values are `true`/`1` and `false`/`0`. Set to True when the share driver can manage, or handle, the share server lifecycle. Set to False when an administrator, rather than a share driver, manages the bare metal storage with some net interface instead of the presence of the share servers.
- `snapshot_support`. Filters back ends by whether they do or do not support share snapshots. Default is `True`. Set to True to find back ends that support share snapshots. Set to False to find back ends that do not support share snapshots.

> **Note**
>
> The extra specifications set in the share types are operated in the *Scheduling*.

Administrators can also set additional extra specifications for a share type for the following purposes:

- *Filter back ends*. Unqualified extra specifications written in this format: `extra_spec=value`. For example, **netapp_raid_type=raid4**.
- *Set data for the driver*. Qualified extra specifications always written with the prefix with a colon, except for the special `capabilities` prefix, in this format: `vendor:extra_spec=value`. For example, **netapp:thin_provisioned=true**.

The scheduler uses the special capabilities prefix for filtering. The scheduler can only

create a share on a back end that reports capabilities matching the un-scoped extra-spec keys for the share type. For details, see Capabilities and Extra-Specs.

Each driver implementation determines which extra specification keys it uses. For details, see the documentation for the driver.

An administrator can use the `policy.json` file to grant permissions for share type creation with extra specifications to other roles.

You set a share type to private or public and *manage the access* to the private share types. By default a share type is created as publicly accessible. Set *--is_public* to `False` to make the share type private.

Share type operations

To create a new share type you need to specify the name of the new share type. You also require an extra spec `driver_handles_share_servers`. The new share type can also be public.

```
$ manila type-create netapp1 False --is_public True

$ manila type-list
+-----+--------+-----------+-----------+----------------------------------+----------------------+
| ID  | Name   | Visibility| is_default| required_extra_specs              | optional_extra_specs |
+-----+--------+-----------+-----------+----------------------------------+----------------------+
| c0..| netapp1| public    | No        | driver_handles_share_servers:False| snapshot_support:True |
+-----+--------+-----------+-----------+----------------------------------+----------------------+
```

You can set or unset extra specifications for a share type using **manila type-key <share_type> set <key=value>** command. Since it is up to each driver what extra specification keys it uses, see the documentation for the specified driver.

```
$ manila type-key netapp1 set thin_provisioned=True
```

It is also possible to view a list of current share types and extra specifications:

```
$ manila extra-specs-list
+-------------+---------+-------------------------------------+
| ID          | Name    | all_extra_specs                     |
+-------------+---------+-------------------------------------+
| c0086582-...| netapp1 | snapshot_support : True             |
|             |         | thin_provisioned : True             |
|             |         | driver_handles_share_servers : True |
+-------------+---------+-------------------------------------+
```

Use **manila type-key <share_type> unset <key>** to unset an extra specification.

The public or private share type can be deleted with the **manila type-delete <share_type>** command.

Share type access

You can manage access to a private share type for different projects. Administrators can provide access, remove access, and retrieve information about access for a specified private share.

Create a private type:

```
$ manila type-create my_type1 True --is_public False
+-----+---------+-----------+-----------+---------------------------------+----------------------+
| ID  | Name    | Visibility| is_default| required_extra_specs            | optional_extra_specs |
+-----+---------+-----------+-----------+---------------------------------+----------------------+
| a4..| my_type1| private   | -         | driver_handles_share_servers:True| snapshot_support:True|
+-----+---------+-----------+-----------+---------------------------------+----------------------+
```

> **Note**
>
> If you run **manila type-list** only public share types appear. To see private share types, run **manila type-list** with *--all* optional argument.

Grant access to created private type for a demo and alt_demo projects by providing their IDs:

```
$ manila type-access-add my_type1 d8f9af6915404114ae4f30668a4f5ba7
$ manila type-access-add my_type1 e4970f57f1824faab2701db61ee7efdf
```

To view information about access for a private share, type `my_type1`:

```
$ manila type-access-list my_type1
+----------------------------------+
| Project_ID                       |
+----------------------------------+
| d8f9af6915404114ae4f30668a4f5ba7 |
| e4970f57f1824faab2701db61ee7efdf |
+----------------------------------+
```

After granting access to the share, the target project can see the share type in the list, and create private shares.

To deny access for a specified project, use **manila type-access-remove <share_type> <project_id>** command.

Share snapshots

The Shared File Systems service provides a snapshot mechanism to help users restore

data by running the **manila snapshot-create** command.

To export a snapshot, create a share from it, then mount the new share to an instance. Copy files from the attached share into the archive.

To import a snapshot, create a new share with appropriate size, attach it to instance, and then copy a file from the archive to the attached file system.

> **Note**
>
> You cannot delete a share while it has saved dependent snapshots.

Create a snapshot from the share:

```
$ manila snapshot-create Share1 --name Snapshot1 --description "Snapshot of
Share1"
+-------------+--------------------------------------+
| Property    | Value                                |
+-------------+--------------------------------------+
| status      | creating                             |
| share_id    | aca648eb-8c03-4394-a5cc-755066b7eb66 |
| name        | Snapshot1                            |
| created_at  | 2015-09-25T05:27:38.862040           |
| share_proto | NFS                                  |
| id          | 962e8126-35c3-47bb-8c00-f0ee37f42ddd |
| size        | 1                                    |
| share_size  | 1                                    |
| description | Snapshot of Share1                   |
+-------------+--------------------------------------+
```

Update snapshot name or description if needed:

```
$ manila snapshot-rename Snapshot1 Snapshot_1 --description "Snapshot of Share1.
Updated."
```

Check that status of a snapshot is `available`:

```
$ manila snapshot-show Snapshot1
+-------------+--------------------------------------+
| Property    | Value                                |
+-------------+--------------------------------------+
| status      | available                            |
| share_id    | aca648eb-8c03-4394-a5cc-755066b7eb66 |
| name        | Snapshot1                            |
| created_at  | 2015-09-25T05:27:38.000000           |
| share_proto | NFS                                  |
| id          | 962e8126-35c3-47bb-8c00-f0ee37f42ddd |
| size        | 1                                    |
| share_size  | 1                                    |
| description | Snapshot of Share1                   |
+-------------+--------------------------------------+
```

To restore your data from a snapshot, use **manila create** with key ***--snapshot-id***. This creates a new share from an existing snapshot. Create a share from a snapshot and check whether it is available:

```
$ manila create nfs 1 --name Share2 --metadata source=snapshot --description
"Share from a snapshot." --snapshot-id 962e8126-35c3-47bb-8c00-f0ee37f42ddd
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| status                      | None                                 |
| share_type_name             | default                              |
| description                 | Share from a snapshot.               |
| availability_zone           | None                                 |
| share_network_id            | None                                 |
| export_locations            | []                                   |
| share_server_id             | None                                 |
| host                        | None                                 |
| snapshot_id                 | 962e8126-35c3-47bb-8c00-f0ee37f42ddd |
| is_public                   | False                                |
| task_state                  | None                                 |
| snapshot_support            | True                                 |
| id                          | b6b0617c-ea51-4450-848e-e7cff69238c7 |
| size                        | 1                                    |
| name                        | Share2                               |
| share_type                  | c0086582-30a6-4060-b096-a42ec9d66b86 |
| created_at                  | 2015-09-25T06:25:50.240417           |
| export_location             | None                                 |
| share_proto                 | NFS                                  |
| consistency_group_id        | None                                 |
| source_cgsnapshot_member_id | None                                 |
| project_id                  | 20787a7ba11946adad976463b57d8a2f     |
| metadata                    | {u'source': u'snapshot'}             |
+-----------------------------+--------------------------------------+

$ manila show Share2
+-----------------------------+-----------------------------------------+
| Property                    | Value                                   |
+-----------------------------+-----------------------------------------+
| status                      | available                               |
| share_type_name             | default                                 |
| description                 | Share from a snapshot.                  |
| availability_zone           | nova                                    |
| share_network_id            | 5c3cbabb-f4da-465f-bc7f-fadbe047b85a    |
| export_locations            | 10.254.0.3:/shares/share-1dc2a471-3d47-...|
| share_server_id             | 41b7829d-7f6b-4c96-aea5-d106c2959961    |
| host                        | manila@generic1#GENERIC1                |
| snapshot_id                 | 962e8126-35c3-47bb-8c00-f0ee37f42ddd    |
| is_public                   | False                                   |
| task_state                  | None                                    |
| snapshot_support            | True                                    |
| id                          | b6b0617c-ea51-4450-848e-e7cff69238c7    |
| size                        | 1                                       |
| name                        | Share2                                  |
| share_type                  | c0086582-30a6-4060-b096-a42ec9d66b86    |
```

```
| created_at                  | 2015-09-25T06:25:50.000000                |
| share_proto                 | NFS                                       |
| consistency_group_id        | None                                      |
| source_cgsnapshot_member_id | None                                      |
| project_id                  | 20787a7ba11946adad976463b57d8a2f          |
| metadata                    | {u'source': u'snapshot'}                  |
+-----------------------------+-------------------------------------------+
```

You can soft-delete a snapshot using **manila snapshot-delete <snapshot_name_or_ID>**. If a snapshot is in busy state, and during the delete an `error_deleting` status appeared, administrator can force-delete it or explicitly reset the state.

Use **snapshot-reset-state [–state <state>] <snapshot>** to update the state of a snapshot explicitly. A valid value of a status are `available`, `error`, `creating`, `deleting`, `error_deleting`. If no state is provided, the `available` state will be used.

Use **manila snapshot-force-delete <snapshot>** to force-delete a specified share snapshot in any state.

Security services

A security service stores client configuration information used for authentication and authorization (AuthN/AuthZ). For example, a share server will be the client for an existing service such as LDAP, Kerberos, or Microsoft Active Directory.

You can associate a share with one to three security service types:

- `ldap`: LDAP.
- `kerberos`: Kerberos.
- `active_directory`: Microsoft Active Directory.

You can configure a security service with these options:

- A DNS IP address.
- An IP address or host name.
- A domain.
- A user or group name.
- The password for the user, if you specify a user name.

You can add the security service to the *share network*.

To create a security service, specify the security service type, a description of a security service, DNS IP address used inside tenant’s network, security service IP address or host name, domain, security service user or group used by tenant, and a password for the user. The share name is optional.

Create a `ldap` security service:

```
$ manila security-service-create ldap --dns-ip 8.8.8.8 --server 10.254.0.3
--name my_ldap_security_service
+-------------+--------------------------------------+
| Property    | Value                                |
+-------------+--------------------------------------+
| status      | new                                  |
| domain      | None                                 |
| password    | None                                 |
| name        | my_ldap_security_service             |
| dns_ip      | 8.8.8.8                              |
| created_at  | 2015-09-25T10:19:06.019527           |
| updated_at  | None                                 |
| server      | 10.254.0.3                           |
| user        | None                                 |
| project_id  | 20787a7ba11946adad976463b57d8a2f     |
| type        | ldap                                 |
| id          | 413479b2-0d20-4c58-a9d3-b129fa592d8e |
| description | None                                 |
+-------------+--------------------------------------+
```

To create `kerberos` security service, run:

```
$ manila security-service-create kerberos --server 10.254.0.3 --user demo
--password secret --name my_kerberos_security_service --description "Kerberos
security service"
+-------------+--------------------------------------+
| Property    | Value                                |
+-------------+--------------------------------------+
| status      | new                                  |
| domain      | None                                 |
| password    | secret                               |
| name        | my_kerberos_security_service         |
| dns_ip      | None                                 |
| created_at  | 2015-09-25T10:26:03.211849           |
| updated_at  | None                                 |
| server      | 10.254.0.3                           |
| user        | demo                                 |
| project_id  | 20787a7ba11946adad976463b57d8a2f     |
| type        | kerberos                             |
| id          | 7f46a447-2534-453d-924d-bd7c8e63bbec |
| description | Kerberos security service            |
+-------------+--------------------------------------+
```

To see the list of created security service use **manila security-service-list**:

```
$ manila security-service-list
+--------------------------------------+------------------------------+--------+----------+
| id                                   | name                         | status | type     |
+--------------------------------------+------------------------------+--------+----------+
| 413479b2-0d20-4c58-a9d3-b129fa592d8e | my_ldap_security_service     | new    | ldap     |
```

```
| 7f46a447-2534-453d-924d-bd7c8e63bbec | my_kerberos_security_service | new    | kerberos |
+--------------------------------------+------------------------------+--------+----------+
```

You can add a security service to the existing *share network*, which is not yet used (a `share network` not associated with a share).

Add a security service to the share network with `share-network-security-service-add` specifying share network and security service. The command returns information about the security service. You can see view new attributes and `share_networks` using the associated share network ID.

```
$ manila share-network-security-service-add share_net2 my_ldap_security_service
```

```
$ manila security-service-show my_ldap_security_service
+---------------+-----------------------------------------+
| Property      | Value                                   |
+---------------+-----------------------------------------+
| status        | new                                     |
| domain        | None                                    |
| password      | None                                    |
| name          | my_ldap_security_service                |
| dns_ip        | 8.8.8.8                                 |
| created_at    | 2015-09-25T10:19:06.000000              |
| updated_at    | None                                    |
| server        | 10.254.0.3                              |
| share_networks| [u'6d36c41f-d310-4aff-a0c2-ffd870e91cab'] |
| user          | None                                    |
| project_id    | 20787a7ba11946adad976463b57d8a2f        |
| type          | ldap                                    |
| id            | 413479b2-0d20-4c58-a9d3-b129fa592d8e    |
| description   | None                                    |
+---------------+-----------------------------------------+
```

It is possible to see the list of security services associated with a given share network. List security services for `share_net2` share network with:

```
$ manila share-network-security-service-list share_net2
+--------------------------------------+--------------------------+--------+------+
| id                                   | name                     | status | type |
+--------------------------------------+--------------------------+--------+------+
| 413479b2-0d20-4c58-a9d3-b129fa592d8e | my_ldap_security_service | new    | ldap |
+--------------------------------------+--------------------------+--------+------+
```

You also can dissociate a security service from the share network and confirm that the security service now has an empty list of share networks:

```
$ manila share-network-security-service-remove share_net2
my_ldap_security_service
```

```
$ manila security-service-show my_ldap_security_service
+--------------+----------------------------------+
```

```
| Property        | Value                                |
+-----------------+--------------------------------------+
| status          | new                                  |
| domain          | None                                 |
| password        | None                                 |
| name            | my_ldap_security_service             |
| dns_ip          | 8.8.8.8                              |
| created_at      | 2015-09-25T10:19:06.000000           |
| updated_at      | None                                 |
| server          | 10.254.0.3                           |
| share_networks  | []                                   |
| user            | None                                 |
| project_id      | 20787a7ba11946adad976463b57d8a2f     |
| type            | ldap                                 |
| id              | 413479b2-0d20-4c58-a9d3-b129fa592d8e |
| description     | None                                 |
+-----------------+--------------------------------------+
```

The Shared File Systems service allows you to update a security service field using **manila security-service-update** command with optional arguments such as *--dns-ip*, *--server*, *--domain*, *--user*, *--password*, *--name*, or *--description*.

To remove a security service not associated with any share networks run:

```
$ manila security-service-delete my_ldap_security_service
```

Consistency groups

Consistency groups enable you to create snapshots from multiple file system shares at the same point in time. For example, a database might place its tables, logs, and configurations on separate shares. Store logs, tables, and configurations at the same point in time to effectively restore a database.

The Shared File System service allows you to create a snapshot of the consistency group and restore all shares that were associated with a consistency group.

> **Important**
>
> The **consistency groups and snapshots** are an **experimental** Shared File Systems API in the Liberty release. Contributors can change or remove the experimental part of the Shared File Systems API in further releases without maintaining backward compatibility. Experimental APIs have an `X-OpenStack-Manila-API-Experimental: true` header in their HTTP requests.

Consistency groups

> **Note**
>
> Before using consistency groups, make sure the Shared File System driver that you are running has consistency group support. You can check it in the `manila-scheduler` service reports. The `consistency_group_support` can have the following values:
>
> - `pool` or `host`. Consistency groups are supported. Specifies the level of consistency groups support.
>
> - `false`. Consistency groups are not supported.

The **manila cg-create** command creates a new consistency group. With this command, you can specify a share network, and one or more share types. In the example a consistency group `cgroup1` was created by specifying two comma-separated share types:

```
$ manila cg-create --name cgroup1 --description "My first CG." --share-types
my_type1,default --share-network my_share_net
+---------------------+--------------------------------------+
| Property            | Value                                |
+---------------------+--------------------------------------+
| status              | creating                             |
| description         | My first CG.                         |
| source_cgsnapshot_id | None                                |
| created_at          | 2015-09-29T15:01:12.102472           |
| share_network_id    | 5c3cbabb-f4da-465f-bc7f-fadbe047b85a |
| share_server_id     | None                                 |
| host                | None                                 |
| project_id          | 20787a7ba11946adad976463b57d8a2f     |
| share_types         | a4218aa5-f16a-42b3-945d-113496d40558 |
|                     | c0086582-30a6-4060-b096-a42ec9d66b86 |
| id                  | 6fdd91bc-7a48-48b4-8e40-0f4f98d0ecd6 |
| name                | cgroup1                              |
+---------------------+--------------------------------------+
```

Check that consistency group status is `available`:

```
$ manila cg-show cgroup1
+---------------------+--------------------------------------+
| Property            | Value                                |
+---------------------+--------------------------------------+
| status              | available                            |
| description         | My first CG.                         |
| source_cgsnapshot_id | None                                |
| created_at          | 2015-09-29T15:05:40.000000           |
| share_network_id    | 5c3cbabb-f4da-465f-bc7f-fadbe047b85a |
```

```
| share_server_id      | None                                 |
| host                 | manila@generic1#GENERIC1             |
| project_id           | 20787a7ba11946adad976463b57d8a2f     |
| share_types          | c0086582-30a6-4060-b096-a42ec9d66b86 |
|                      | a4218aa5-f16a-42b3-945d-113496d40558 |
| id                   | 6fdd91bc-7a48-48b4-8e40-0f4f98d0ecd6 |
| name                 | cgroup1                              |
+----------------------+--------------------------------------+
```

To add a share to the consistency group, create a share by adding the *--consistency-group* option where you specify the ID of the consistency group in `available` status:

```
$ manila create nfs 1 --name "Share2" --description "My second share" \
--share-type default --share-network my_share_net --consistency-group cgroup1
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| status                      | None                                 |
| share_type_name             | default                              |
| description                 | My second share                      |
| availability_zone           | None                                 |
| share_network_id            | None                                 |
| export_locations            | []                                   |
| share_server_id             | None                                 |
| host                        | None                                 |
| snapshot_id                 | None                                 |
| is_public                   | False                                |
| task_state                  | None                                 |
| snapshot_support            | True                                 |
| id                          | 7bcd888b-681b-4836-ac9c-c3add4e62537 |
| size                        | 1                                    |
| name                        | Share2                               |
| share_type                  | c0086582-30a6-4060-b096-a42ec9d66b86 |
| created_at                  | 2015-09-29T15:09:24.156387           |
| export_location             | None                                 |
| share_proto                 | NFS                                  |
| consistency_group_id        | 6fdd91bc-7a48-48b4-8e40-0f4f98d0ecd6 |
| source_cgsnapshot_member_id | None                                 |
| project_id                  | 20787a7ba11946adad976463b57d8a2f     |
| metadata                    | {}                                   |
+-----------------------------+--------------------------------------+
```

Administrators can rename the consistency group, or change its description using the **manila cg-update** command. Delete the group with the **manila cg-delete** command.

As an administrator, you can also reset the state of a consistency group and force delete a specified consistency group in any state. Use the `policy.json` file to grant permissions for these actions to other roles.

Use **manila cg-reset-state [–state <state>] <consistency_group>** to update the state of a consistency group explicitly. A valid value of a status are `available`, `error`, `creating`, `deleting`, `error_deleting`. If no state is provided, `available` will be used.

```
$ manila cg-reset-state cgroup1 --state error
```

Use **manila cg-delete <consistency_group> [<consistency_group> ...]** to soft-delete one or more consistency groups.

> **Note**
>
> A consistency group can be deleted only if it has no dependent *Consistency group snapshots*.

```
$ manila cg-delete cgroup1
```

Use **manila cg-delete –force <consistency_group> [<consistency_group> ...]** to force-delete a specified consistency group in any state.

```
$ manila cg-delete --force cgroup1
```

Consistency group snapshots

To create a snapshot, specify the ID or name of the consistency group. After creating a consistency group snapshot, it is possible to generate a new consistency group.

Create a snapshot of consistency group `cgroup1`:

```
$ manila cg-snapshot-create cgroup1 --name CG_snapshot1 --description "A
snapshot of the first CG."
+----------------------+--------------------------------------+
| Property             | Value                                |
+----------------------+--------------------------------------+
| status               | creating                             |
| name                 | CG_snapshot1                         |
| created_at           | 2015-09-29T15:26:16.839704           |
| consistency_group_id | 6fdd91bc-7a48-48b4-8e40-0f4f98d0ecd6 |
| project_id           | 20787a7ba11946adad976463b57d8a2f     |
| id                   | 876ad24c-1efd-4607-a2b1-6a2c90034fa5 |
| description          | A snapshot of the first CG.          |
+----------------------+--------------------------------------+
```

Check the status of created consistency group snapshot:

```
$ manila cg-snapshot-show CG_snapshot1
+----------------------+--------------------------------------+
| Property             | Value                                |
+----------------------+--------------------------------------+
| status               | available                            |
```

```
| name                | CG_snapshot1                         |
| created_at          | 2015-09-29T15:26:22.000000           |
| consistency_group_id | 6fdd91bc-7a48-48b4-8e40-0f4f98d0ecd6 |
| project_id          | 20787a7ba11946adad976463b57d8a2f     |
| id                  | 876ad24c-1efd-4607-a2b1-6a2c90034fa5 |
| description         | A snapshot of the first CG.          |
+---------------------+--------------------------------------+
```

Administrators can rename a consistency group snapshot, change its description using the **cg-snapshot-update** command, or delete it with the **cg-snapshot-delete** command.

A consistency group snapshot can have `members`. To add a member, include the *--consistency-group* optional parameter in the create share command. This ID must match the ID of the consistency group from which the consistency group snapshot was created. Then, while restoring data, and operating with consistency group snapshots, you can quickly find which shares belong to a specified consistency group.

You created the share `Share2` in `cgroup1` consistency group. Since you made a snapshot of it, you can see that the only member of the consistency group snapshot is `Share2` share:

```
$ manila cg-snapshot-members CG_snapshot1
+--------------+------+----------------------------+----------------+--------------+--------------+
| Id           | Size | Created_at                 | Share_protocol | Share_id     | Share_type_id|
+--------------+------+----------------------------+----------------+--------------+--------------+
| 5c62af2b-... | 1    | 2015-09-29T15:26:22.000000 | NFS            | 7bcd888b-... | c0086582-... |
+--------------+------+----------------------------+----------------+--------------+--------------+
```

After you create a consistency group snapshot, you can create a consistency group from the new snapshot:

```
$ manila cg-create --source-cgsnapshot-id 876ad24c-1efd-4607-a2b1-6a2c90034fa5
--name cgroup2 --description "A consistency group from a CG snapshot."
+----------------------+----------------------------------------+
| Property             | Value                                  |
+----------------------+----------------------------------------+
| status               | creating                               |
| description          | A consistency group from a CG snapshot. |
| source_cgsnapshot_id | 876ad24c-1efd-4607-a2b1-6a2c90034fa5   |
| created_at           | 2015-09-29T15:47:47.937991             |
| share_network_id     | None                                   |
| share_server_id      | None                                   |
| host                 | manila@generic1#GENERIC1               |
| project_id           | 20787a7ba11946adad976463b57d8a2f       |
| share_types          | c0086582-30a6-4060-b096-a42ec9d66b86   |
|                      | a4218aa5-f16a-42b3-945d-113496d40558   |
| id                   | ffee08d9-c86c-45e5-861e-175c731daca2   |
| name                 | cgroup2                                |
+----------------------+----------------------------------------+
```

Check the consistency group list. Two groups now appear:

```
$ manila cg-list
+--------------------+---------+----------------------------------------+-----------+
| id                 | name    | description                            | status    |
+--------------------+---------+----------------------------------------+-----------+
| 6fdd91bc-7a48-... | cgroup1 | My first CG.                           | available |
| ffee08d9-c86c-... | cgroup2 | A consistency group from a CG snapshot. | available |
+--------------------+---------+----------------------------------------+-----------+
```

Check a list of the shares. New share with `ba52454e-2ea3-47fa-a683-3176a01295e6` ID appeared after the consistency group `cgroup2` was built from a snapshot with a member.

```
$ manila list
+------+-------+-----+------------+----------+----------+-----------+-------------------------+
| ID   | Name  | Size| Share Proto| Status   | Is Public| Share Type| Host                    |
+------+-------+-----+------------+----------+----------+-----------+-------------------------+
| 7bc..| Share2| 1   | NFS        | available| False    | c008658...| manila@generic1#GENERIC1 |
| ba5..| None  | 1   | NFS        | available| False    | c008658...| manila@generic1#GENERIC1 |
+------+-------+-----+------------+----------+----------+-----------+-------------------------+
```

Print detailed information about new share:

> **Note**
>
> Pay attention on the `source_cgsnapshot_member_id` and `consistency_group_id` fields in a new share. It has `source_cgsnapshot_member_id` that is equal to the ID of the consistency group snapshot and `consistency_group_id` that is equal to the ID of `cgroup2` created from a snapshot.

```
$ manila show ba52454e-2ea3-47fa-a683-3176a01295e6
+-----------------------------+---------------------------------------------------------------+
| Property                    | Value                                                         |
+-----------------------------+---------------------------------------------------------------+
| status                      | available                                                     |
| share_type_name             | default                                                       |
| description                 | None                                                          |
| availability_zone           | None                                                          |
| share_network_id            | None                                                          |
| export_locations            | 10.254.0.5:/shares/share-5acadf4d-f81a-4515-b5ce-3ab641ab4d1e |
| share_server_id             | None                                                          |
| host                        | manila@generic1#GENERIC1                                      |
| snapshot_id                 | None                                                          |
| is_public                   | False                                                         |
| task_state                  | None                                                          |
| snapshot_support            | True                                                          |
| id                          | ba52454e-2ea3-47fa-a683-3176a01295e6                          |
| size                        | 1                                                             |
| name                        | None                                                          |
| share_type                  | c0086582-30a6-4060-b096-a42ec9d66b86                          |
| created_at                  | 2015-09-29T15:47:48.000000                                    |
| share_proto                 | NFS                                                           |
| consistency_group_id        | ffee08d9-c86c-45e5-861e-175c731daca2                          |
| source_cgsnapshot_member_id | 5c62af2b-0870-4d00-b3fa-174831eb15ca                          |
| project_id                  | 20787a7ba11946adad976463b57d8a2f                              |
| metadata                    | {}                                                            |
+-----------------------------+---------------------------------------------------------------+
```

As an administrator, you can also reset the state of a consistency group snapshot with the **cg-snapshot-reset-state** command, and force delete a specified consistency group snapshot in any state using the **cg-snapshot-delete** command with the *--force* key. Use the `policy.json` file to grant permissions for these actions to other roles.

Share replication

Replication of data has a number of use cases in the cloud. One use case is High Availability of the data in a shared file system, used for example, to support a production database. Another use case is ensuring Data Protection; i.e being prepared for a disaster by having a replication location that will be ready to back up your primary data source.

The Shared File System service supports user facing APIs that allow users to create shares that support replication, add and remove share replicas and manage their snapshots and access rules. Three replication types are currently supported and they vary in the semantics associated with the primary share and the secondary copies.

> **Important**
>
> **Share replication** is an **experimental** Shared File Systems API in the Mitaka release. Contributors can change or remove the experimental part of the Shared File Systems API in further releases without maintaining backward compatibility. Experimental APIs have an `X-OpenStack-Manila-API-Experimental: true` header in their HTTP requests.

Replication types supported

Before using share replication, make sure the Shared File System driver that you are running supports this feature. You can check it in the `manila-scheduler` service reports. The `replication_type` capability reported can have one of the following values:

writable
: The driver supports creating `writable` share replicas. All share replicas can be accorded read/write access and would be synchronously mirrored.

readable
: The driver supports creating `read-only` share replicas. All secondary share replicas can be accorded read access. Only the primary (or `active` share replica) can be written into.

dr
: The driver supports creating `dr` (abbreviated from Disaster Recovery) share replicas. A secondary share replica is inaccessible until after a `promotion`.

None
The driver does not support Share Replication.

> **Note**
>
> The term `active` share replica refers to the `primary` share. In `writable` style of replication, all share replicas are `active`, and there could be no distinction of a `primary` share. In `readable` and `dr` styles of replication, a `secondary` share replica may be referred to as `passive`, `non-active` or simply, `replica`.

Configuration

Two new configuration options have been introduced to support Share Replication.

replica_state_update_interval
Specify this option in the `DEFAULT` section of your `manila.conf`. The Shared File Systems service requests periodic update of the replica_state of all `non-active` share replicas. The update occurs with respect to an interval corresponding to this option. If it is not specified, it defaults to 300 seconds.

replication_domain
Specify this option in the backend stanza when using a multi-backend style configuration. The value can be any ASCII string. Two backends that can replicate between each other would have the same `replication_domain`. This comes from the premise that the Shared File Systems service expects Share Replication to be performed between symmetric backends. This option is *required* for using the Share Replication feature.

Health of a share replica

Apart from the `status` attribute, share replicas have the `replica_state` attribute to denote the state of data replication on the storage backend. The `primary` share replica will have it's replica_state attribute set to active. The `secondary` share replicas may have one of the following as their `replica_state`:

in_sync
The share replica is up to date with the `active` share replica (possibly within a backend-specific `recovery point objective`).

out_of_sync
The share replica is out of date (all new share replicas start out in this `replica_state`).

error
When the scheduler fails to schedule this share replica or some potentially irrecoverable

error occurred with regard to updating data for this replica.

Promotion or failover

For `readable` and `dr` types of replication, we refer to the task of switching a non-active share replica with the `active` replica as promotion. For the `writable` style of replication, promotion does not make sense since all share replicas are `active` (or writable) at all times.

The status attribute of the non-active replica being promoted will be set to `replication_change` during its promotion. This has been classified as a `busy` state and thus API interactions with the share are restricted while one of its share replicas is in this state.

Share replication workflows

The following examples have been implemented with the ZFSonLinux driver that is a reference implementation in the Shared File Systems service. It operates in `driver_handles_share_servers=False` mode and supports the `readable` type of replication. In the example, we assume a configuration of two Availability Zones (configuration option: `storage_availability_zone`), called availability_zone_1 and availability_zone_2.

Multiple availability zones are not necessary to use the replication feature. However, the use of an availability zone as a `failure domain` is encouraged.

Pay attention to the network configuration for the ZFS driver. Here, we assume a configuration of `zfs_service_ip` and `zfs_share_export_ip` from two separate networks. The service network is reachable from the host where the `manila-share` service is running. The share export IP is from a network that allows user access.

See Configuring the ZFSonLinux driver for information on how to set up the ZFSonLinux driver.

Creating a share that supports replication

Create a new share type and specify the replication_type as an extra-spec within the share-type being used.

Use the **manila type-create** command to create a new share type. Specify the name and the value for the extra-spec `driver_handles_share_servers`.

```
$ manila type-create readable_type_replication False
```

```
+---------------------+-------------------------------------+
| Property            | Value                               |
+---------------------+-------------------------------------+
| required_extra_specs | driver_handles_share_servers : False |
| Name                | readable_type_replication           |
| Visibility          | public                              |
| is_default          | -                                   |
| ID                  | 3b3ee3f7-6e43-4aa1-859d-0b0511c43074 |
| optional_extra_specs | snapshot_support : True             |
+---------------------+-------------------------------------+
```

Use the **manila type-key** command to set an extra-spec to the share type.

```
$ manila type-key readable_type_replication set replication_type=readable
```

> **Note**
>
> This command has no output. To verify the extra-spec, use the **manila extra-specs-list** command and specify the share type's name or ID as a parameter.

Create a share with the share type

Use the **manila create** command to create a share. Specify the share protocol, size and the availability zone.

```
$ manila create NFS 1 --share_type readable_type_replication --name my_share
--description "This share will have replicas" --az availability_zone_1
+----------------------------+--------------------------------------+
| Property                   | Value                                |
+----------------------------+--------------------------------------+
| status                     | creating                             |
| share_type_name            | readable_type_replication            |
| description                | This share will have replicas        |
| availability_zone          | availability_zone_1                  |
| share_network_id           | None                                 |
| share_server_id            | None                                 |
| host                       |                                      |
| access_rules_status        | active                               |
| snapshot_id                | None                                 |
| is_public                  | False                                |
| task_state                 | None                                 |
| snapshot_support           | True                                 |
| id                         | e496ed61-8f2e-436b-b299-32c3e90991cc |
| size                       | 1                                    |
| name                       | my_share                             |
```

```
| share_type                  | 3b3ee3f7-6e43-4aa1-859d-0b0511c43074 |
| has_replicas                | False                                |
| replication_type            | readable                             |
| created_at                  | 2016-03-29T20:22:18.000000           |
| share_proto                 | NFS                                  |
| consistency_group_id        | None                                 |
| source_cgsnapshot_member_id | None                                 |
| project_id                  | 48a5ca76ac69405e99dc1c13c5195186     |
| metadata                    | {}                                   |
+-----------------------------+--------------------------------------+
```

Use the **manila show** command to retrieve details of the share. Specify the share ID or name as a parameter.

```
$ manila show my_share
+-----------------------------+---------------------------------------------------------------------+
| Property                    | Value                                                               |
+-----------------------------+---------------------------------------------------------------------+
| status                      | available                                                           |
| share_type_name             | readable_type_replication                                           |
| description                 | This share will have replicas                                       |
| availability_zone           | availability_zone_1                                                 |
| share_network_id            | None                                                                |
| export_locations            |                                                                     |
|                             | path =                                                              |
|                             |10.32.62.26:/alpha/manila_share_38efc042_50c2_4825_a6d8_cba2a8277b28|
|                             | preferred = False                                                   |
|                             | is_admin_only = False                                               |
|                             | id = e1d754b5-ec06-42d2-afff-3e98c0013faf                           |
|                             | share_instance_id = 38efc042-50c2-4825-a6d8-cba2a8277b28            |
|                             | path =                                                              |
|                             |172.21.0.23:/alpha/manila_share_38efc042_50c2_4825_a6d8_cba2a8277b28|
|                             | preferred = False                                                   |
|                             | is_admin_only = True                                                |
|                             | id = 6f843ecd-a7ea-4939-86de-e1e01d9e8672                           |
|                             | share_instance_id = 38efc042-50c2-4825-a6d8-cba2a8277b28            |
| share_server_id             | None                                                                |
| host                        | openstack4@zfsonlinux_1#alpha                                       |
| access_rules_status         | active                                                              |
| snapshot_id                 | None                                                                |
| is_public                   | False                                                               |
| task_state                  | None                                                                |
| snapshot_support            | True                                                                |
| id                          | e496ed61-8f2e-436b-b299-32c3e90991cc                                |
| size                        | 1                                                                   |
| name                        | my_share                                                            |
| share_type                  | 3b3ee3f7-6e43-4aa1-859d-0b0511c43074                                |
| has_replicas                | False                                                               |
| replication_type            | readable                                                            |
| created_at                  | 2016-03-29T20:22:18.000000                                          |
| share_proto                 | NFS                                                                 |
| consistency_group_id        | None                                                                |
| source_cgsnapshot_member_id | None                                                                |
| project_id                  | 48a5ca76ac69405e99dc1c13c5195186                                    |
| metadata                    | {}                                                                  |
+-----------------------------+---------------------------------------------------------------------+
```

> **Note**
>
> When you create a share that supports replication, an `active` replica is created for you. You can verify this with the **manila share-replica-list** command.

Creating and promoting share replicas

Create a share replica

Use the **manila share-replica-create** command to create a share replica. Specify the share ID or name as a parameter. You may optionally provide the availability_zone and share_network_id. In the example below, share_network_id is not used since the ZFSonLinux driver does not support it.

```
$ manila share-replica-create my_share --az availability_zone_2
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | creating                             |
| share_id          | e496ed61-8f2e-436b-b299-32c3e90991cc |
| availability_zone | availability_zone_2                  |
| created_at        | 2016-03-29T20:24:53.148992           |
| updated_at        | None                                 |
| share_network_id  | None                                 |
| share_server_id   | None                                 |
| host              |                                      |
| replica_state     | None                                 |
| id                | 78a5ef96-6c36-42e0-b50b-44efe7c1807e |
+-------------------+--------------------------------------+
```

See details of the newly created share replica

Use the **manila share-replica-show** command to see details of the newly created share replica. Specify the share replica's ID as a parameter.

```
$ manila share-replica-show 78a5ef96-6c36-42e0-b50b-44efe7c1807e
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | available                            |
| share_id          | e496ed61-8f2e-436b-b299-32c3e90991cc |
| availability_zone | availability_zone_2                  |
| created_at        | 2016-03-29T20:24:53.000000           |
| updated_at        | 2016-03-29T20:24:58.000000           |
| share_network_id  | None                                 |
| share_server_id   | None                                 |
```

```
| host                | openstack4@zfsonlinux_2#beta          |
| replica_state       | in_sync                               |
| id                  | 78a5ef96-6c36-42e0-b50b-44efe7c1807e |
+---------------------+---------------------------------------+
```

See all replicas of the share

Use the **manila share-replica-list** command to see all the replicas of the share. Specify the share ID or name as an optional parameter.

```
$ manila share-replica-list --share-id my_share
```

```
+--------------------------------------+-----------+---------------+--------------------------------------+-------------------------------------+-------------------+----------------------------+
| ID                                   | Status    | Replica State | Share ID                             | Host                                | Availability Zone | Updated At                 |
+--------------------------------------+-----------+---------------+--------------------------------------+-------------------------------------+-------------------+----------------------------+
| 38efc042-50c2-4825-a6d8-cba2a8277b28 | available | active        | e496ed61-8f2e-436b-b299-32c3e90991cc | openstack4@zfsonlinux_1#alpha | availability_zone_1 | 2016-03-29T20:22:19.000000 |
| 78a5ef96-6c36-42e0-b50b-44efe7c1807e | available | in_sync       | e496ed61-8f2e-436b-b299-32c3e90991cc | openstack4@zfsonlinux_2#beta  | availability_zone_2 | 2016-03-29T20:24:58.000000 |
+--------------------------------------+-----------+---------------+--------------------------------------+-------------------------------------+-------------------+----------------------------+
```

Promote the secondary share replica to be the new active replica

Use the **manila share-replica-promote** command to promote a non-active share replica to become the `active` replica. Specify the non-active replica's ID as a parameter.

```
$ manila share-replica-promote 78a5ef96-6c36-42e0-b50b-44efe7c1807e
```

> **Note**
>
> This command has no output.

The promotion may take time. During the promotion, the `replica_state` attribute of the share replica being promoted will be set to `replication_change`.

```
$ manila share-replica-list --share-id my_share
```

```
+--------------------------------------+-----------+--------------------+--------------------------------------+-------------------------------------+-------------------+----------------------------+
| ID                                   | Status    | Replica State      | Share ID                             | Host                                | Availability Zone | Updated At                 |
+--------------------------------------+-----------+--------------------+--------------------------------------+-------------------------------------+-------------------+----------------------------+
| 38efc042-50c2-4825-a6d8-cba2a8277b28 | available |       active       | e496ed61-8f2e-436b-b299-32c3e90991cc | openstack4@zfsonlinux_1#alpha | availability_zone_1 | 2016-03-29T20:32:19.000000 |
| 78a5ef96-6c36-42e0-b50b-44efe7c1807e | available | replication_change | e496ed61-8f2e-436b-b299-32c3e90991cc | openstack4@zfsonlinux_2#beta  | availability_zone_2 | 2016-03-29T20:32:19.000000 |
+--------------------------------------+-----------+--------------------+--------------------------------------+-------------------------------------+-------------------+----------------------------+
```

Once the promotion is complete, the `replica_state` will be set to `active`.

```
$ manila share-replica-list --share-id my_share
```

```
+--------------------------------------+-----------+---------------+--------------------------------------+-------------------------------------+-------------------+----------------------------+
| ID                                   | Status    | Replica State | Share ID                             | Host                                | Availability Zone | Updated At                 |
+--------------------------------------+-----------+---------------+--------------------------------------+-------------------------------------+-------------------+----------------------------+
| 38efc042-50c2-4825-a6d8-cba2a8277b28 | available | in_sync       | e496ed61-8f2e-436b-b299-32c3e90991cc | openstack4@zfsonlinux_1#alpha | availability_zone_1 | 2016-03-29T20:32:19.000000 |
| 78a5ef96-6c36-42e0-b50b-44efe7c1807e | available | active        | e496ed61-8f2e-436b-b299-32c3e90991cc | openstack4@zfsonlinux_2#beta  | availability_zone_2 | 2016-03-29T20:32:19.000000 |
+--------------------------------------+-----------+---------------+--------------------------------------+-------------------------------------+-------------------+----------------------------+
```

Access rules

Create an IP access rule for the share

Use the **manila access-allow** command to add an access rule. Specify the share ID or name, protocol and the target as parameters.

```
$ manila access-allow my_share ip 0.0.0.0/0 --access-level rw
```

```
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| share_id     | e496ed61-8f2e-436b-b299-32c3e90991cc |
| access_type  | ip                                   |
| access_to    | 0.0.0.0/0                            |
| access_level | rw                                   |
| state        | new                                  |
| id           | 8b339cdc-c1e0-448f-bf6d-f068ee6e8f45 |
+--------------+--------------------------------------+
```

> **Note**
>
> Access rules are not meant to be different across the replicas of the share. However, as per the type of replication, drivers may choose to modify the access level prescribed. In the above example, even though read/write access was requested for the share, the driver will provide read-only access to the non-active replica to the same target, because of the semantics of the replication type: `readable`. However, the target will have read/write access to the (currently) non-active replica when it is promoted to become the `active` replica.

The **manila access-deny** command can be used to remove a previously applied access rule.

List the export locations of the share

Use the **manila share-export-locations-list** command to list the export locations of a share.

```
$ manila share-export-location-list my_share
+--------------------------------------+-----------------------------------------------------------------------------+-----------+
| ID                                   | Path                                                                        | Preferred |
+--------------------------------------+-----------------------------------------------------------------------------+-----------+
| 3ed3fbf5-2fa1-4dc0-8440-a0af72398cb6 | 10.32.62.21:/beta/subdir/manila_share_78a5ef96_6c36_42e0_b50b_44efe7c1807e| False     |
| 6f843ecd-a7ea-4939-86de-e1e01d9e8672 | 172.21.0.23:/alpha/manila_share_38efc042_50c2_4825_a6d8_cba2a8277b28      | False     |
| e1d754b5-ec06-42d2-afff-3e98c0013faf | 10.32.62.26:/alpha/manila_share_38efc042_50c2_4825_a6d8_cba2a8277b28      | False     |
| f3c5585f-c2f7-4264-91a7-a4a1e754e686 | 172.21.0.29:/beta/subdir/manila_share_78a5ef96_6c36_42e0_b50b_44efe7c1807e| False     |
+--------------------------------------+-----------------------------------------------------------------------------+-----------+
```

Identify the export location corresponding to the share replica on the user accessible network and you may mount it on the target node.

> **Note**
>
> As an administrator, you can list the export locations for a particular share replica by using the **manila share-instance-export-location-list** command and specifying the share replica's ID as a parameter.

Snapshots

Create a snapshot of the share

Use the **manila snapshot-create** command to create a snapshot of the share. Specify the share ID or name as a parameter.

```
$ manila snapshot-create my_share --name "my_snapshot"
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | creating                             |
| share_id          | e496ed61-8f2e-436b-b299-32c3e90991cc |
| description       | None                                 |
| created_at        | 2016-03-29T21:14:03.000000           |
| share_proto       | NFS                                  |
| provider_location | None                                 |
| id                | 06cdccaf-93a0-4e57-9a39-79fb1929c649 |
| size              | 1                                    |
| share_size        | 1                                    |
| name              | my_snapshot                          |
+-------------------+--------------------------------------+
```

Show the details of the snapshot

Use the **manila snapshot-show** to view details of a snapshot. Specify the snapshot ID or name as a parameter.

```
$ manila snapshot-show my_snapshot
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | available                            |
| share_id          | e496ed61-8f2e-436b-b299-32c3e90991cc |
| description       | None                                 |
| created_at        | 2016-03-29T21:14:03.000000           |
| share_proto       | NFS                                  |
| provider_location | None                                 |
| id                | 06cdccaf-93a0-4e57-9a39-79fb1929c649 |
| size              | 1                                    |
| share_size        | 1                                    |
| name              | my_snapshot                          |
+-------------------+--------------------------------------+
```

> **Note**
>
> The `status` attribute of a snapshot will transition from `creating` to `available` only when it is present on all the share replicas that have their `replica_state` attribute set to `active` or `in_sync`.

> Likewise, the `replica_state` attribute of a share replica will transition from `out_of_sync` to `in_sync` only when all `available` snapshots are present on it.

Planned failovers

As an administrator, you can use the **manila share-replica-resync** command to attempt to sync data between `active` and `non-active` share replicas of a share before promotion. This will ensure that share replicas have the most up-to-date data and their relationships can be safely switched.

```
$ manila share-replica-resync 38efc042-50c2-4825-a6d8-cba2a8277b28
```

> **Note**
>
> This command has no output.

Updating attributes

If an error occurs while updating data or replication relationships (during a `promotion`), the Shared File Systems service may not be able to determine the consistency or health of a share replica. It may require administrator intervention to make any fixes on the storage backend as necessary. In such a situation, state correction within the Shared File Systems service is possible.

As an administrator, you can:

Reset the `status` attribute of a share replica

Use the **manila share-replica-reset-state** command to reset the `status` attribute. Specify the share replica's ID as a parameter and use the `--state` option to specify the state intended.

```
$ manila share-replica-reset-state 38efc042-50c2-4825-a6d8-cba2a8277b28
--state=available
```

> **Note**
>
> This command has no output.

Reset the `replica_state` attribute

Use the **manila share-replica-reset-replica-state** command to reset the `replica_state` attribute. Specify the share replica's ID and use the `--state` option to specify the state intended.

```
$ manila share-replica-reset-replica-state 38efc042-50c2-4825-a6d8-cba2a8277b28
--state=out_of_sync
```

> **Note**
>
> This command has no output.

Force delete a specified share replica in any state

Use the **manila share-replica-delete** command with the '–force' key to remove the share replica, regardless of the state it is in.

```
$ manila share-replica-show 9513de5d-0384-4528-89fb-957dd9b57680
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | error                                |
| share_id          | e496ed61-8f2e-436b-b299-32c3e90991cc |
| availability_zone | availability_zone_1                  |
| created_at        | 2016-03-30T01:32:47.000000           |
| updated_at        | 2016-03-30T01:34:25.000000           |
| share_network_id  | None                                 |
| share_server_id   | None                                 |
| host              | openstack4@zfsonlinux_1#alpha        |
| replica_state     | out_of_sync                          |
| id                | 38efc042-50c2-4825-a6d8-cba2a8277b28 |
+-------------------+--------------------------------------+

$ manila share-replica-delete --force 38efc042-50c2-4825-a6d8-cba2a8277b28
```

> **Note**
>
> This command has no output.

Use the `policy.json` file to grant permissions for these actions to other roles.

Deleting share replicas

Use the **manila share-replica-delete** command with the share replica's ID to delete a share replica.

```
$ manila share-replica-delete 38efc042-50c2-4825-a6d8-cba2a8277b28
```

> **Note**
>
> This command has no output.

> **Note**
>
> You cannot delete the last `active` replica with this command. You should use the **manila delete** command to remove the share.

Multi-storage configuration

The Shared File Systems service can provide access to multiple file storage back ends. In general, the workflow with multiple back ends looks similar to the Block Storage service one, see *Configure multiple-storage back ends in Block Storage service*.

Using `manila.conf`, you can spawn multiple share services. To do it, you should set the enabled_share_backends flag in the `manila.conf` file. This flag defines the comma-separated names of the configuration stanzas for the different back ends. One name is associated to one configuration group for a back end.

The following example runs three configured share services:

```
[DEFAULT]
enabled_share_backends=backendEMC2,backendGeneric1,backendNetApp

[backendGeneric1]
share_driver=manila.share.drivers.generic.GenericShareDriver
share_backend_name=one_name_for_two_backends
service_instance_user=ubuntu_user
service_instance_password=ubuntu_user_password
service_image_name=ubuntu_image_name
path_to_private_key=/home/foouser/.ssh/id_rsa
path_to_public_key=/home/foouser/.ssh/id_rsa.pub

[backendEMC2]
share_driver=manila.share.drivers.emc.driver.EMCShareDriver
share_backend_name=backendEMC2
emc_share_backend=vnx
emc_nas_server=1.1.1.1
emc_nas_password=password
emc_nas_login=user
emc_nas_server_container=server_3
emc_nas_pool_name="Pool 2"

[backendNetApp]
share_driver = manila.share.drivers.netapp.common.NetAppDriver
driver_handles_share_servers = True
```

```
share_backend_name=backendNetApp
netapp_login=user
netapp_password=password
netapp_server_hostname=1.1.1.1
netapp_root_volume_aggregate=aggr01
```

To spawn separate groups of share services, you can use separate configuration files. If it is necessary to control each back end in a separate way, you should provide a single configuration file per each back end.

- Scheduling
- Manage shares services

Scheduling

The Shared File Systems service uses a scheduler to provide unified access for a variety of different types of shared file systems. The scheduler collects information from the active shared services, and makes decisions such as what shared services will be used to create a new share. To manage this process, the Shared File Systems service provides Share types API.

A share type is a list from key-value pairs called extra-specs. The scheduler uses required and un-scoped extra-specs to look up the shared service most suitable for a new share with the specified share type. For more information about extra-specs and their type, see Capabilities and Extra-Specs section in developer documentation.

The general scheduler workflow:

1. Share services report information about their existing pool number, their capacities, and their capabilities.
2. When a request on share creation arrives, the scheduler picks a service and pool that best serves the request, using share type filters and back end capabilities. If back end capabilities pass through, all filters request the selected back end where the target pool resides.
3. The share driver receives a reply on the request status, and lets the target pool serve the request as the scheduler instructs. The scoped and un-scoped share types are available for the driver implementation to use as needed.

Manage shares services

The Shared File Systems service provides API that allows to manage running share services (Share services API). Using the **manila service-list** command, it is possible to get a list of all kinds of running services. To select only share services, you can pick items that have field `binary` equal to `manila-share`. Also, you can enable or disable share services using raw API requests. Disabling means that share services are excluded from the scheduler cycle and new shares will not be placed on the disabled back end. However, shares from this service stay available.

Networking

Unlike the OpenStack Block Storage service, the Shared File Systems service must connect to the Networking service. The share service requires the option to self-manage share servers. For client authentication and authorization, you can configure the Shared File Systems service to work with different network authentication services, like LDAP, Kerberos protocols, or Microsoft Active Directory.

- Share networks
  - How to create share network
- Network plug-ins
  - What network plug-ins are available?

Share networks

Share network is an entity that encapsulates interaction with the OpenStack Networking service. If the share driver that you selected runs in a mode requiring Networking service interaction, specify the share network when creating a new share network.

How to create share network

To list networks in a tenant, run:

```
$ neutron net-list
+--------------+---------+--------------------------------------+
| id           | name    | subnets                              |
+--------------+---------+--------------------------------------+
| bee7411d-... | public  | 884a6564-0f11-... 2001:db8::/64      |
|              |         | e6da81fa-5d5f-... 172.24.4.0/24      |
| 5ed5a854-... | private | 74dcfb5a-b4d7-... 10.0.0.0/24        |
|              |         | cc297be2-5213-... fd7d:177d:a48b::/64 |
+--------------+---------+--------------------------------------+
```

A share network stores network information that share servers can use where shares are hosted. You can associate a share with a single share network. When you create or update a share, you can optionally specify the ID of a share network through which instances can access the share.

When you create a share network, you can specify only one type of network:

- OpenStack Networking (neutron). Specify a network ID and subnet ID. In this case `manila.network.nova_network_plugin.NeutronNetworkPlugin` will be used.
- Legacy networking (nova-network). Specify a network ID. In this case `manila.network.nova_network_plugin.NoveNetworkPlugin` will be used.

For more information about supported plug-ins for share networks, see *Network plug-ins*.

A share network has these attributes:

- The IP block in Classless Inter-Domain Routing (CIDR) notation from which to allocate the network.
- The IP version of the network.
- The network type, which is vlan, vxlan, gre, or flat.

If the network uses segmentation, a segmentation identifier. For example, VLAN, VXLAN, and GRE networks use segmentation.

To create a share network with private network and subnetwork, run:

```
$ manila share-network-create --neutron-net-id 5ed5a854-21dc-4ed3-870a-
117b7064eb21 \
--neutron-subnet-id 74dcfb5a-b4d7-4855-86f5-a669729428dc --name my_share_net --description "My first
share network"
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| name              | my_share_net                         |
| segmentation_id   | None                                 |
| created_at        | 2015-09-24T12:06:32.602174           |
| neutron_subnet_id | 74dcfb5a-b4d7-4855-86f5-a669729428dc |
| updated_at        | None                                 |
| network_type      | None                                 |
| neutron_net_id    | 5ed5a854-21dc-4ed3-870a-117b7064eb21 |
| ip_version        | None                                 |
| nova_net_id       | None                                 |
| cidr              | None                                 |
| project_id        | 20787a7ba11946adad976463b57d8a2f     |
| id                | 5c3cbabb-f4da-465f-bc7f-fadbe047b85a |
| description       | My first share network               |
+-------------------+--------------------------------------+
```

The `segmentation_id`, `cidr`, `ip_version`, and `network_type` share network attributes are automatically set to the values determined by the network provider.

To check the network list, run:

```
$ manila share-network-list
+--------------------------------------+--------------+
| id                                   | name         |
+--------------------------------------+--------------+
| 5c3cbabb-f4da-465f-bc7f-fadbe047b85a | my_share_net |
+--------------------------------------+--------------+
```

If you configured the generic driver with `driver_handles_share_servers = True` (with the share servers) and already had previous operations in the Shared File Systems service, you can see `manila_service_network` in the neutron list of networks. This network was created by the generic driver for internal use.

```
$ neutron net-list
+--------------+-------------------------+--------------------------------------+
| id           | name                    | subnets                              |
+--------------+-------------------------+--------------------------------------+
| 3b5a629a-e...| manila_service_network | 4f366100-50... 10.254.0.0/28         |
| bee7411d-d...| public                  | 884a6564-01... 2001:db8::/64         |
|              |                         | e6da81fa-55... 172.24.4.0/24         |
| 5ed5a854-2...| private                 | 74dcfb5a-bd... 10.0.0.0/24           |
|              |                         | cc297be2-51... fd7d:177d:a48b::/64   |
+--------------+-------------------------+--------------------------------------+
```

You also can see detailed information about the share network including `network_type`, and `segmentation_id` fields:

```
$ neutron net-show manila_service_network
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 3b5a629a-e7a1-46a3-afb2-ab666fb884bc |
| mtu                       | 0                                    |
| name                      | manila_service_network               |
| port_security_enabled     | True                                 |
| provider:network_type     | vxlan                                |
| provider:physical_network |                                      |
| provider:segmentation_id  | 1068                                 |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   | 4f366100-5108-4fa2-b5b1-989a121c1403 |
| tenant_id                 | 24c6491074e942309a908c674606f598     |
+---------------------------+--------------------------------------+
```

You also can add and remove the security services from the share network. For more detail, see *Security services*.

Network plug-ins

The Shared File Systems service architecture defines an abstraction layer for network resource provisioning and allowing administrators to choose from a different options for how network resources are assigned to their tenants' networked storage. There are a set of network plug-ins that provide a variety of integration approaches with the network services that are available with OpenStack.

The Shared File Systems service may need a network resource provisioning if share service with specified driver works in mode, when a share driver manages lifecycle of share servers on its own. This behavior is defined by a flag `driver_handles_share_servers` in share service configuration. When `driver_handles_share_servers` is set to `True`, a share driver will be called to create share servers for shares using information provided within a share network. This information will be provided to one of the enabled network plug-ins that will handle reservation, creation and deletion of network resources including IP addresses and network interfaces.

What network plug-ins are available?

There are three different network plug-ins and five python classes in the Shared File Systems service:

1. Network plug-in for using the OpenStack Networking service. It allows to use any network segmentation that the Networking service supports. It is up to each share driver to support at least one network segmentation type.

    1. `manila.network.neutron.neutron_network_plugin.NeutronNetworkPlugin`. This is a default network plug-in. It requires the `neutron_net_id` and the `neutron_subnet_id` to be provided when defining the share network that will be used for the creation of share servers. The user may define any number of share networks corresponding to the various physical network segments in a tenant environment.
    2. `manila.network.neutron.neutron_network_plugin.NeutronSingleNetworkPlugin`. This is a simplification of the previous case. It accepts values for `neutron_net_id` and `neutron_subnet_id` from the `manila.conf` configuration file and uses one network for all shares.

    When only a single network is needed, the NeutronSingleNetworkPlugin (1.b) is a simple solution. Otherwise NeutronNetworkPlugin (1.a) should be chosen.

2. Network plug-in for working with OpenStack Networking from the Compute service. It supports either flat networks or VLAN-segmented networks.

    1. `manila.network.nova_network_plugin.NovaNetworkPlugin`. This plug-in serves the networking needs when `Nova networking` is configured in the

cloud instead of Neutron. It requires a single parameter, `nova_net_id`.

2. `manila.network.nova_network_plugin.NovaSingleNetworkPlugin`. This plug-in works the same way as `manila.network.nova_network_plugin.NovaNetworkPlugin`, except it takes `nova_net_id` from the Shared File Systems service configuration file and creates the share servers using only one network.

When only a single network is needed, the NovaSingleNetworkPlugin (2.b) is a simple solution. Otherwise NovaNetworkPlugin (2.a) should be chosen.

3. Network plug-in for specifying networks independently from OpenStack networking services.

    1. `manila.network.standalone_network_plugin.StandaloneNetworkPlugin`. This plug-in uses a pre-existing network that is available to the manila-share host. This network may be handled either by OpenStack or be created independently by any other means. The plug-in supports any type of network - flat and segmented. As above, it is completely up to the share driver to support the network type for which the network plug-in is configured.

> **Note**
>
> These network plug-ins were introduced in the OpenStack Kilo release. In the OpenStack Juno version, only NeutronNetworkPlugin is available.

More information about network plug-ins can be found in Manila developer documentation

Troubleshoot Shared File Systems service

Failures in Share File Systems service during a share creation

Problem

New shares can enter `error` state during the creation process.

Solution

1. Make sure, that share services are running in debug mode. If the debug mode is not set, you will not get any tips from logs how to fix your issue.
2. Find what share service holds a specified share. To do that, run command **manila**

show <share_id_or_name> and find a share host in the output. Host uniquely identifies what share service holds the broken share.

3. Look thought logs of this share service. Usually, it can be found at `/etc/var/log/manila-share.log`. This log should contain kind of traceback with extra information to help you to find the origin of issues.

No valid host was found

Problem

If a share type contains invalid extra specs, the scheduler will not be able to locate a valid host for the shares.

Solution

To diagnose this issue, make sure that scheduler service is running in debug mode. Try to create a new share and look for message `Failed to schedule create_share: No valid host was found.` in `/etc/var/log/manila-scheduler.log`.

To solve this issue look carefully through the list of extra specs in the share type, and the list of share services reported capabilities. Make sure that extra specs are pointed in the right way.

Created share is unreachable

Problem

By default, a new share does not have any active access rules.

Solution

To provide access to new share, you need to create appropriate access rule with the right value. The value must defines access.

Service becomes unavailable after upgrade

Problem

After upgrading the Shared File Systems service from version v1 to version v2.x, you must update the service endpoint in the OpenStack Identity service. Otherwise, the service may become unavailable.

Solution

1. To get the service type related to the Shared File Systems service, run:

   ```
   # openstack endpoint list

   # openstack endpoint show <share-service-type>
   ```

   You will get the endpoints expected from running the Shared File Systems service.

2. Make sure that these endpoints are updated. Otherwise, delete the outdated endpoints and create new ones.

Failures during management of internal resources

Problem

The Shared File System service manages internal resources effectively. Administrators may need to manually adjust internal resources to handle failures.

Solution

Some drivers in the Shared File Systems service can create service entities, like servers and networks. If it is necessary, you can log in to tenant `service` and take manual control over it.

Networking

Learn OpenStack Networking concepts, architecture, and basic and advanced `neutron` and `nova` command-line interface (CLI) commands.

Introduction to Networking

The Networking service, code-named neutron, provides an API that lets you define network connectivity and addressing in the cloud. The Networking service enables operators to leverage different networking technologies to power their cloud networking. The Networking service also provides an API to configure and manage a variety of network services ranging from L3 forwarding and NAT to load balancing, edge firewalls, and IPsec VPN.

For a detailed description of the Networking API abstractions and their attributes, see the OpenStack Networking API v2.0 Reference.

> **Note**
>
> If you use the Networking service, do not run the Compute `nova-network` service (like you do in traditional Compute deployments). When you configure networking, see the Compute-related topics in this Networking section.

Networking API

Networking is a virtual network service that provides a powerful API to define the network connectivity and IP addressing that devices from other services, such as Compute, use.

The Compute API has a virtual server abstraction to describe computing resources. Similarly, the Networking API has virtual network, subnet, and port abstractions to describe networking resources.

Resource	Description
Network	An isolated L2 segment, analogous to VLAN in the physical networking world.
Subnet	A block of v4 or v6 IP addresses and associated configuration state.
Port	A connection point for attaching a single device, such as the NIC of a virtual server, to a virtual network. Also describes the associated network configuration, such as the MAC and IP addresses to be used on that port.

Networking resources

To configure rich network topologies, you can create and configure networks and subnets

and instruct other OpenStack services like Compute to attach virtual devices to ports on these networks.

In particular, Networking supports each tenant having multiple private networks and enables tenants to choose their own IP addressing scheme, even if those IP addresses overlap with those that other tenants use.

The Networking service:

- Enables advanced cloud networking use cases, such as building multi-tiered web applications and enabling migration of applications to the cloud without changing IP addresses.
- Offers flexibility for administrators to customize network offerings.
- Enables developers to extend the Networking API. Over time, the extended functionality becomes part of the core Networking API.

Configure SSL support for networking API

OpenStack Networking supports SSL for the Networking API server. By default, SSL is disabled but you can enable it in the `neutron.conf` file.

Set these options to configure SSL:

`use_ssl = True`
Enables SSL on the networking API server.

`ssl_cert_file = PATH_TO_CERTFILE`
Certificate file that is used when you securely start the Networking API server.

`ssl_key_file = PATH_TO_KEYFILE`
Private key file that is used when you securely start the Networking API server.

`ssl_ca_file = PATH_TO_CAFILE`
Optional. CA certificate file that is used when you securely start the Networking API server. This file verifies connecting clients. Set this option when API clients must authenticate to the API server by using SSL certificates that are signed by a trusted CA.

`tcp_keepidle = 600`
The value of TCP_KEEPIDLE, in seconds, for each server socket when starting the API server. Not supported on OS X.

`retry_until_window = 30`
Number of seconds to keep retrying to listen.

`backlog = 4096`
Number of backlog requests with which to configure the socket.

Load-Balancer-as-a-Service (LBaaS) overview

Load-Balancer-as-a-Service (LBaaS) enables Networking to distribute incoming requests evenly among designated instances. This distribution ensures that the workload is shared predictably among instances and enables more effective use of system resources. Use one of these load balancing methods to distribute incoming requests:

Round robin
Rotates requests evenly between multiple instances.

Source IP
Requests from a unique source IP address are consistently directed to the same instance.

Least connections
Allocates requests to the instance with the least number of active connections.

Feature	Description
Monitors	LBaaS provides availability monitoring with the `ping`, TCP, HTTP and HTTPS GET methods. Monitors are implemented to determine whether pool members are available to handle requests.
Management	LBaaS is managed using a variety of tool sets. The REST API is available for programmatic administration and scripting. Users perform administrative management of load balancers through either the CLI (`neutron`) or the OpenStack Dashboard.
Connection limits	Ingress traffic can be shaped with *connection limits*. This feature allows workload control, and can also assist with mitigating DoS (Denial of Service) attacks.
Session persistence	LBaaS supports session persistence by ensuring incoming requests are routed to the same instance within a pool of multiple instances. LBaaS supports routing decisions based on cookies and source IP address.

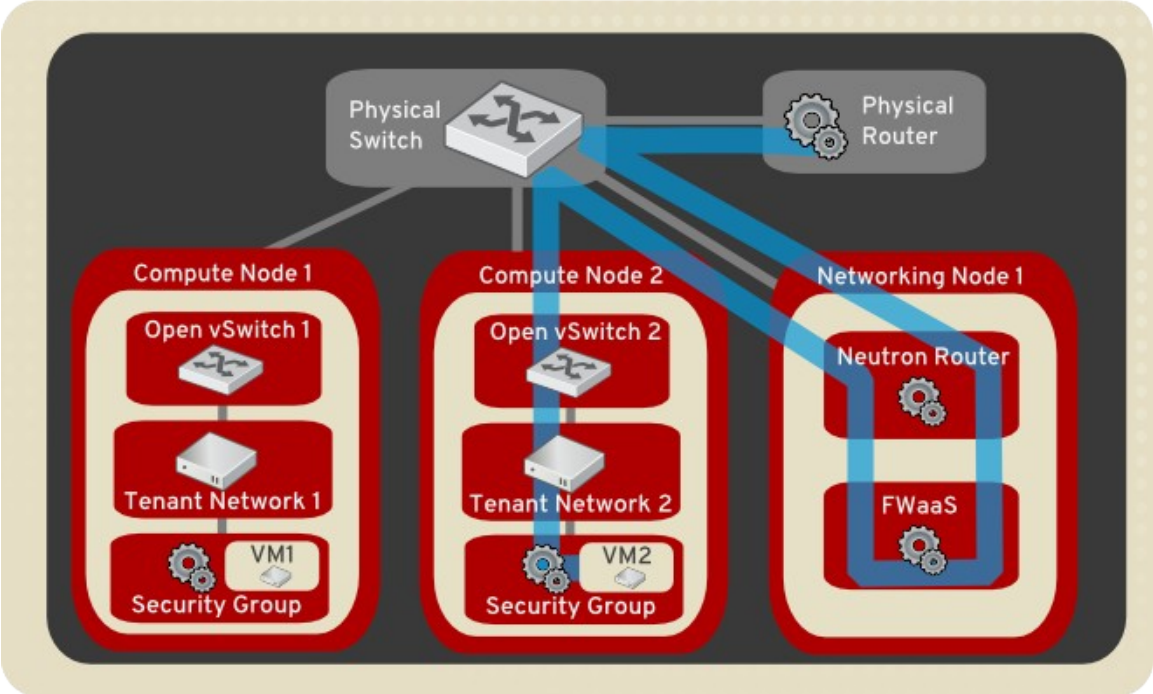
Firewall-as-a-Service (FWaaS) overview

The Firewall-as-a-Service (FWaaS) plug-in adds perimeter firewall management to Networking. FWaaS uses iptables to apply firewall policy to all Networking routers within a

project. FWaaS supports one firewall policy and logical firewall instance per project.

Whereas security groups operate at the instance-level, FWaaS operates at the perimeter to filter traffic at the neutron router.

The example diagram illustrates the flow of ingress and egress traffic for the VM2 instance:



Enable FWaaS

FWaaS management options are also available in the Dashboard.

1. Enable the FWaaS plug-in in the `/etc/neutron/neutron.conf` file:

```
service_plugins = firewall
[service_providers]
...
service_provider = FIREWALL:Iptables:neutron.agent.linux.iptables_
firewall.OVSHybridIptablesFirewallDriver:default

[fwaas]
driver = neutron_fwaas.services.firewall.drivers.linux.iptables_
fwaas.IptablesFwaasDriver
enabled = True
```

> **Note**
>
> On Ubuntu, modify the `[fwaas]` section in the `/etc/neutron/fwaas_driver.ini` file instead of `/etc/neutron/neutron.conf`.

2. Create the required tables in the database:

```
# neutron-db-manage --subproject neutron-fwaas upgrade head
```

3. Enable the option in the `local_settings.py` file, which is typically located on the controller node:

```
OPENSTACK_NEUTRON_NETWORK = {
    ...
    'enable_firewall' = True,
    ...
}
```

> **Note**
>
> By default, `enable_firewall` option value is `True` in `local_settings.py` file.

   Apply the settings by restarting the web server.

4. Restart the `neutron-l3-agent` and `neutron-server` services to apply the settings.

Configure Firewall-as-a-Service

Create the firewall rules and create a policy that contains them. Then, create a firewall that applies the policy.

1. Create a firewall rule:

```
$ neutron firewall-rule-create --protocol {tcp,udp,icmp,any} \
  --source-ip-address SOURCE_IP_ADDRESS \
  --destination-ip-address DESTINATION_IP_ADDRESS \
  --source-port SOURCE_PORT_RANGE --destination-port DEST_PORT_RANGE \
  --action {allow,deny,reject}
```

   The Networking client requires a protocol value; if the rule is protocol agnostic, you can use the any value.

> **Note**
>
> When the source or destination IP address are not of the same IP version (for example, IPv6), the command returns an error.

2. Create a firewall policy:

```
$ neutron firewall-policy-create --firewall-rules \
  "FIREWALL_RULE_IDS_OR_NAMES" myfirewallpolicy
```

Separate firewall rule IDs or names with spaces. The order in which you specify the rules is important.

You can create a firewall policy without any rules and add rules later, as follows:

- To add multiple rules, use the update operation.
- To add a single rule, use the insert-rule operation.

For more details, see Networking command-line client in the OpenStack Command-Line Interface Reference.

> **Note**
>
> FWaaS always adds a default `deny all` rule at the lowest precedence of each policy. Consequently, a firewall policy with no rules blocks all traffic by default.

3. Create a firewall:

```
$ neutron firewall-create FIREWALL_POLICY_UUID
```

> **Note**
>
> The firewall remains in PENDING_CREATE state until you create a Networking router and attach an interface to it.

Allowed-address-pairs

`Allowed-address-pairs` enables you to specify mac_address and ip_address(cidr) pairs that pass through a port regardless of subnet. This enables the use of protocols such as VRRP, which floats an IP address between two instances to enable fast data plane failover.

> **Note**
>
> Currently, only the ML2, Open vSwitch, and VMware NSX plug-ins support the allowed-address-pairs extension.

Basic allowed-address-pairs operations.

- Create a port with a specified allowed address pair:

```
$ neutron port-create net1 --allowed-address-pairs type=dict \
  list=true mac_address=MAC_ADDRESS,ip_address=IP_CIDR
```

- Update a port by adding allowed address pairs:

```
$ neutron port-update PORT_UUID --allowed-address-pairs type=dict \
list=true mac_address=MAC_ADDRESS,ip_address=IP_CIDR
```

Virtual-Private-Network-as-a-Service (VPNaaS)

The VPNaaS extension enables OpenStack tenants to extend private networks across the internet.

VPNaas is a *service*. It is a parent object that associates a VPN with a specific subnet and router. Only one VPN service object can be created for each router and each subnet. However, each VPN service object can have any number of IP security connections.

The Internet Key Exchange (IKE) policy specifies the authentication and encryption algorithms to use during phase one and two negotiation of a VPN connection. The IP security policy specifies the authentication and encryption algorithm and encapsulation mode to use for the established VPN connection. Note that you cannot update the IKE and IPSec parameters for live tunnels.

You can set parameters for site-to-site IPsec connections, including peer CIDRs, MTU, authentication mode, peer address, DPD settings, and status.

The current implementation of the VPNaaS extension provides:

- Site-to-site VPN that connects two private networks.
- Multiple VPN connections per tenant.
- IKEv1 policy support with 3des, aes-128, aes-256, or aes-192 encryption.
- IPSec policy support with 3des, aes-128, aes-192, or aes-256 encryption, sha1 authentication, ESP, AH, or AH-ESP transform protocol, and tunnel or transport mode encapsulation.
- Dead Peer Detection (DPD) with hold, clear, restart, disabled, or restart-by-peer actions.

The VPNaaS driver plugin can be configured in the neutron configuration file. You can then enable the service.

Networking architecture

Before you deploy Networking, it is useful to understand the Networking services and how they interact with the OpenStack components.

Overview

Networking is a standalone component in the OpenStack modular architecture. It is positioned alongside OpenStack components such as Compute, Image service, Identity, or Dashboard. Like those components, a deployment of Networking often involves deploying several services to a variety of hosts.

The Networking server uses the neutron-server daemon to expose the Networking API and enable administration of the configured Networking plug-in. Typically, the plug-in requires access to a database for persistent storage (also similar to other OpenStack services).

If your deployment uses a controller host to run centralized Compute components, you can deploy the Networking server to that same host. However, Networking is entirely standalone and can be deployed to a dedicated host. Depending on your configuration, Networking can also include the following agents:

Agent	Description
plug-in agent (`neutron-*-agent`)	Runs on each hypervisor to perform local vSwitch configuration. The agent that runs, depends on the plug-in that you use. Certain plug-ins do not require an agent.
dhcp agent (`neutron-dhcp-agent`)	Provides DHCP services to tenant networks. Required by certain plug-ins.
l3 agent (`neutron-l3-agent`)	Provides L3/NAT forwarding to provide external network access for VMs on tenant networks. Required by certain plug-ins.
metering agent (`neutron-metering-agent`)	Provides L3 traffic metering for tenant networks.

These agents interact with the main neutron process through RPC (for example, RabbitMQ or Qpid) or through the standard Networking API. In addition, Networking integrates with

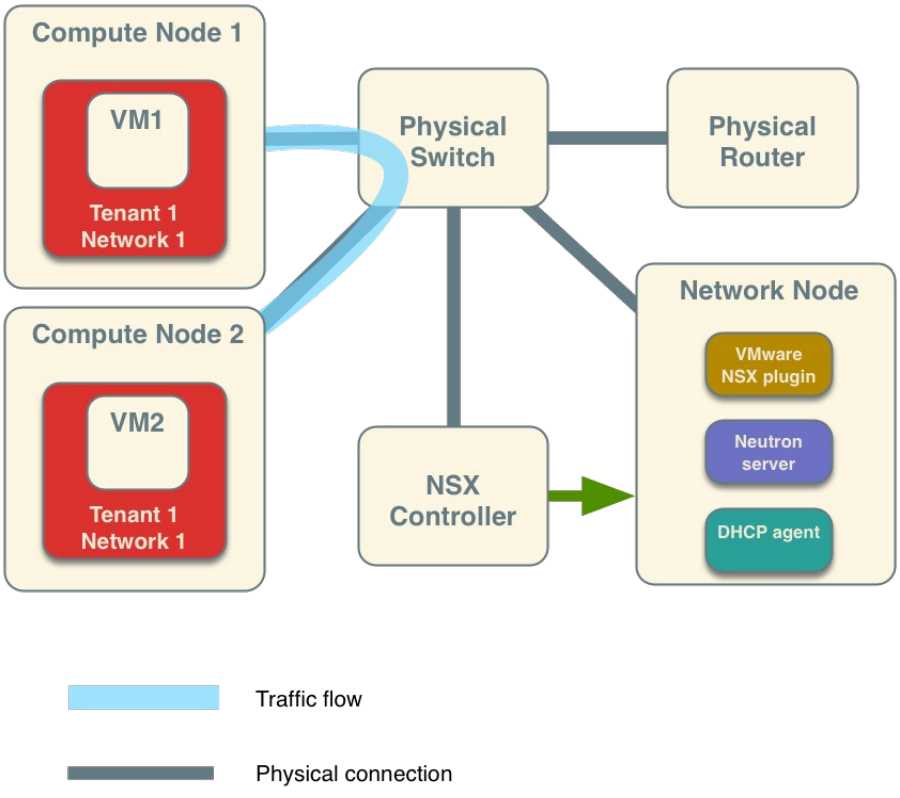
OpenStack components in a number of ways:

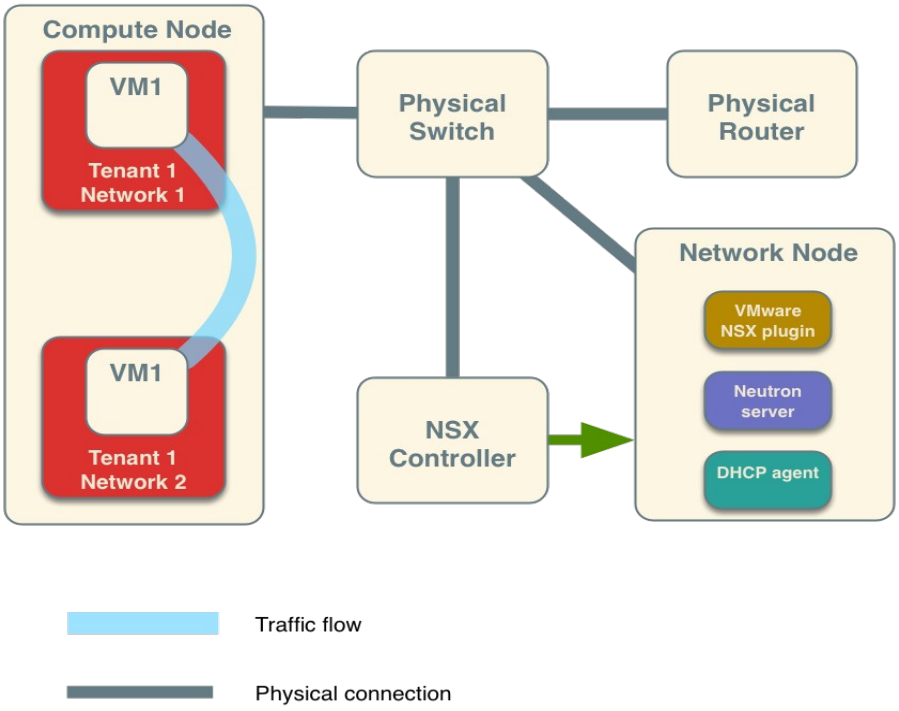
- Networking relies on the Identity service (keystone) for the authentication and authorization of all API requests.
- Compute (nova) interacts with Networking through calls to its standard API. As part of creating a VM, the `nova-compute` service communicates with the Networking API to plug each virtual NIC on the VM into a particular network.
- The dashboard (horizon) integrates with the Networking API, enabling administrators and tenant users to create and manage network services through a web-based GUI.

VMware NSX integration

OpenStack Networking uses the NSX plug-in to integrate with an existing VMware vCenter deployment. When installed on the network nodes, the NSX plug-in enables a NSX controller to centrally manage configuration settings and push them to managed network nodes. Network nodes are considered managed when they are added as hypervisors to the NSX controller.

The diagrams below depict some VMware NSX deployment examples. The first diagram illustrates the traffic flow between VMs on separate Compute nodes, and the second diagram between two VMs on a single Compute node. Note the placement of the VMware NSX plug-in and the neutron-server service on the network node. The green arrow indicates the management relationship between the NSX controller and the network node.





Plug-in configurations

For configurations options, see Networking configuration options in Configuration Reference. These sections explain how to configure specific plug-ins.

Configure Big Switch (Floodlight REST Proxy) plug-in

1. Edit the `/etc/neutron/neutron.conf` file and add this line:

   ```
   core_plugin = bigswitch
   ```

2. In the `/etc/neutron/neutron.conf` file, set the `service_plugins` option:

   ```
   service_plugins = neutron.plugins.bigswitch.l3_router_plugin.L3RestProxy
   ```

3. Edit the `/etc/neutron/plugins/bigswitch/restproxy.ini` file for the plug-in and specify a comma-separated list of controller_ip:port pairs:

   ```
   server = CONTROLLER_IP:PORT
   ```

   For database configuration, see Install Networking Services in the Installation Guide in the OpenStack Documentation index. (The link defaults to the Ubuntu version.)

4. Restart the `neutron-server` to apply the settings:

   ```
   # service neutron-server restart
   ```

Configure Brocade plug-in

1. Install the Brocade-modified Python netconf client (ncclient) library, which is available at https://github.com/brocade/ncclient:

   ```
   $ git clone https://github.com/brocade/ncclient
   ```

2. As root, run this command:

   ```
   # cd ncclient;python setup.py install
   ```

3. Edit the `/etc/neutron/neutron.conf` file and set the following option:

   ```
   core_plugin = brocade
   ```

4. Edit the `/etc/neutron/plugins/brocade/brocade.ini` file for the Brocade plug-in and specify the admin user name, password, and IP address of the Brocade switch:

   ```
   [SWITCH]
   username = ADMIN
   password = PASSWORD
   address  = SWITCH_MGMT_IP_ADDRESS
   ostype   = NOS
   ```

   For database configuration, see Install Networking Services in any of the Installation Guides in the OpenStack Documentation index. (The link defaults to the Ubuntu version.)

5. Restart the `neutron-server` service to apply the settings:

   ```
   # service neutron-server restart
   ```

Configure NSX-mh plug-in

The instructions in this section refer to the VMware NSX-mh platform, formerly known as Nicira NVP.

1. Install the NSX plug-in:

   ```
   # apt-get install neutron-plugin-vmware
   ```

2. Edit the `/etc/neutron/neutron.conf` file and set this line:

   ```
   core_plugin = vmware
   ```

   Example `neutron.conf` file for NSX-mh integration:

   ```
   core_plugin = vmware
   ```

```
rabbit_host = 192.168.203.10
allow_overlapping_ips = True
```

3. To configure the NSX-mh controller cluster for OpenStack Networking, locate the `[default]` section in the `/etc/neutron/plugins/vmware/nsx.ini` file and add the following entries:

   - To establish and configure the connection with the controller cluster you must set some parameters, including NSX-mh API endpoints, access credentials, and optionally specify settings for HTTP timeouts, redirects and retries in case of connection failures:

     ```
     nsx_user = ADMIN_USER_NAME
     nsx_password = NSX_USER_PASSWORD
     http_timeout = HTTP_REQUEST_TIMEOUT # (seconds) default 75 seconds
     retries = HTTP_REQUEST_RETRIES # default 2
     redirects = HTTP_REQUEST_MAX_REDIRECTS # default 2
     nsx_controllers = API_ENDPOINT_LIST # comma-separated list
     ```

     To ensure correct operations, the `nsx_user` user must have administrator credentials on the NSX-mh platform.

     A controller API endpoint consists of the IP address and port for the controller; if you omit the port, port 443 is used. If multiple API endpoints are specified, it is up to the user to ensure that all these endpoints belong to the same controller cluster. The OpenStack Networking VMware NSX-mh plug-in does not perform this check, and results might be unpredictable.

     When you specify multiple API endpoints, the plug-in takes care of load balancing requests on the various API endpoints.

   - The UUID of the NSX-mh transport zone that should be used by default when a tenant creates a network. You can get this value from the Transport Zones page for the NSX-mh manager:

     Alternatively the transport zone identifier can be retrieved by query the NSX-mh API: `/ws.v1/transport-zone`

     ```
     default_tz_uuid = TRANSPORT_ZONE_UUID
     ```

   - `default_l3_gw_service_uuid = GATEWAY_SERVICE_UUID`

> **Warning**
>
> Ubuntu packaging currently does not update the neutron init script to point to the NSX-mh configuration file. Instead, you must manually update `/etc/default/neutron-server` to add this line:
>
> NEUTRON_PLUGIN_CONFIG = /etc/neutron/plugins/vmware/nsx.ini

For database configuration, see Install Networking Services in the Installation Guide.

4. Restart `neutron-server` to apply settings:

```
# service neutron-server restart
```

> **Warning**
>
> The neutron NSX-mh plug-in does not implement initial re-synchronization of Neutron resources. Therefore resources that might already exist in the database when Neutron is switched to the NSX-mh plug-in will not be created on the NSX-mh backend upon restart.

Example `nsx.ini` file:

```
[DEFAULT]
default_tz_uuid = d3afb164-b263-4aaa-a3e4-48e0e09bb33c
default_l3_gw_service_uuid=5c8622cc-240a-40a1-9693-e6a5fca4e3cf
nsx_user=admin
nsx_password=changeme
nsx_controllers=10.127.0.100,10.127.0.200:8888
```

> **Note**
>
> To debug `nsx.ini` configuration issues, run this command from the host that runs neutron-server:

```
# neutron-check-nsx-config PATH_TO_NSX.INI
```

This command tests whether `neutron-server` can log into all of the NSX-mh controllers and the SQL server, and whether all UUID values are correct.

Configure PLUMgrid plug-in

1. Edit the `/etc/neutron/neutron.conf` file and set this line:

   ```
   core_plugin = plumgrid
   ```

2. Edit the [PLUMgridDirector] section in the `/etc/neutron/plugins/plumgrid/plumgrid.ini` file and specify the IP address, port, admin user name, and password of the PLUMgrid Director:

   ```
   [PLUMgridDirector]
   director_server = "PLUMgrid-director-ip-address"
   director_server_port = "PLUMgrid-director-port"
   username = "PLUMgrid-director-admin-username"
   password = "PLUMgrid-director-admin-password"
   ```

   For database configuration, see Install Networking Services in the Installation Guide.

3. Restart the `neutron-server` service to apply the settings:

   ```
   # service neutron-server restart
   ```

Configure neutron agents

Plug-ins typically have requirements for particular software that must be run on each node that handles data packets. This includes any node that runs nova-compute and nodes that run dedicated OpenStack Networking service agents such as `neutron-dhcp-agent`, `neutron-l3-agent`, `neutron-metering-agent` or `neutron-lbaas-agent`.

A data-forwarding node typically has a network interface with an IP address on the management network and another interface on the data network.

This section shows you how to install and configure a subset of the available plug-ins, which might include the installation of switching software (for example, `Open vSwitch`) and as agents used to communicate with the `neutron-server` process running elsewhere in the data center.

Configure data-forwarding nodes

Node set up: NSX plug-in

If you use the NSX plug-in, you must also install Open vSwitch on each data-forwarding node. However, you do not need to install an additional agent on each node.

> **Warning**
>
> It is critical that you run an Open vSwitch version that is compatible with the current version of the NSX Controller software. Do not use the Open vSwitch version that is installed by default on Ubuntu. Instead, use the Open vSwitch version that is provided on the VMware support portal for your NSX Controller version.

To set up each node for the NSX plug-in

1. Ensure that each data-forwarding node has an IP address on the management network, and an IP address on the data network that is used for tunneling data traffic. For full details on configuring your forwarding node, see the `NSX Administrator Guide`.
2. Use the `NSX Administrator Guide` to add the node as a Hypervisor by using the NSX Manager GUI. Even if your forwarding node has no VMs and is only used for services agents like `neutron-dhcp-agent` or `neutron-lbaas-agent`, it should still be added to NSX as a Hypervisor.
3. After following the NSX Administrator Guide, use the page for this Hypervisor in the NSX Manager GUI to confirm that the node is properly connected to the NSX Controller Cluster and that the NSX Controller Cluster can see the `br-int` integration bridge.

Configure DHCP agent

The DHCP service agent is compatible with all existing plug-ins and is required for all deployments where VMs should automatically receive IP addresses through DHCP.

To install and configure the DHCP agent

1. You must configure the host running the neutron-dhcp-agent as a data forwarding node according to the requirements for your plug-in.
2. Install the DHCP agent:

   ```
   # apt-get install neutron-dhcp-agent
   ```

3. Update any options in the `/etc/neutron/dhcp_agent.ini` file that depend on the plug-in in use. See the sub-sections.

> **Important**
>
> If you reboot a node that runs the DHCP agent, you must run the **neutron-ovs-cleanup** command before the `neutron-dhcp-agent` service starts.
>
> On Red Hat, SUSE, and Ubuntu based systems, the `neutron-ovs-cleanup` service runs the **neutron-ovs-cleanup** command automatically. However, on Debian-based systems, you must manually run this command or write your own system script that runs on boot before the `neutron-dhcp-agent` service starts.

Networking dhcp-agent can use dnsmasq driver which supports stateful and stateless DHCPv6 for subnets created with `--ipv6_address_mode` set to `dhcpv6-stateful` or `dhcpv6-stateless`.

For example:

```
$ neutron subnet-create --ip-version 6 --ipv6_ra_mode dhcpv6-stateful \
  --ipv6_address_mode dhcpv6-stateful NETWORK CIDR

$ neutron subnet-create --ip-version 6 --ipv6_ra_mode dhcpv6-stateless \
  --ipv6_address_mode dhcpv6-stateless NETWORK CIDR
```

If no dnsmasq process for subnet's network is launched, Networking will launch a new one on subnet's dhcp port in `qdhcp-XXX` namespace. If previous dnsmasq process is already launched, restart dnsmasq with a new configuration.

Networking will update dnsmasq process and restart it when subnet gets updated.

> **Note**
>
> For dhcp-agent to operate in IPv6 mode use at least dnsmasq v2.63.

After a certain, configured timeframe, networks uncouple from DHCP agents when the agents are no longer in use. You can configure the DHCP agent to automatically detach from a network when the agent is out of service, or no longer needed.

This feature applies to all plug-ins that support DHCP scaling. For more information, see the DHCP agent configuration options listed in the OpenStack Configuration Reference.

DHCP agent setup: OVS plug-in

These DHCP agent options are required in the `/etc/neutron/dhcp_agent.ini` file for the OVS plug-in:

```
[DEFAULT]
enable_isolated_metadata = True
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
```

DHCP agent setup: NSX plug-in

These DHCP agent options are required in the `/etc/neutron/dhcp_agent.ini` file for the NSX plug-in:

```
[DEFAULT]
enable_metadata_network = True
enable_isolated_metadata = True
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
```

Configure L3 agent

The OpenStack Networking service has a widely used API extension to allow administrators and tenants to create routers to interconnect L2 networks, and floating IPs to make ports on private networks publicly accessible.

Many plug-ins rely on the L3 service agent to implement the L3 functionality. However, the following plug-ins already have built-in L3 capabilities:

- Big Switch/Floodlight plug-in, which supports both the open source Floodlight controller and the proprietary Big Switch controller.

  > **Note**
  >
  > Only the proprietary BigSwitch controller implements L3 functionality. When using Floodlight as your OpenFlow controller, L3 functionality is not available.

- IBM SDN-VE plug-in
- MidoNet plug-in
- NSX plug-in
- PLUMgrid plug-in

> **Warning**
>
> Do not configure or use `neutron-l3-agent` if you use one of these plug-ins.

To install the L3 agent for all other plug-ins

1. Install the `neutron-l3-agent` binary on the network node:

   ```
   # apt-get install neutron-l3-agent
   ```

2. To uplink the node that runs `neutron-l3-agent` to the external network, create a bridge named `br-ex` and attach the NIC for the external network to this bridge.

   For example, with Open vSwitch and NIC eth1 connected to the external network, run:

   ```
   # ovs-vsctl add-br br-ex
   # ovs-vsctl add-port br-ex eth1
   ```

   When the `br-ex` port is added to the `eth1` interface, external communication is interrupted. To avoid this, edit the `/etc/network/interfaces` file to contain the following information:

   ```
   ## External bridge
   auto br-ex
   iface br-ex inet static
   address 192.27.117.101
   netmask 255.255.240.0
   gateway 192.27.127.254
   dns-nameservers 8.8.8.8

   ## External network interface
   auto eth1
   iface eth1 inet manual
   up ifconfig $IFACE 0.0.0.0 up
   up ip link set $IFACE promisc on
   down ip link set $IFACE promisc off
   down ifconfig $IFACE down
   ```

> **Note**
>
> The external bridge configuration address is the external IP address. This address and gateway should be configured in `/etc/network/interfaces`.

After editing the configuration, restart `br-ex`:

```
# ifdown br-ex && ifup br-ex
```

Do not manually configure an IP address on the NIC connected to the external network for the node running `neutron-l3-agent`. Rather, you must have a range of IP addresses from the external network that can be used by OpenStack Networking for routers that uplink to the external network. This range must be large enough to have an IP address for each router in the deployment, as well as each floating IP.

3. The `neutron-l3-agent` uses the Linux IP stack and iptables to perform L3 forwarding and NAT. In order to support multiple routers with potentially overlapping IP addresses, `neutron-l3-agent` defaults to using Linux network namespaces to provide isolated forwarding contexts. As a result, the IP addresses of routers are not visible simply by running the **ip addr list** or **ifconfig** command on the node. Similarly, you cannot directly **ping** fixed IPs.

   To do either of these things, you must run the command within a particular network namespace for the router. The namespace has the name `qrouter-ROUTER_UUID`. These example commands run in the router namespace with UUID 47af3868-0fa8-4447-85f6-1304de32153b:

   ```
   # ip netns exec qrouter-47af3868-0fa8-4447-85f6-1304de32153b ip addr list
   ```

   ```
   # ip netns exec qrouter-47af3868-0fa8-4447-85f6-1304de32153b ping FIXED_IP
   ```

> **Important**
>
> If you reboot a node that runs the L3 agent, you must run the **neutron-ovs-cleanup** command before the `neutron-l3-agent` service starts.
>
> On Red Hat, SUSE and Ubuntu based systems, the neutron-ovs-cleanup service runs the **neutron-ovs-cleanup** command automatically. However, on Debian-based systems, you must manually run this command or write your own system script that runs on boot before the neutron-l3-agent service starts.

How routers are assigned to L3 agents By default, a router is assigned to the L3 agent with the least number of routers (LeastRoutersScheduler). This can be changed by altering the `router_scheduler_driver` setting in the configuration file.

Configure metering agent

The Neutron Metering agent resides beside neutron-l3-agent.

To install the metering agent and configure the node

1. Install the agent by running:

   ```
   # apt-get install neutron-metering-agent
   ```

2. If you use one of the following plug-ins, you need to configure the metering agent with these lines as well:

   - An OVS-based plug-in such as OVS, NSX, NEC, BigSwitch/Floodlight:

     ```
     interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
     ```

   - A plug-in that uses LinuxBridge:

     ```
     interface_driver = neutron.agent.linux.interface.
     BridgeInterfaceDriver
     ```

3. To use the reference implementation, you must set:

   ```
   driver = neutron.services.metering.drivers.iptables.iptables_driver
   .IptablesMeteringDriver
   ```

4. Set the `service_plugins` option in the `/etc/neutron/neutron.conf` file on the host that runs `neutron-server`:

   ```
   service_plugins = metering
   ```

   If this option is already defined, add `metering` to the list, using a comma as separator. For example:

   ```
   service_plugins = router,metering
   ```

Configure Load-Balancer-as-a-Service (LBaaS v2)

For the back end, use either Octavia or Haproxy. This example uses Octavia.

To configure LBaaS V2

1. Install Octavia using your distribution's package manager.
2. Edit the `/etc/neutron/neutron_lbaas.conf` file and change the `service_provider` parameter to enable Octavia:

   ```
   service_provider = LOADBALANCERV2:Octavia:neutron_lbaas.
   drivers.octavia.driver.OctaviaDriver:default
   ```

3. Edit the `/etc/neutron/neutron.conf` file and add the `service_plugins` parameter to enable the load-balancing plug-in:

```
service_plugins =
neutron_lbaas.services.loadbalancer.plugin.LoadBalancerPluginv2
```

If this option is already defined, add the load-balancing plug-in to the list using a comma as a separator. For example:

```
service_plugins = [already defined
plugins],neutron_lbaas.services.loadbalancer.plugin.LoadBalancerPluginv2
```

4. Create the required tables in the database:

```
# neutron-db-manage --subproject neutron-lbaas upgrade head
```

5. Restart the `neutron-server` service.

6. Enable load balancing in the Project section of the dashboard.

> **Warning**
>
> Horizon panels are enabled only for LBaaSV1. LBaaSV2 panels are still being developed.

By default, the `enable_lb` option is `True` in the local_settings.py file.

```
OPENSTACK_NEUTRON_NETWORK = {
    'enable_lb': True,
    ...
}
```

Apply the settings by restarting the web server. You can now view the Load Balancer management options in the Project view in the dashboard.

Configure Hyper-V L2 agent

Before you install the OpenStack Networking Hyper-V L2 agent on a Hyper-V compute node, ensure the compute node has been configured correctly using these instructions.

To install the OpenStack Networking Hyper-V agent and configure the node

1. Download the OpenStack Networking code from the repository:

```
> cd C:\OpenStack\
> git clone https://git.openstack.org/openstack/neutron
```

2. Install the OpenStack Networking Hyper-V Agent:

```
> cd C:\OpenStack\neutron\
> python setup.py install
```

3. Copy the `policy.json` file:

```
> xcopy C:\OpenStack\neutron\etc\policy.json C:\etc\
```

4. Create the `C:\etc\neutron-hyperv-agent.conf` file and add the proper configuration options and the Hyper-V related options. Here is a sample config file:

```
[DEFAULT]
control_exchange = neutron
policy_file = C:\etc\policy.json
rpc_backend = neutron.openstack.common.rpc.impl_kombu
rabbit_host = IP_ADDRESS
rabbit_port = 5672
rabbit_userid = guest
rabbit_password = <password>
logdir = C:\OpenStack\Log
logfile = neutron-hyperv-agent.log

[AGENT]
polling_interval = 2
physical_network_vswitch_mappings = *:YOUR_BRIDGE_NAME
enable_metrics_collection = true

[SECURITYGROUP]
firewall_driver = neutron.plugins.hyperv.agent.security_groups_driver.
HyperVSecurityGroupsDriver
enable_security_group = true
```

5. Start the OpenStack Networking Hyper-V agent:

```
> C:\Python27\Scripts\neutron-hyperv-agent.exe --config-file
C:\etc\neutron-hyperv-agent.conf
```

Basic operations on agents

This table shows examples of Networking commands that enable you to complete basic operations on agents:

Operation	Command
List all available agents.	`$ neutron agent-list`
Show information of a given agent.	`$ neutron agent-show AGENT_ID`
Update the admin status and description for a specified agent. The	`$ neutron agent-update --admin -state-up False`

Operation	Command
command can be used to enable and disable agents by using *--admin-state-up* parameter set to `False` or `True`.	`AGENT_ID`
Delete a given agent. Consider disabling the agent before deletion.	`$ neutron agent-delete AGENT_ID`

Basic operations on Networking agents

See the OpenStack Command-Line Interface Reference for more information on Networking commands.

Configure Identity service for Networking

To configure the Identity service for use with Networking

1. Create the `get_id()` function

   The `get_id()` function stores the ID of created objects, and removes the need to copy and paste object IDs in later steps:

   1. Add the following function to your `.bashrc` file:

      ```
      function get_id () {
      echo `"$@" | awk '/ id / { print $4 }'`
      }
      ```

   2. Source the `.bashrc` file:

      ```
      $ source .bashrc
      ```

2. Create the Networking service entry

   Networking must be available in the Compute service catalog. Create the service:

   ```
   $ NEUTRON_SERVICE_ID=$(get_id openstack service create network \
     --name neutron --description 'OpenStack Networking Service')
   ```

3. Create the Networking service endpoint entry

   The way that you create a Networking endpoint entry depends on whether you are using the SQL or the template catalog driver:

   1. If you are using the `SQL driver`, run the following command with the specified region (`$REGION`), IP address of the Networking server (`$IP`), and service ID (`$NEUTRON_SERVICE_ID`, obtained in the previous step).

```
$ openstack endpoint create $NEUTRON_SERVICE_ID --region $REGION \
  --publicurl 'http://$IP:9696/' --adminurl 'http://$IP:9696/' \
  --internalurl 'http://$IP:9696/'
```

For example:

```
$ openstack endpoint create $NEUTRON_SERVICE_ID --region myregion \
  --publicurl "http://10.211.55.17:9696/" \
  --adminurl "http://10.211.55.17:9696/" \
  --internalurl "http://10.211.55.17:9696/"
```

2. If you are using the `template driver`, specify the following parameters in your Compute catalog template file (`default_catalog.templates`), along with the region (`$REGION`) and IP address of the Networking server (`$IP`).

```
catalog.$REGION.network.publicURL = http://$IP:9696
catalog.$REGION.network.adminURL = http://$IP:9696
catalog.$REGION.network.internalURL = http://$IP:9696
catalog.$REGION.network.name = Network Service
```

For example:

```
catalog.$Region.network.publicURL = http://10.211.55.17:9696
catalog.$Region.network.adminURL = http://10.211.55.17:9696
catalog.$Region.network.internalURL = http://10.211.55.17:9696
catalog.$Region.network.name = Network Service
```

4. Create the Networking service user

You must provide admin user credentials that Compute and some internal Networking components can use to access the Networking API. Create a special `service` tenant and a `neutron` user within this tenant, and assign an `admin` role to this role.

1. Create the `admin` role:

```
$ ADMIN_ROLE=$(get_id openstack role create admin)
```

2. Create the `neutron` user:

```
$ NEUTRON_USER=$(get_id openstack user create neutron \
  --password "$NEUTRON_PASSWORD" --email demo@example.com \
  --project service)
```

3. Create the `service` tenant:

```
$ SERVICE_TENANT=$(get_id openstack project create service \
  --description "Services project")
```

4. Establish the relationship among the tenant, user, and role:

```
$ openstack role add $ADMIN_ROLE --user $NEUTRON_USER \
  --project $SERVICE_TENANT
```

For information about how to create service entries and users, see the OpenStack Installation Guide for your distribution (docs.openstack.org).

Compute

If you use Networking, do not run the Compute `nova-network` service (like you do in traditional Compute deployments). Instead, Compute delegates most network-related decisions to Networking.

> **Note**
>
> Uninstall `nova-network` and reboot any physical nodes that have been running `nova-network` before using them to run Networking. Inadvertently running the `nova-network` process while using Networking can cause problems, as can stale iptables rules pushed down by previously running `nova-network`.

Compute proxies tenant-facing API calls to manage security groups and floating IPs to Networking APIs. However, operator-facing tools such as `nova-manage`, are not proxied and should not be used.

> **Warning**
>
> When you configure networking, you must use this guide. Do not rely on Compute networking documentation or past experience with Compute. If a **nova** command or configuration option related to networking is not mentioned in this guide, the command is probably not supported for use with Networking. In particular, you cannot use CLI tools like `nova-manage` and `nova` to manage networks or IP addressing, including both fixed and floating IPs, with Networking.

To ensure that Compute works properly with Networking (rather than the legacy `nova-network` mechanism), you must adjust settings in the `nova.conf` configuration file.

Networking API and credential configuration

Each time you provision or de-provision a VM in Compute, `nova-\*` services communicate with Networking using the standard API. For this to happen, you must configure the following items in the `nova.conf` file (used by each `nova-compute` and `nova-api` instance).

nova.conf API and credential settings

Attribute name	Required
`[DEFAULT] use_neutron`	Modify from the default to `True` to indicate that Networking should be used rather than the traditional nova-network networking model.
`[neutron] url`	Update to the host name/IP and port of the neutron-server instance for this deployment.
`[neutron] auth_strategy`	Keep the default `keystone` value for all production deployments.
`[neutron] admin_tenant_name`	Update to the name of the service tenant created in the above section on Identity configuration.
`[neutron] admin_username`	Update to the name of the user created in the above section on Identity configuration.
`[neutron] admin_password`	Update to the password of the user created in the above section on Identity configuration.
`[neutron] admin_auth_url`	Update to the Identity server IP and port. This is the Identity (keystone) admin API server IP and port value, and not the Identity service API IP and port.

Configure security groups

The Networking service provides security group functionality using a mechanism that is more flexible and powerful than the security group capabilities built into Compute. Therefore, if you use Networking, you should always disable built-in security groups and proxy all security group calls to the Networking API. If you do not, security policies will conflict by being simultaneously applied by both services.

To proxy security groups to Networking, use the following configuration values in the `nova.conf` file:

nova.conf security group settings

Item	Configuration
`firewall_driver`	Update to `nova.virt.firewall.NoopFirewallDriver`, so that nova-compute does not perform iptables-based filtering itself.

Configure metadata

The Compute service allows VMs to query metadata associated with a VM by making a web request to a special 169.254.169.254 address. Networking supports proxying those requests to nova-api, even when the requests are made from isolated networks, or from multiple networks that use overlapping IP addresses.

To enable proxying the requests, you must update the following fields in `[neutron]` section in the `nova.conf`.

nova.conf metadata settings

Item	Configuration
`service_metadata_proxy`	Update to `true`, otherwise nova-api will not properly respond to requests from the neutron-metadata-agent.
`metadata_proxy_shared_secret`	Update to a string “password” value. You must also configure the same value in the `metadata_agent.ini` file, to authenticate requests made for metadata. The default value of an empty string in both files will allow metadata to function, but will not be secure if any non-trusted entities have access to the metadata APIs exposed by nova-api.

> **Note**
>
> As a precaution, even when using `metadata_proxy_shared_secret`, we recommend that you do not expose metadata using the same nova-api instances that are used for tenants. Instead, you should run a dedicated set of nova-api instances for metadata that are available only on your management network. Whether a given nova-api instance exposes metadata APIs is determined by the value of `enabled_apis` in its `nova.conf`.

Example nova.conf (for nova-compute and nova-api)

Example values for the above settings, assuming a cloud controller node running Compute and Networking with an IP address of 192.168.1.2:

```
[DEFAULT]
use_neutron = True
firewall_driver=nova.virt.firewall.NoopFirewallDriver

[neutron]
url=http://192.168.1.2:9696
auth_strategy=keystone
admin_tenant_name=service
admin_username=neutron
admin_password=password
admin_auth_url=http://192.168.1.2:35357/v2.0
service_metadata_proxy=true
metadata_proxy_shared_secret=foo
```

Advanced configuration options

This section describes advanced configuration options for various system components. For example, configuration options where the default works but that the user wants to customize options. After installing from packages, `$NEUTRON_CONF_DIR` is `/etc/neutron`.

L3 metering agent

You can run an L3 metering agent that enables layer-3 traffic metering. In general, you should launch the metering agent on all nodes that run the L3 agent:

```
$ neutron-metering-agent --config-file NEUTRON_CONFIG_FILE \
  --config-file L3_METERING_CONFIG_FILE
```

You must configure a driver that matches the plug-in that runs on the service. The driver adds metering to the routing interface.

Option	Value
Open vSwitch	
interface_driver ($NEUTRON_CONF_DIR/metering_agent.ini)	neutron.agent.linux.interface. OVSInterfaceDriver
Linux Bridge	
interface_driver ($NEUTRON_CONF_DIR/metering_agent.ini)	neutron.agent.linux.interface. BridgeInterfaceDriver

L3 metering driver

You must configure any driver that implements the metering abstraction. Currently the only available implementation uses iptables for metering.

```
driver = neutron.services.metering.drivers.
iptables.iptables_driver.IptablesMeteringDriver
```

L3 metering service driver

To enable L3 metering, you must set the following option in the `neutron.conf` file on the host that runs `neutron-server`:

```
service_plugins = metering
```

Scalable and highly available DHCP agents

This section is fully described in the Networking Guide.

Use Networking

You can manage OpenStack Networking services by using the service command. For example:

```
# service neutron-server stop
# service neutron-server status
# service neutron-server start
# service neutron-server restart
```

Log files are in the `/var/log/neutron` directory.

Configuration files are in the `/etc/neutron` directory.

Administrators and tenants can use OpenStack Networking to build rich network topologies. Administrators can create network connectivity on behalf of tenants.

Core Networking API features

After you install and configure Networking, tenants and administrators can perform create-read-update-delete (CRUD) API networking operations by using the Networking API directly or neutron command-line interface (CLI). The neutron CLI is a wrapper around the Networking API. Every Networking API call has a corresponding neutron command.

The CLI includes a number of options. For details, see the OpenStack End User Guide.

Basic Networking operations

To learn about advanced capabilities available through the neutron command-line interface (CLI), read the networking section in the OpenStack End User Guide.

This table shows example neutron commands that enable you to complete basic network operations:

Operation	Command
Creates a network.	`$ neutron net-create net1`
Creates a subnet that is associated with net1.	`$ neutron subnet-create net1 10.0.0.0/24`
Lists ports for a specified tenant.	`$ neutron port-list`
Lists ports for a specified tenant and displays the `id`, `fixed_ips`, and `device_owner` columns.	`$ neutron port-list -c id -c fixed_ips -c device_owner`
Shows information for a specified port.	`$ neutron port-show PORT_ID`

Basic Networking operations

> **Note**
>
> The `device_owner` field describes who owns the port. A port whose `device_owner` begins with:
>
> - `network` is created by Networking.
> - `compute` is created by Compute.

Administrative operations

The administrator can run any **neutron** command on behalf of tenants by specifying an Identity `tenant_id` in the command, as follows:

```
$ neutron net-create --tenant-id TENANT_ID NETWORK_NAME
```

For example:

```
$ neutron net-create --tenant-id 5e4bbe24b67a4410bc4d9fae29ec394e net1
```

> **Note**
>
> To view all tenant IDs in Identity, run the following command as an Identity service admin user:
>
> `$ openstack project list`

Advanced Networking operations

This table shows example Networking commands that enable you to complete advanced network operations:

Operation	Command
Creates a network that all tenants can use.	`$ neutron net-create --shared public-net`
Creates a subnet with a specified gateway IP address.	`$ neutron subnet-create --gateway 10.0.0.254 net1 10.0.0.0/24`
Creates a subnet that has	`$ neutron subnet-create --no-gateway`

Operation	Command
no gateway IP address.	`net1 10.0.0.0/24`
Creates a subnet with DHCP disabled.	`$ neutron subnet-create net1 10.0.0.0/24 --enable-dhcp=False`
Specifies a set of host routes	`$ neutron subnet-create test-net1 40.0.0.0/24 --host-routes type=dict list=true destination=40.0.1.0/24, nexthop=40.0.0.2`
Creates a subnet with a specified set of dns name servers.	`$ neutron subnet-create test-net1 40.0.0.0/24 --dns-nameservers list=true 8.8.4.4 8.8.8.8`
Displays all ports and IPs allocated on a network.	`$ neutron port-list --network_id NET_ID`

Advanced Networking operations

Use Compute with Networking

Basic Compute and Networking operations

This table shows example neutron and nova commands that enable you to complete basic VM networking operations:

Action	Command
Checks available networks.	`$ neutron net-list`
Boots a VM with a single NIC on a selected Networking network.	`$ nova boot --image IMAGE --flavor FLAVOR --nic net-id=NET_ID VM_NAME`
Searches for ports with a `device_id` that matches the Compute instance UUID. See :ref: Create and delete VMs	`$ neutron port-list --device_id VM_ID`
Searches for ports, but shows only the `mac_address` of the port.	`$ neutron port-list --field mac_address --device_id VM_ID`
Temporarily disables a port from sending traffic.	`$ neutron port-update PORT_ID --admin_state_up False`

Basic Compute and Networking operations

> **Note**
>
> The `device_id` can also be a logical router ID.
>
> **Note**
>
> When you boot a Compute VM, a port on the network that corresponds to the VM NIC is automatically created and associated with the default security group. You can configure security group rules to enable users to access the VM.

Advanced VM creation operations

This table shows example nova and neutron commands that enable you to complete advanced VM creation operations:

Operation	Command
Boots a VM with multiple NICs.	`$ nova boot --image IMAGE --flavor FLAVOR --nic net-id=NET1-ID --nic net-id=NET2-ID VM_NAME`
Boots a VM with a specific IP address. Note that you cannot use the `--num-instances` parameter in this case.	`$ nova boot --image IMAGE --flavor FLAVOR --nic net-id=NET-ID, v4-fixed-ip=IP-ADDR VM_NAME`
Boots a VM that connects to all networks that are accessible to the tenant who submits the request (without the `--nic` option).	`$ nova boot --image IMAGE --flavor FLAVOR VM_NAME`

Advanced VM creation operations

> **Note**
>
> Cloud images that distribution vendors offer usually have only one active NIC configured. When you boot with multiple NICs, you must configure additional interfaces on the image or the NICs are not reachable.
>
> The following Debian/Ubuntu-based example shows how to set up the interfaces within the instance in the `/etc/network/interfaces` file. You must apply this configuration to the image.
>
> *# The loopback network interface* auto lo iface lo inet loopback auto eth0 iface eth0 inet dhcp auto eth1 iface eth1 inet dhcp

Enable ping and SSH on VMs (security groups)

You must configure security group rules depending on the type of plug-in you are using. If you are using a plug-in that:

- Implements Networking security groups, you can configure security group rules directly by using the **neutron security-group-rule-create** command. This example enables `ping` and `ssh` access to your VMs.

```
$ neutron security-group-rule-create --protocol icmp \
    --direction ingress default

$ neutron security-group-rule-create --protocol tcp --port-range-min 22 \
    --port-range-max 22 --direction ingress default
```

- Does not implement Networking security groups, you can configure security group rules by using the **nova secgroup-add-rule** or **euca-authorize** command. These **nova** commands enable `ping` and `ssh` access to your VMs.

```
$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
```

> **Note**
>
> If your plug-in implements Networking security groups, you can also leverage Compute security groups by setting `security_group_api = neutron` in the `nova.conf` file. After you set this option, all Compute security group commands are proxied to Networking.

Advanced features through API extensions

Several plug-ins implement API extensions that provide capabilities similar to what was available in `nova-network`. These plug-ins are likely to be of interest to the OpenStack community.

Provider networks

Networks can be categorized as either tenant networks or provider networks. Tenant networks are created by normal users and details about how they are physically realized are hidden from those users. Provider networks are created with administrative credentials, specifying the details of how the network is physically realized, usually to match some existing network in the data center.

Provider networks enable administrators to create networks that map directly to the physical networks in the data center. This is commonly used to give tenants direct access to a public network that can be used to reach the Internet. It might also be used to integrate with VLANs in the network that already have a defined meaning (for example, enable a VM from the marketing department to be placed on the same VLAN as bare-metal marketing hosts in the same data center).

The provider extension allows administrators to explicitly manage the relationship between Networking virtual networks and underlying physical mechanisms such as VLANs and tunnels. When this extension is supported, Networking client users with administrative privileges see additional provider attributes on all virtual networks and are able to specify these attributes in order to create provider networks.

The provider extension is supported by the Open vSwitch and Linux Bridge plug-ins. Configuration of these plug-ins requires familiarity with this extension.

Terminology

A number of terms are used in the provider extension and in the configuration of plug-ins supporting the provider extension:

Provider extension terminology

Term	Description
virtual network	A Networking L2 network (identified by a UUID and optional name) whose ports can be attached as vNICs to Compute instances and to various Networking agents. The Open vSwitch and Linux

Term	Description
	Bridge plug-ins each support several different mechanisms to realize virtual networks.
physical network	A network connecting virtualization hosts (such as compute nodes) with each other and with other network resources. Each physical network might support multiple virtual networks. The provider extension and the plug-in configurations identify physical networks using simple string names.
tenant network	A virtual network that a tenant or an administrator creates. The physical details of the network are not exposed to the tenant.
provider network	A virtual network administratively created to map to a specific network in the data center, typically to enable direct access to non-OpenStack resources on that network. Tenants can be given access to provider networks.
VLAN network	A virtual network implemented as packets on a specific physical network containing IEEE 802.1Q headers with a specific VID field value. VLAN networks sharing the same physical network are isolated from each other at L2 and can even have overlapping IP address spaces. Each distinct physical network supporting VLAN networks is treated as a separate VLAN trunk, with a distinct space of VID values. Valid VID values are 1 through 4094.
flat network	A virtual network implemented as packets on a specific physical network containing no IEEE 802.1Q header. Each physical network can realize at most one flat network.

Term	Description
local network	A virtual network that allows communication within each host, but not across a network. Local networks are intended mainly for single-node test scenarios, but can have other uses.
GRE network	A virtual network implemented as network packets encapsulated using GRE. GRE networks are also referred to as *tunnels*. GRE tunnel packets are routed by the IP routing table for the host, so GRE networks are not associated by Networking with specific physical networks.
Virtual Extensible LAN (VXLAN) network	VXLAN is a proposed encapsulation protocol for running an overlay network on existing Layer 3 infrastructure. An overlay network is a virtual network that is built on top of existing network Layer 2 and Layer 3 technologies to support elastic compute architectures.

The ML2, Open vSwitch, and Linux Bridge plug-ins support VLAN networks, flat networks, and local networks. Only the ML2 and Open vSwitch plug-ins currently support GRE and VXLAN networks, provided that the required features exist in the hosts Linux kernel, Open vSwitch, and iproute2 packages.

Provider attributes

The provider extension extends the Networking network resource with these attributes:

Provider network attributes

Attribute name	Type	Default Value	Description
provider: network_type	String	N/A	The physical mechanism by which the virtual network is implemented. Possible values are `flat`, `vlan`, `local`, `gre`, and `vxlan`, corresponding to flat networks, VLAN networks, local

Attribute name	Type	Default Value	Description
			networks, GRE networks, and VXLAN networks as defined above. All types of provider networks can be created by administrators, while tenant networks can be implemented as `vlan`, `gre`, `vxlan`, or `local` network types depending on plug-in configuration.
provider: physical_network	String	If a physical network named “default” has been configured and if provider:network_type is `flat` or `vlan`, then “default” is used.	The name of the physical network over which the virtual network is implemented for flat and VLAN networks. Not applicable to the `local` or `gre` network types.
provider:segmentation_id	Integer	N/A	For VLAN networks, the VLAN VID on the physical network that realizes the virtual network. Valid VLAN VIDs are 1 through 4094. For GRE networks, the tunnel ID. Valid tunnel IDs are any 32 bit unsigned integer. Not applicable to the `flat` or `local` network types.

To view or set provider extended attributes, a client must be authorized for the

`extension:provider_network:view` and `extension:provider_network:set` actions in the Networking policy configuration. The default Networking configuration authorizes both actions for users with the admin role. An authorized client or an administrative user can view and set the provider extended attributes through Networking API calls. See the section called *Authentication and authorization* for details on policy configuration.

L3 routing and NAT

The Networking API provides abstract L2 network segments that are decoupled from the technology used to implement the L2 network. Networking includes an API extension that provides abstract L3 routers that API users can dynamically provision and configure. These Networking routers can connect multiple L2 Networking networks and can also provide a gateway that connects one or more private L2 networks to a shared external network. For example, a public network for access to the Internet. See the OpenStack Configuration Reference for details on common models of deploying Networking L3 routers.

The L3 router provides basic NAT capabilities on gateway ports that uplink the router to external networks. This router SNATs all traffic by default and supports floating IPs, which creates a static one-to-one mapping from a public IP on the external network to a private IP on one of the other subnets attached to the router. This allows a tenant to selectively expose VMs on private networks to other hosts on the external network (and often to all hosts on the Internet). You can allocate and map floating IPs from one port to another, as needed.

Basic L3 operations

External networks are visible to all users. However, the default policy settings enable only administrative users to create, update, and delete external networks.

This table shows example neutron commands that enable you to complete basic L3 operations:

Basic L3 Operations

Operation	Command
Creates external networks.	`# neutron net-create public --router:external True` `$ neutron subnet-create public 172.16.1.0/24`
Lists external networks.	`$ neutron net-list --router:external True`

Creates an internal-only router that connects to	`$ neutron net-create net1`
	`$ neutron subnet-create net1 10.0.0.0/24`
	`$ neutron net-create net2`
	`$ neutron subnet-create net2 10.0.1.0/24`

Operation	Command

multiple L2 networks privately.	`$ neutron router-create router1`
	`$ neutron router-interface-add router1 SUBNET1_UUID`
	`$ neutron router-interface-add router1 SUBNET2_UUID`
	
	An internal router port can have only one IPv4 subnet and multiple IPv6 subnets that belong to the same network ID. When you call `router-interface-add` with an IPv6 subnet, this operation adds the interface to an existing internal port with the same network ID. If a port with the same network ID does not exist, a new port is created.

Connects a router to an external network, which enables that router to act as a NAT gateway for external connectivity.	`$ neutron router-gateway-set router1 EXT_NET_ID` The router obtains an interface with the gateway_ip address of the subnet and this interface is attached to a port on the L2 Networking network associated with the subnet. The router also gets a gateway interface to the specified external network. This provides SNAT connectivity to the external network as well as support for floating IPs allocated on that external networks. Commonly an external network maps to a network in the provider.
Lists routers.	`$ neutron router-list`
Shows information for a specified router.	`$ neutron router-show ROUTER_ID`
Shows all internal interfaces for a router.	`$ neutron router-port-list ROUTER_ID` `$ neutron router-port-list ROUTER_NAME`
Identifies the PORT_ID that represents the VM NIC to which the floating IP should map.	`$ neutron port-list -c id -c fixed_ips --device_id INSTANCE_ID` This port must be on a Networking subnet that is attached to a router uplinked to the external network used to create the floating IP. Conceptually, this is because the router must be able to perform the Destination NAT (DNAT) rewriting of packets from the floating IP address (chosen from a subnet on the external network) to the internal fixed IP (chosen from a private subnet that is behind the router).
Creates a floating IP address and associates it	`$ neutron floatingip-create EXT_NET_ID` `$ neutron floatingip-associate FLOATING_IP_ID INTERNAL_VM_PORT_ID`

Operation	Command
with a port.	
Creates a floating IP on a specific subnet in the external network.	`$ neutron floatingip-create EXT_NET_ID SUBNET_ID` If there are multiple subnets in the external network, you can choose a specific subnet based on quality and costs.
Creates a floating IP address and associates it with a port, in a single step.	`$ neutron floatingip-create --port_id INTERNAL_VM_PORT_ID EXT_NET_ID`
Lists floating IPs	`$ neutron floatingip-list`
Finds floating IP for a specified VM port.	`$ neutron floatingip-list --port_id INTERNAL_VM_PORT_ID`
Disassociates a floating IP address.	`$ neutron floatingip-disassociate FLOATING_IP_ID`
Deletes the floating IP address.	`$ neutron floatingip-delete FLOATING_IP_ID`
Clears the gateway.	`$ neutron router-gateway-clear router1`
Removes the interfaces from the router.	`$ neutron router-interface-delete router1 SUBNET_ID` If this subnet ID is the last subnet on the port, this operation deletes the port itself.
Deletes the router.	`$ neutron router-delete router1`

Security groups

Security groups and security group rules allow administrators and tenants to specify the type of traffic and direction (ingress/egress) that is allowed to pass through a port. A security group is a container for security group rules.

When a port is created in Networking it is associated with a security group. If a security group is not specified the port is associated with a 'default' security group. By default, this group drops all ingress traffic and allows all egress. Rules can be added to this group in order to change the behavior.

To use the Compute security group APIs or use Compute to orchestrate the creation of ports for instances on specific security groups, you must complete additional configuration. You must configure the `/etc/nova/nova.conf` file and set the `security_group_api=neutron` option on every node that runs nova-compute and nova-api. After you make this change, restart nova-api and nova-compute to pick up this change. Then, you can use both the Compute and OpenStack Network security group APIs at the same time.

> **Note**
>
> - To use the Compute security group API with Networking, the Networking plug-in must implement the security group API. The following plug-ins currently implement this: ML2, Open vSwitch, Linux Bridge, NEC, and VMware NSX.
>
> - You must configure the correct firewall driver in the `securitygroup` section of the plug-in/agent configuration file. Some plug-ins and agents, such as Linux Bridge Agent and Open vSwitch Agent, use the no-operation driver as the default, which results in non-working security groups.
>
> - When using the security group API through Compute, security groups are applied to all ports on an instance. The reason for this is that Compute security group APIs are instances based and not port based as Networking.

Basic security group operations

This table shows example neutron commands that enable you to complete basic security group operations:

Basic security group operations

Operation	Command
Creates a security group for our web servers.	`$ neutron security-group-create webservers \` `  --description "security group for webservers"`

Operation	Command
Lists security groups.	`$ neutron security-group-list`
Creates a security group rule to allow port 80 ingress.	`$ neutron security-group-rule-create --direction ingress \ --protocol tcp --port_range_min 80 --port_range_max 80 \ SECURITY_GROUP_UUID`
Lists security group rules.	`$ neutron security-group-rule-list`
Deletes a security group rule.	`$ neutron security-group-rule-delete SECURITY_GROUP_RULE_UUID`
Deletes a security group.	`$ neutron security-group-delete SECURITY_GROUP_UUID`
Creates a port and associates two security groups.	`$ neutron port-create --security-group SECURITY_GROUP_ID1 \ --security-group SECURITY_GROUP_ID2 NETWORK_ID`
Removes security groups from a port.	`$ neutron port-update --no-security-groups PORT_ID`

Basic Load-Balancer-as-a-Service operations

> **Note**
>
> The Load-Balancer-as-a-Service (LBaaS) API provisions and configures load balancers. The reference implementation is based on the HAProxy software load balancer.

This list shows example neutron commands that enable you to complete basic LBaaS operations:

- Creates a load balancer pool by using specific provider.

  --provider is an optional argument. If not used, the pool is created with default provider for LBaaS service. You should configure the default provider in the `[service_providers]` section of the `neutron.conf` file. If no default provider is specified for LBaaS, the *--provider* parameter is required for pool creation.

  ```
  $ neutron lb-pool-create --lb-method ROUND_ROBIN --name mypool \
    --protocol HTTP --subnet-id SUBNET_UUID --provider PROVIDER_NAME
  ```

- Associates two web servers with pool.

  ```
  $ neutron lb-member-create --address  WEBSERVER1_IP --protocol-port 80
  mypool
  $ neutron lb-member-create --address  WEBSERVER2_IP --protocol-port 80
  mypool
  ```

- Creates a health monitor that checks to make sure our instances are still running on the specified protocol-port.

  ```
  $ neutron lb-healthmonitor-create --delay 3 --type HTTP --max-retries 3 \
    --timeout 3
  ```

- Associates a health monitor with pool.

  ```
  $ neutron lb-healthmonitor-associate  HEALTHMONITOR_UUID mypool
  ```

- Creates a virtual IP (VIP) address that, when accessed through the load balancer, directs the requests to one of the pool members.

  ```
  $ neutron lb-vip-create --name myvip --protocol-port 80 --protocol \
    HTTP --subnet-id SUBNET_UUID mypool
  ```

Plug-in specific extensions

Each vendor can choose to implement additional API extensions to the core API. This section describes the extensions for each plug-in.

VMware NSX extensions

These sections explain NSX plug-in extensions.

VMware NSX QoS extension

The VMware NSX QoS extension rate-limits network ports to guarantee a specific amount of bandwidth for each port. This extension, by default, is only accessible by a tenant with an admin role but is configurable through the `policy.json` file. To use this extension,

create a queue and specify the min/max bandwidth rates (kbps) and optionally set the QoS Marking and DSCP value (if your network fabric uses these values to make forwarding decisions). Once created, you can associate a queue with a network. Then, when ports are created on that network they are automatically created and associated with the specific queue size that was associated with the network. Because one size queue for a every port on a network might not be optimal, a scaling factor from the nova flavor `rxtx_factor` is passed in from Compute when creating the port to scale the queue.

Lastly, if you want to set a specific baseline QoS policy for the amount of bandwidth a single port can use (unless a network queue is specified with the network a port is created on) a default queue can be created in Networking which then causes ports created to be associated with a queue of that size times the rxtx scaling factor. Note that after a network or default queue is specified, queues are added to ports that are subsequently created but are not added to existing ports.

Basic VMware NSX QoS operations

This table shows example neutron commands that enable you to complete basic queue operations:

Basic VMware NSX QoS operations

Operation	Command
Creates QoS queue (admin-only).	`$ neutron queue-create --min 10 --max 1000 myqueue`
Associates a queue with a network.	`$ neutron net-create network --queue_id QUEUE_ID`
Creates a default system queue.	`$ neutron queue-create --default True --min 10 --max 2000 default`
Lists QoS queues.	`$ neutron queue-list`
Deletes a QoS queue.	`$ neutron queue-delete QUEUE_ID_OR_NAME`

VMware NSX provider networks extension

Provider networks can be implemented in different ways by the underlying NSX platform.

The *FLAT* and *VLAN* network types use bridged transport connectors. These network types enable the attachment of large number of ports. To handle the increased scale, the NSX plug-in can back a single OpenStack Network with a chain of NSX logical switches. You can

specify the maximum number of ports on each logical switch in this chain on the `max_lp_per_bridged_ls` parameter, which has a default value of 5,000.

The recommended value for this parameter varies with the NSX version running in the back-end, as shown in the following table.

Recommended values for max_lp_per_bridged_ls

NSX version	Recommended Value
2.x	64
3.0.x	5,000
3.1.x	5,000
3.2.x	10,000

In addition to these network types, the NSX plug-in also supports a special *l3_ext* network type, which maps external networks to specific NSX gateway services as discussed in the next section.

VMware NSX L3 extension

NSX exposes its L3 capabilities through gateway services which are usually configured out of band from OpenStack. To use NSX with L3 capabilities, first create an L3 gateway service in the NSX Manager. Next, in `/etc/neutron/plugins/vmware/nsx.ini` set `default_l3_gw_service_uuid` to this value. By default, routers are mapped to this gateway service.

VMware NSX L3 extension operations

Create external network and map it to a specific NSX gateway service:

```
$ neutron net-create public --router:external True --provider:network_type
l3_ext \
--provider:physical_network L3_GATEWAY_SERVICE_UUID
```

Terminate traffic on a specific VLAN from a NSX gateway service:

```
$ neutron net-create public --router:external True --provider:network_type
l3_ext \
--provider:physical_network L3_GATEWAY_SERVICE_UUID --provider:segmentation_id
VLAN_ID
```

Operational status synchronization in the VMware NSX plug-in

Starting with the Havana release, the VMware NSX plug-in provides an asynchronous mechanism for retrieving the operational status for neutron resources from the NSX back-end; this applies to *network*, *port*, and *router* resources.

The back-end is polled periodically and the status for every resource is retrieved; then the

status in the Networking database is updated only for the resources for which a status change occurred. As operational status is now retrieved asynchronously, performance for `GET` operations is consistently improved.

Data to retrieve from the back-end are divided in chunks in order to avoid expensive API requests; this is achieved leveraging NSX APIs response paging capabilities. The minimum chunk size can be specified using a configuration option; the actual chunk size is then determined dynamically according to: total number of resources to retrieve, interval between two synchronization task runs, minimum delay between two subsequent requests to the NSX back-end.

The operational status synchronization can be tuned or disabled using the configuration options reported in this table; it is however worth noting that the default values work fine in most cases.

Configuration options for tuning operational status synchronization in the NSX plug-in

Option name	Group	Default value	Type and constraints	Notes
`state_sync_interval`	`nsx_sync`	10 seconds	Integer; no constraint.	Interval in seconds between two run of the synchronization task. If the synchronization task takes more than `state_sync_interval` seconds to execute, a new instance of the task is started as soon as the other is completed. Setting the value for this option to 0 will disable the synchronization task.
`max_random_sync_delay`	`nsx_sync`	0 seconds	Integer. Must not exceed `min_sync_req_delay`	When different from zero, a random delay between 0 and `max_random_sync_delay` will be added before processing the next chunk.

Option name	Group	Default value	Type and constraints	Notes
min_sync_req_delay	nsx_sync	1 second	Integer. Must not exceed state_sync_interval.	The value of this option can be tuned according to the observed load on the NSX controllers. Lower values will result in faster synchronization, but might increase the load on the controller cluster.
min_chunk_size	nsx_sync	500 resources	Integer; no constraint.	Minimum number of resources to retrieve from the back-end for each synchronization chunk. The expected number of synchronization chunks is given by the ratio between state_sync_interval and min_sync_req_delay. This size of a chunk might increase if the total number of resources is such that more than min_chunk_size resources must be fetched in one chunk with the current number of chunks.
always_read_status	nsx_sync	False	Boolean; no constraint.	When this option is enabled, the operational status will always be retrieved from the NSX back-end ad every GET request.

Option name	Group	Default value	Type and constraints	Notes
				In this case it is advisable to disable the synchronization task.

When running multiple OpenStack Networking server instances, the status synchronization task should not run on every node; doing so sends unnecessary traffic to the NSX back-end and performs unnecessary DB operations. Set the `state_sync_interval` configuration option to a non-zero value exclusively on a node designated for back-end status synchronization.

The `fields=status` parameter in Networking API requests always triggers an explicit query to the NSX back end, even when you enable asynchronous state synchronization. For example, `GET /v2.0/networks/NET_ID?fields=status&fields=name`.

Big Switch plug-in extensions

This section explains the Big Switch neutron plug-in-specific extension.

Big Switch router rules

Big Switch allows router rules to be added to each tenant router. These rules can be used to enforce routing policies such as denying traffic between subnets or traffic to external networks. By enforcing these at the router level, network segmentation policies can be enforced across many VMs that have differing security groups.

Router rule attributes

Each tenant router has a set of router rules associated with it. Each router rule has the attributes in this table. Router rules and their attributes can be set using the **neutron router-update** command, through the horizon interface or the Networking API.

Big Switch Router rule attributes

Attribute name	Required	Input type	Description
source	Yes	A valid CIDR or one of the keywords ‘any’ or ‘external’	The network that a packet’s source IP must match for the rule to be applied.
destination	Yes	A valid CIDR or one of the keywords ‘any’ or ‘external’	The network that a packet’s destination IP must match for the rule to be applied.

Attribute name	Required	Input type	Description
action	Yes	‘permit’ or ‘deny’	Determines whether or not the matched packets will allowed to cross the router.
nexthop	No	A plus-separated (+) list of next-hop IP addresses. For example, `1.1.1.1+1.1.1.2`.	Overrides the default virtual router used to handle traffic for packets that match the rule.

Order of rule processing

The order of router rules has no effect. Overlapping rules are evaluated using longest prefix matching on the source and destination fields. The source field is matched first so it always takes higher precedence over the destination field. In other words, longest prefix matching is used on the destination field only if there are multiple matching rules with the same source.

Big Switch router rules operations

Router rules are configured with a router update operation in OpenStack Networking. The update overrides any previous rules so all rules must be provided at the same time.

Update a router with rules to permit traffic by default but block traffic from external networks to the 10.10.10.0/24 subnet:

```
$ neutron router-update ROUTER_UUID --router_rules type=dict list=true \
  source=any,destination=any,action=permit \
  source=external,destination=10.10.10.0/24,action=deny
```

Specify alternate next-hop addresses for a specific subnet:

```
$ neutron router-update ROUTER_UUID --router_rules type=dict list=true  \
  source=any,destination=any,action=permit \

source=10.10.10.0/24,destination=any,action=permit,nexthops=10.10.10.254+10.10.1
0.253
```

Block traffic between two subnets while allowing everything else:

```
$ neutron router-update ROUTER_UUID --router_rules type=dict list=true \
  source=any,destination=any,action=permit \
  source=10.10.10.0/24,destination=10.20.20.20/24,action=deny
```

L3 metering

The L3 metering API extension enables administrators to configure IP ranges and assign a specified label to them to be able to measure traffic that goes through a virtual router.

The L3 metering extension is decoupled from the technology that implements the measurement. Two abstractions have been added: One is the metering label that can contain metering rules. Because a metering label is associated with a tenant, all virtual routers in this tenant are associated with this label.

Basic L3 metering operations

Only administrators can manage the L3 metering labels and rules.

This table shows example **neutron** commands that enable you to complete basic L3 metering operations:

Basic L3 operations

Operation	Command
Creates a metering label.	`$ neutron meter-label-create LABEL1 --description "DESCRIPTION_LABEL1"`
Lists metering labels.	`$ neutron meter-label-list`
Shows information for a specified label.	`$ neutron meter-label-show LABEL_UUID` `$ neutron meter-label-show LABEL1`
Deletes a metering label.	`$ neutron meter-label-delete LABEL_UUID` `$ neutron meter-label-delete LABEL1`

Creates a metering rule.	`$ neutron meter-label-rule-create LABEL_UUID CIDR --direction DIRECTION \`
	`  --excluded`
	
	For example:
	
	`$ neutron meter-label-rule-create label1 10.0.0.0/24 --direction ingress`

Operation	Command
	`$ neutron meter-label-rule-create label1 20.0.0.0/24 --excluded`
Lists metering all label rules.	`$ neutron meter-label-rule-list`
Shows information for a specified label rule.	`$ neutron meter-label-rule-show RULE_UUID`
Deletes a metering label rule.	`$ neutron meter-label-rule-delete RULE_UUID`

Lists the value of created metering label rules.	`$ ceilometer sample-list -m SNMP_MEASUREMENT`
	For example:
	`$ ceilometer sample-list -m hardware.network.bandwidth.bytes`
	`$ ceilometer sample-list -m hardware.network.incoming.bytes`
	`$ ceilometer sample-list -m hardware.network.outgoing.bytes`
	`$ ceilometer sample-list -m hardware.network.outgoing.errors`

Advanced operational features

Logging settings

Networking components use Python logging module to do logging. Logging configuration can be provided in `neutron.conf` or as command-line options. Command options override ones in `neutron.conf`.

To configure logging for Networking components, use one of these methods:

- Provide logging settings in a logging configuration file.

  See Python logging how-to to learn more about logging.

- Provide logging setting in `neutron.conf`.

  ```
  [DEFAULT]
  ```

```
# Default log level is WARNING
# Show debugging output in logs (sets DEBUG log level output)
# debug = False

# log_date_format = %Y-%m-%d %H:%M:%S

# use_syslog = False
# syslog_log_facility = LOG_USER

# if use_syslog is False, we can set log_file and log_dir.
# if use_syslog is False and we do not set log_file,
# the log will be printed to stdout.
# log_file =
# log_dir =
```

Notifications

Notifications can be sent when Networking resources such as network, subnet and port are created, updated or deleted.

Notification options

To support DHCP agent, `rpc_notifier` driver must be set. To set up the notification, edit notification options in `neutron.conf`:

```
# Driver or drivers to handle sending notifications. (multi
# valued)
#notification_driver=messagingv2

# AMQP topic used for OpenStack notifications. (list value)
# Deprecated group/name - [rpc_notifier2]/topics
  notification_topics = notifications
```

Setting cases

Logging and RPC

These options configure the Networking server to send notifications through logging and RPC. The logging options are described in OpenStack Configuration Reference . RPC notifications go to `notifications.info` queue bound to a topic exchange defined by `control_exchange` in `neutron.conf`.

Notification System Options

A notification can be sent when a network, subnet, or port is created, updated or deleted.

The notification system options are:

- **`notification_driver`**

  Defines the driver or drivers to handle the sending of a notification. The six available options are:

  - **`messaging`**

    Send notifications using the 1.0 message format.

  - **`messagingv2`**

    Send notifications using the 2.0 message format (with a message envelope).

  - **`routing`**

    Configurable routing notifier (by priority or event_type).

  - **`log`**

    Publish notifications using Python logging infrastructure.

  - **`test`**

    Store notifications in memory for test verification.

  - **`noop`**

    Disable sending notifications entirely.

- **`default_notification_level`**

  Is used to form topic names or to set a logging level.

- **`default_publisher_id`**

  Is a part of the notification payload.

- **`notification_topics`**

  AMQP topic used for OpenStack notifications. They can be comma-separated values. The actual topic names will be the values of `default_notification_level`.

- **`control_exchange`**

  This is an option defined in oslo.messaging. It is the default exchange under

which topics are scoped. May be overridden by an exchange name specified in the `transport_url` option. It is a string value.

Below is a sample `neutron.conf` configuration file:

```
notification_driver = messagingv2

default_notification_level = INFO

host = myhost.com
default_publisher_id = $host

notification_topics = notifications

control_exchange=openstack
```

Authentication and authorization

Networking uses the Identity service as the default authentication service. When the Identity service is enabled, users who submit requests to the Networking service must provide an authentication token in `X-Auth-Token` request header. Users obtain this token by authenticating with the Identity service endpoint. For more information about authentication with the Identity service, see OpenStack Identity service API v2.0 Reference. When the Identity service is enabled, it is not mandatory to specify the tenant ID for resources in create requests because the tenant ID is derived from the authentication token.

The default authorization settings only allow administrative users to create resources on behalf of a different tenant. Networking uses information received from Identity to authorize user requests. Networking handles two kind of authorization policies:

- **Operation-based** policies specify access criteria for specific operations, possibly with fine-grained control over specific attributes.
- **Resource-based** policies specify whether access to specific resource is granted or not according to the permissions configured for the resource (currently available only for the network resource). The actual authorization policies enforced in Networking might vary from deployment to deployment.

The policy engine reads entries from the `policy.json` file. The actual location of this file might vary from distribution to distribution. Entries can be updated while the system is running, and no service restart is required. Every time the policy file is updated, the policies are automatically reloaded. Currently the only way of updating such policies is to edit the policy file. In this section, the terms *policy* and *rule* refer to objects that are

specified in the same way in the policy file. There are no syntax differences between a rule and a policy. A policy is something that is matched directly from the Networking policy engine. A rule is an element in a policy, which is evaluated. For instance in `"create_subnet": "rule:admin_or_network_owner"`, *create_subnet* is a policy, and *admin_or_network_owner* is a rule.

Policies are triggered by the Networking policy engine whenever one of them matches a Networking API operation or a specific attribute being used in a given operation. For instance the `create_subnet` policy is triggered every time a `POST /v2.0/subnets` request is sent to the Networking server; on the other hand `create_network:shared` is triggered every time the *shared* attribute is explicitly specified (and set to a value different from its default) in a `POST /v2.0/networks` request. It is also worth mentioning that policies can also be related to specific API extensions; for instance `extension:provider_network:set` is triggered if the attributes defined by the Provider Network extensions are specified in an API request.

An authorization policy can be composed by one or more rules. If more rules are specified then the evaluation policy succeeds if any of the rules evaluates successfully; if an API operation matches multiple policies, then all the policies must evaluate successfully. Also, authorization rules are recursive. Once a rule is matched, the rule(s) can be resolved to another rule, until a terminal rule is reached.

The Networking policy engine currently defines the following kinds of terminal rules:

- **Role-based rules** evaluate successfully if the user who submits the request has the specified role. For instance `"role:admin"` is successful if the user who submits the request is an administrator.
- **Field-based rules** evaluate successfully if a field of the resource specified in the current request matches a specific value. For instance `"field:networks:shared=True"` is successful if the `shared` attribute of the `network` resource is set to true.
- **Generic rules** compare an attribute in the resource with an attribute extracted from the user's security credentials and evaluates successfully if the comparison is successful. For instance `"tenant_id:%(tenant_id)s"` is successful if the tenant identifier in the resource is equal to the tenant identifier of the user submitting the request.

This extract is from the default `policy.json` file:

- A rule that evaluates successfully if the current user is an administrator or the owner of the resource specified in the request (tenant identifier is equal).

```
{
 "admin_or_owner": [
     [
```

```
            "role:admin"
        ],
        [
            "tenant_id:%(tenant_id)s"
        ]
    ],
    "admin_or_network_owner": [
        [
            "role:admin"
        ],
        [
            "tenant_id:%(network_tenant_id)s"
        ]
    ],
    "admin_only": [
        [
            "role:admin"
    ]
    ],
    "regular_user": [],
    "shared": [
        [
            "field:networks:shared=True"
        ]
    ],
    "default": [
        [
```

- The default policy that is always evaluated if an API operation does not match any of the policies in `policy.json`.

```
            "rule:admin_or_owner"
        ]
    ],
    "create_subnet": [
        [
            "rule:admin_or_network_owner"
        ]
    ],
    "get_subnet": [
        [
            "rule:admin_or_owner"
        ],
        [
```

```
        "rule:shared"
    ]
],
"update_subnet": [
    [
        "rule:admin_or_network_owner"
    ]
],
"delete_subnet": [
    [
        "rule:admin_or_network_owner"
    ]
],
"create_network": [],
"get_network": [
    [
        "rule:admin_or_owner"
    ],
```

- This policy evaluates successfully if either *admin_or_owner*, or *shared* evaluates successfully.

```
    [
        "rule:shared"
    ]
],
"create_network:shared": [
    [
        "rule:admin_only"
    ]
```

- This policy restricts the ability to manipulate the *shared* attribute for a network to administrators only.

```
],
"update_network": [
    [
        "rule:admin_or_owner"
    ]
],
"delete_network": [
    [
        "rule:admin_or_owner"
    ]
],
```

```
    "create_port": [],
    "create_port:mac_address": [
        [
            "rule:admin_or_network_owner"
        ]
    ],
    "create_port:fixed_ips": [
```

- This policy restricts the ability to manipulate the *mac_address* attribute for a port only to administrators and the owner of the network where the port is attached.

```
        [
            "rule:admin_or_network_owner"
        ]
    ],
    "get_port": [
        [
            "rule:admin_or_owner"
        ]
    ],
    "update_port": [
        [
            "rule:admin_or_owner"
        ]
    ],
     "delete_port": [
        [
            "rule:admin_or_owner"
        ]
    ]
}
```

In some cases, some operations are restricted to administrators only. This example shows you how to modify a policy file to permit tenants to define networks, see their resources, and permit administrative users to perform all other operations:

```
{
        "admin_or_owner": [["role:admin"], ["tenant_id:%(tenant_id)s"]],
        "admin_only": [["role:admin"]], "regular_user": [],
        "default": [["rule:admin_only"]],
        "create_subnet": [["rule:admin_only"]],
        "get_subnet": [["rule:admin_or_owner"]],
        "update_subnet": [["rule:admin_only"]],
        "delete_subnet": [["rule:admin_only"]],
        "create_network": [],
        "get_network": [["rule:admin_or_owner"]],
```

```
        "create_network:shared": [["rule:admin_only"]],
        "update_network": [["rule:admin_or_owner"]],
        "delete_network": [["rule:admin_or_owner"]],
        "create_port": [["rule:admin_only"]],
        "get_port": [["rule:admin_or_owner"]],
        "update_port": [["rule:admin_only"]],
        "delete_port": [["rule:admin_only"]]
}
```

Telemetry

Even in the cloud industry, providers must use a multi-step process for billing. The required steps to bill for usage in a cloud environment are metering, rating, and billing. Because the provider's requirements may be far too specific for a shared solution, rating and billing solutions cannot be designed in a common module that satisfies all. Providing users with measurements on cloud services is required to meet the `measured service` definition of cloud computing.

The Telemetry service was originally designed to support billing systems for OpenStack cloud resources. This project only covers the metering portion of the required processing for billing. This service collects information about the system and stores it in the form of samples in order to provide data about anything that can be billed.

In addition to system measurements, the Telemetry service also captures event notifications triggered when various actions are executed in the OpenStack system. This data is captured as Events and stored alongside metering data.

The list of meters is continuously growing, which makes it possible to use the data collected by Telemetry for different purposes, other than billing. For example, the autoscaling feature in the Orchestration service can be triggered by alarms this module sets and then gets notified within Telemetry.

The sections in this document contain information about the architecture and usage of Telemetry. The first section contains a brief summary about the system architecture used in a typical OpenStack deployment. The second section describes the data collection mechanisms. You can also read about alarming to understand how alarm definitions can be posted to Telemetry and what actions can happen if an alarm is raised. The last section contains a troubleshooting guide, which mentions error situations and possible solutions to the problems.

You can retrieve the collected samples in three different ways: with the REST API, with the command-line interface, or with the Metering tab on an OpenStack dashboard.

- System architecture

System architecture

The Telemetry service uses an agent-based architecture. Several modules combine their responsibilities to collect data, store samples in a database, or provide an API service for handling incoming requests.

The Telemetry service is built from the following agents and services:

ceilometer-api
: Presents aggregated metering data to consumers (such as billing engines and analytics tools).

ceilometer-polling
: Polls for different kinds of meter data by using the polling plug-ins (pollsters) registered in different namespaces. It provides a single polling interface across different namespaces.

ceilometer-agent-central
: Polls the public RESTful APIs of other OpenStack services such as Compute service and Image service, in order to keep tabs on resource existence, by using the polling plug-ins (pollsters) registered in the central polling namespace.

ceilometer-agent-compute
: Polls the local hypervisor or libvirt daemon to acquire performance data for the local instances, messages and emits the data as AMQP messages, by using the polling plug-ins (pollsters) registered in the compute polling namespace.

ceilometer-agent-ipmi
: Polls the local node with IPMI support, in order to acquire IPMI sensor data and Intel Node Manager data, by using the polling plug-ins (pollsters) registered in the IPMI

polling namespace.

ceilometer-agent-notification

Consumes AMQP messages from other OpenStack services.

ceilometer-collector

Consumes AMQP notifications from the agents, then dispatches these data to the appropriate data store.

ceilometer-alarm-evaluator

Determines when alarms fire due to the associated statistic trend crossing a threshold over a sliding time window.

ceilometer-alarm-notifier

Initiates alarm actions, for example calling out to a webhook with a description of the alarm state transition.

Note

The `ceilometer-polling` service is available since the Kilo release. It is intended to replace `ceilometer-agent-central`, `ceilometer-agent-compute`, and `ceilometer-agent-ipmi`.

Besides the `ceilometer-agent-compute` and the `ceilometer-agent-ipmi` services, all the other services are placed on one or more controller nodes.

The Telemetry architecture highly depends on the AMQP service both for consuming notifications coming from OpenStack services and internal communication.

Supported databases

The other key external component of Telemetry is the database, where events, samples, alarm definitions, and alarms are stored.

> **Note**
>
> Multiple database back ends can be configured in order to store events, samples, and alarms separately.

The list of supported database back ends:

- ElasticSearch (events only)
- MongoDB
- MySQL
- PostgreSQL
- HBase

Supported hypervisors

The Telemetry service collects information about the virtual machines, which requires close connection to the hypervisor that runs on the compute hosts.

The following is a list of supported hypervisors.

- The following hypervisors are supported via libvirt
  - Kernel-based Virtual Machine (KVM)
  - Quick Emulator (QEMU)
  - Linux Containers (LXC)
  - User-mode Linux (UML)

> **Note**
>
> For details about hypervisor support in libvirt please check the Libvirt API support matrix.

- Hyper-V
- XEN
- VMware vSphere

Supported networking services

Telemetry is able to retrieve information from OpenStack Networking and external networking services:

- OpenStack Networking:
  - Basic network meters
  - Firewall-as-a-Service (FWaaS) meters
  - Load-Balancer-as-a-Service (LBaaS) meters
  - VPN-as-a-Service (VPNaaS) meters
- SDN controller meters:
  - OpenDaylight
  - OpenContrail

Users, roles, and tenants

This service of OpenStack uses OpenStack Identity for authenticating and authorizing users. The required configuration options are listed in the Telemetry section in the OpenStack Configuration Reference.

The system uses two roles:`admin` and `non-admin`. The authorization happens before processing each API request. The amount of returned data depends on the role the requestor owns.

The creation of alarm definitions also highly depends on the role of the user, who initiated the action. Further details about *Alarms* handling can be found in this guide.

Data collection

The main responsibility of Telemetry in OpenStack is to collect information about the system that can be used by billing systems or interpreted by analytic tooling. The original focus, regarding to the collected data, was on the counters that can be used for billing, but the range is getting wider continuously.

Collected data can be stored in the form of samples or events in the supported databases, listed in *Supported databases*.

Samples can have various sources regarding to the needs and configuration of Telemetry, which requires multiple methods to collect data.

The available data collection mechanisms are:

Notifications
: Processing notifications from other OpenStack services, by consuming messages from the configured message queue system.

Polling
: Retrieve information directly from the hypervisor or from the host machine using SNMP, or by using the APIs of other OpenStack services.

RESTful API
: Pushing samples via the RESTful API of Telemetry.

Notifications

All the services send notifications about the executed operations or system state in OpenStack. Several notifications carry information that can be metered, like the CPU time of a VM instance created by OpenStack Compute service.

The Telemetry service has a separate agent that is responsible for consuming notifications, namely the notification agent. This component is responsible for consuming from the message bus and transforming notifications into events and measurement samples. Beginning in the Liberty release, the notification agent is responsible for all data processing such as transformations and publishing. After processing, the data is sent via AMQP to the collector service or any external service, which is responsible for persisting the data into the configured database back end.

The different OpenStack services emit several notifications about the various types of events that happen in the system during normal operation. Not all these notifications are consumed by the Telemetry service, as the intention is only to capture the billable events and notifications that can be used for monitoring or profiling purposes. The notification agent filters by the event type, that is contained by each notification message. The following table contains the event types by each OpenStack service that are transformed to samples by Telemetry.

OpenStack service	Event types	Note
OpenStack Compute	scheduler.run_instance.scheduled scheduler.select_destinations compute.instance.*	For a more detailed list of Compute notifications please check the System Usage Data Data wiki page.
Bare metal service	hardware.ipmi.*	
OpenStack Image service	image.update image.upload image.delete image.send	The required configuration for Image service can be found in Configure the Image service for Telemetry section section in the OpenStack Installation Guide
OpenStack Networking	floatingip.create.end floatingip.update.* floatingip.exists network.create.end network.update.* network.exists port.create.end port.update.* port.exists router.create.end router.update.* router.exists subnet.create.end subnet.update.*	

OpenStack service	Event types	Note
	subnet.exists l3.meter	
Orchestration service	orchestration.stack.create.end orchestration.stack.update.end orchestration.stack.delete.end orchestration.stack.resume.end orchestration.stack.suspend.end	
OpenStack Block Storage	volume.exists volume.create.* volume.delete.* volume.update.* volume.resize.* volume.attach.* volume.detach.* snapshot.exists snapshot.create.* snapshot.delete.* snapshot.update.* volume.backup.create.* volume.backup.delete.* volume.backup.restore.*	The required configuration for Block Storage service can be found in the Add the Block Storage service agent for Telemetry section section in the OpenStack Installation Guide.

> **Note**
>
> Some services require additional configuration to emit the notifications using the correct control exchange on the message queue and so forth. These configuration needs are referred in the above table for each OpenStack service that needs it.

Specific notifications from the Compute service are important for administrators and users. Configuring `nova_notifications` in the `nova.conf` file allows administrators to respond to events rapidly. For more information on configuring notifications for the compute service, see Telemetry services in the OpenStack Installation Guide.

> **Note**

> When the `store_events` option is set to `True` in `ceilometer.conf`, Prior to the Kilo release, the notification agent needed database access in order to work properly.

Middleware for the OpenStack Object Storage service

A subset of Object Store statistics requires additional middleware to be installed behind the proxy of Object Store. This additional component emits notifications containing data-flow-oriented meters, namely the `storage.objects.(incoming|outgoing).bytes` values. The list of these meters are listed in *OpenStack Object Storage*, marked with `notification` as origin.

The instructions on how to install this middleware can be found in Configure the Object Storage service for Telemetry section in the OpenStack Installation Guide.

Telemetry middleware

Telemetry provides the capability of counting the HTTP requests and responses for each API endpoint in OpenStack. This is achieved by storing a sample for each event marked as `audit.http.request`, `audit.http.response`, `http.request` or `http.response`.

It is recommended that these notifications be consumed as events rather than samples to better index the appropriate values and avoid massive load on the Metering database. If preferred, Telemetry can consume these events as samples if the services are configured to emit `http.*` notifications.

Polling

The Telemetry service is intended to store a complex picture of the infrastructure. This goal requires additional information than what is provided by the events and notifications published by each service. Some information is not emitted directly, like resource usage of the VM instances.

Therefore Telemetry uses another method to gather this data by polling the infrastructure including the APIs of the different OpenStack services and other assets, like hypervisors. The latter case requires closer interaction with the compute hosts. To solve this issue, Telemetry uses an agent based architecture to fulfill the requirements against the data collection.

There are three types of agents supporting the polling mechanism, the `compute agent`, the `central agent`, and the `IPMI agent`. Under the hood, all the types of polling agents are the same `ceilometer-polling` agent, except that they load different polling plug-ins (pollsters) from different namespaces to gather data. The following subsections give further information regarding the architectural and configuration details of these components.

Running **ceilometer-agent-compute** is exactly the same as:

```
$ ceilometer-polling --polling-namespaces compute
```

Running **ceilometer-agent-central** is exactly the same as:

```
$ ceilometer-polling --polling-namespaces central
```

Running **ceilometer-agent-ipmi** is exactly the same as:

```
$ ceilometer-polling --polling-namespaces ipmi
```

In addition to loading all the polling plug-ins registered in the specified namespaces, the `ceilometer-polling` agent can also specify the polling plug-ins to be loaded by using the `pollster-list` option:

```
$ ceilometer-polling --polling-namespaces central \
        --pollster-list image image.size storage.*
```

> **Note**
>
> HA deployment is NOT supported if the `pollster-list` option is used.

> **Note**
>
> The `ceilometer-polling` service is available since Kilo release.

Central agent

This agent is responsible for polling public REST APIs to retrieve additional information on OpenStack resources not already surfaced via notifications, and also for polling hardware resources over SNMP.

The following services can be polled with this agent:

- OpenStack Networking
- OpenStack Object Storage
- OpenStack Block Storage
- Hardware resources via SNMP
- Energy consumption meters via Kwapi framework

To install and configure this service use the Add the Telemetry service section in the OpenStack Installation Guide.

The central agent does not need direct database connection. The samples collected by this agent are sent via AMQP to the notification agent to be processed.

> **Note**
>
> Prior to the Liberty release, data from the polling agents was processed locally and published accordingly rather than by the notification agent.

Compute agent

This agent is responsible for collecting resource usage data of VM instances on individual compute nodes within an OpenStack deployment. This mechanism requires a closer interaction with the hypervisor, therefore a separate agent type fulfills the collection of the related meters, which is placed on the host machines to locally retrieve this information.

A compute agent instance has to be installed on each and every compute node, installation instructions can be found in the Install the Compute agent for Telemetry section in the OpenStack Installation Guide.

Just like the central agent, this component also does not need a direct database connection. The samples are sent via AMQP to the notification agent.

The list of supported hypervisors can be found in *Supported hypervisors*. The compute agent uses the API of the hypervisor installed on the compute hosts. Therefore the supported meters may be different in case of each virtualization back end, as each inspection tool provides a different set of meters.

The list of collected meters can be found in *OpenStack Compute*. The support column provides the information that which meter is available for each hypervisor supported by the Telemetry service.

> **Note**
>
> Telemetry supports Libvirt, which hides the hypervisor under it.

IPMI agent

This agent is responsible for collecting IPMI sensor data and Intel Node Manager data on individual compute nodes within an OpenStack deployment. This agent requires an IPMI capable node with the ipmitool utility installed, which is commonly used for IPMI control on various Linux distributions.

An IPMI agent instance could be installed on each and every compute node with IPMI support, except when the node is managed by the Bare metal service and the `conductor.send_sensor_data` option is set to `true` in the Bare metal service. It is no harm to install this agent on a compute node without IPMI or Intel Node Manager support, as the agent checks for the hardware and if none is available, returns empty data. It is suggested that you install the IPMI agent only on an IPMI capable node for performance

reasons.

Just like the central agent, this component also does not need direct database access. The samples are sent via AMQP to the notification agent.

The list of collected meters can be found in *Bare metal service*.

> **Note**
>
> Do not deploy both the IPMI agent and the Bare metal service on one compute node. If `conductor.send_sensor_data` is set, this misconfiguration causes duplicated IPMI sensor samples.

Support for HA deployment

Both the polling agents and notification agents can run in an HA deployment, which means that multiple instances of these services can run in parallel with workload partitioning among these running instances.

The Tooz library provides the coordination within the groups of service instances. It provides an API above several back ends that can be used for building distributed applications.

Tooz supports various drivers including the following back end solutions:

- Zookeeper. Recommended solution by the Tooz project.
- Redis. Recommended solution by the Tooz project.
- Memcached. Recommended for testing.

You must configure a supported Tooz driver for the HA deployment of the Telemetry services.

For information about the required configuration options that have to be set in the `ceilometer.conf` configuration file for both the central and compute agents, see the Coordination section in the OpenStack Configuration Reference.

Notification agent HA deployment

In the Kilo release, workload partitioning support was added to the notification agent. This is particularly useful as the pipeline processing is handled exclusively by the notification agent now which may result in a larger amount of load.

To enable workload partitioning by notification agent, the `backend_url` option must be set in the `ceilometer.conf` configuration file. Additionally, `workload_partitioning` should be enabled in the Notification section in the OpenStack Configuration Reference.

> **Note**
>
> In Liberty, the notification agent creates multiple queues to divide the workload across all active agents. The number of queues can be controlled by the `pipeline_processing_queues` option in the `ceilometer.conf` configuration file. A larger value will result in better distribution of tasks but will also require more memory and longer startup time. It is recommended to have a value approximately three times the number of active notification agents. At a minimum, the value should be equal to the number of active agents.

Polling agent HA deployment

> **Note**
>
> Without the `backend_url` option being set only one instance of both the central and compute agent service is able to run and function correctly.

The availability check of the instances is provided by heartbeat messages. When the connection with an instance is lost, the workload will be reassigned within the remained instances in the next polling cycle.

> **Note**
>
> `Memcached` uses a `timeout` value, which should always be set to a value that is higher than the `heartbeat` value set for Telemetry.

For backward compatibility and supporting existing deployments, the central agent configuration also supports using different configuration files for groups of service instances of this type that are running in parallel. For enabling this configuration set a value for the `partitioning_group_prefix` option in the Central section in the OpenStack Configuration Reference.

> **Warning**
>
> For each sub-group of the central agent pool with the same `partitioning_group_prefix` a disjoint subset of meters must be polled, otherwise samples may be missing or duplicated. The list of meters to poll can be set in the `/etc/ceilometer/pipeline.yaml` configuration file. For more information about pipelines see *Data collection and processing*.

To enable the compute agent to run multiple instances simultaneously with workload partitioning, the `workload_partitioning` option has to be set to `True` under the Compute section in the `ceilometer.conf` configuration file.

Send samples to Telemetry

While most parts of the data collection in the Telemetry service are automated, Telemetry provides the possibility to submit samples via the REST API to allow users to send custom samples into this service.

This option makes it possible to send any kind of samples without the need of writing extra code lines or making configuration changes.

The samples that can be sent to Telemetry are not limited to the actual existing meters. There is a possibility to provide data for any new, customer defined counter by filling out all the required fields of the POST request.

If the sample corresponds to an existing meter, then the fields like `meter-type` and meter name should be matched accordingly.

The required fields for sending a sample using the command-line client are:

- ID of the corresponding resource. (*--resource-id*)
- Name of meter. (*--meter-name*)
- Type of meter. (*--meter-type*)

  Predefined meter types:

  - Gauge
  - Delta
  - Cumulative
- Unit of meter. (*--meter-unit*)
- Volume of sample. (*--sample-volume*)

To send samples to Telemetry using the command-line client, the following command should be invoked:

```
$ ceilometer sample-create -r 37128ad6-daaa-4d22-9509-b7e1c6b08697 \
  -m memory.usage --meter-type gauge --meter-unit MB --sample-volume 48
+-------------------+--------------------------------------------+
| Property          | Value                                      |
+-------------------+--------------------------------------------+
| message_id        | 6118820c-2137-11e4-a429-08002715c7fb       |
| name              | memory.usage                               |
| project_id        | e34eaa91d52a4402b4cb8bc9bbd308c1           |
| resource_id       | 37128ad6-daaa-4d22-9509-b7e1c6b08697       |
| resource_metadata | {}                                         |
```

```
| source            | e34eaa91d52a4402b4cb8bc9bbd308c1:openstack |
| timestamp         | 2014-08-11T09:10:46.358926                 |
| type              | gauge                                      |
| unit              | MB                                         |
| user_id           | 679b0499e7a34ccb9d90b64208401f8e           |
| volume            | 48.0                                       |
+-------------------+--------------------------------------------+
```

Data collection and processing

The mechanism by which data is collected and processed is called a pipeline. Pipelines, at the configuration level, describe a coupling between sources of data and the corresponding sinks for transformation and publication of data.

A source is a producer of data: `samples` or `events`. In effect, it is a set of pollsters or notification handlers emitting datapoints for a set of matching meters and event types.

Each source configuration encapsulates name matching, polling interval determination, optional resource enumeration or discovery, and mapping to one or more sinks for publication.

Data gathered can be used for different purposes, which can impact how frequently it needs to be published. Typically, a meter published for billing purposes needs to be updated every 30 minutes while the same meter may be needed for performance tuning every minute.

> **Warning**
>
> Rapid polling cadences should be avoided, as it results in a huge amount of data in a short time frame, which may negatively affect the performance of both Telemetry and the underlying database back end. We therefore strongly recommend you do not use small granularity values like 10 seconds.

A sink, on the other hand, is a consumer of data, providing logic for the transformation and publication of data emitted from related sources.

In effect, a sink describes a chain of handlers. The chain starts with zero or more transformers and ends with one or more publishers. The first transformer in the chain is passed data from the corresponding source, takes some action such as deriving rate of change, performing unit conversion, or aggregating, before passing the modified data to the next step that is described in *Publishers*.

Pipeline configuration

Pipeline configuration by default, is stored in separate configuration files, called `pipeline.yaml` and `event_pipeline.yaml`, next to the `ceilometer.conf` file. The meter

pipeline and event pipeline configuration files can be set by the `pipeline_cfg_file` and `event_pipeline_cfg_file` options listed in the Description of configuration options for api table section in the OpenStack Configuration Reference respectively. Multiple pipelines can be defined in one pipeline configuration file.

The meter pipeline definition looks like:

```
---
sources:
  - name: 'source name'
    interval: 'how often should the samples be injected into the pipeline'
    meters:
      - 'meter filter'
    resources:
      - 'list of resource URLs'
    sinks
      - 'sink name'
sinks:
  - name: 'sink name'
    transformers: 'definition of transformers'
    publishers:
      - 'list of publishers'
```

The interval parameter in the sources section should be defined in seconds. It determines the polling cadence of sample injection into the pipeline, where samples are produced under the direct control of an agent.

There are several ways to define the list of meters for a pipeline source. The list of valid meters can be found in *Measurements*. There is a possibility to define all the meters, or just included or excluded meters, with which a source should operate:

- To include all meters, use the * wildcard symbol. It is highly advisable to select only the meters that you intend on using to avoid flooding the metering database with unused data.
- To define the list of meters, use either of the following:
  - To define the list of included meters, use the `meter_name` syntax.
  - To define the list of excluded meters, use the `!meter_name` syntax.
  - For meters, which have variants identified by a complex name field, use the wildcard symbol to select all, for example, for `instance:m1.tiny`, use `instance:\*`.

> **Note**
>
> Please be aware that we do not have any duplication check between pipelines and if

> you add a meter to multiple pipelines then it is assumed the duplication is intentional and may be stored multiple times according to the specified sinks.

The above definition methods can be used in the following combinations:

- Use only the wildcard symbol.
- Use the list of included meters.
- Use the list of excluded meters.
- Use wildcard symbol with the list of excluded meters.

> **Note**
>
> At least one of the above variations should be included in the meters section. Included and excluded meters cannot co-exist in the same pipeline. Wildcard and included meters cannot co-exist in the same pipeline definition section.

The optional resources section of a pipeline source allows a static list of resource URLs to be configured for polling.

The transformers section of a pipeline sink provides the possibility to add a list of transformer definitions. The available transformers are:

Name of transformer	Reference name for configuration
Accumulator	accumulator
Aggregator	aggregator
Arithmetic	arithmetic
Rate of change	rate_of_change
Unit conversion	unit_conversion
Delta	delta

The publishers section contains the list of publishers, where the samples data should be sent after the possible transformations.

Similarly, the event pipeline definition looks like:

```
---
sources:
  - name: 'source name'
    events:
      - 'event filter'
    sinks
      - 'sink name'
sinks:
  - name: 'sink name'
```

```
    publishers:
      - 'list of publishers'
```

The event filter uses the same filtering logic as the meter pipeline.

Transformers

The definition of transformers can contain the following fields:

name
Name of the transformer.

parameters
Parameters of the transformer.

The parameters section can contain transformer specific fields, like source and target fields with different subfields in case of the rate of change, which depends on the implementation of the transformer.

In the case of the transformer that creates the `cpu_util` meter, the definition looks like:

```
transformers:
    - name: "rate_of_change"
      parameters:
          target:
              name: "cpu_util"
              unit: "%"
              type: "gauge"
              scale: "100.0 / (10**9 * (resource_metadata.cpu_number or 1))"
```

The rate of change the transformer generates is the `cpu_util` meter from the sample values of the `cpu` counter, which represents cumulative CPU time in nanoseconds. The transformer definition above defines a scale factor (for nanoseconds and multiple CPUs), which is applied before the transformation derives a sequence of gauge samples with unit `%`, from sequential values of the `cpu` meter.

The definition for the disk I/O rate, which is also generated by the rate of change transformer:

```
transformers:
    - name: "rate_of_change"
      parameters:
          source:
              map_from:
                  name: "disk\\.(read|write)\\.(bytes|requests)"
                  unit: "(B|request)"
          target:
```

```
            map_to:
                name: "disk.\\1.\\2.rate"
                unit: "\\1/s"
            type: "gauge"
```

Unit conversion transformer

Transformer to apply a unit conversion. It takes the volume of the meter and multiplies it with the given `scale` expression. Also supports `map_from` and `map_to` like the rate of change transformer.

Sample configuration:

```
transformers:
    - name: "unit_conversion"
      parameters:
          target:
              name: "disk.kilobytes"
              unit: "KB"
              scale: "volume * 1.0 / 1024.0"
```

With `map_from` and `map_to`:

```
transformers:
    - name: "unit_conversion"
      parameters:
          source:
              map_from:
                  name: "disk\\.(read|write)\\.bytes"
          target:
              map_to:
                  name: "disk.\\1.kilobytes"
              scale: "volume * 1.0 / 1024.0"
              unit: "KB"
```

Aggregator transformer

A transformer that sums up the incoming samples until enough samples have come in or a timeout has been reached.

Timeout can be specified with the `retention_time` option. If we want to flush the aggregation after a set number of samples have been aggregated, we can specify the size parameter.

The volume of the created sample is the sum of the volumes of samples that came into the transformer. Samples can be aggregated by the attributes `project_id`, `user_id` and `resource_metadata`. To aggregate by the chosen attributes, specify them in the

configuration and set which value of the attribute to take for the new sample (first to take the first sample's attribute, last to take the last sample's attribute, and drop to discard the attribute).

To aggregate 60s worth of samples by `resource_metadata` and keep the `resource_metadata` of the latest received sample:

```
transformers:
    - name: "aggregator"
      parameters:
          retention_time: 60
          resource_metadata: last
```

To aggregate each 15 samples by `user_id` and `resource_metadata` and keep the `user_id` of the first received sample and drop the `resource_metadata`:

```
transformers:
    - name: "aggregator"
      parameters:
          size: 15
          user_id: first
          resource_metadata: drop
```

Accumulator transformer

This transformer simply caches the samples until enough samples have arrived and then flushes them all down the pipeline at once:

```
transformers:
    - name: "accumulator"
      parameters:
          size: 15
```

Multi meter arithmetic transformer

This transformer enables us to perform arithmetic calculations over one or more meters and/or their metadata, for example:

```
memory_util = 100 * memory.usage / memory
```

A new sample is created with the properties described in the `target` section of the transformer's configuration. The sample's volume is the result of the provided expression. The calculation is performed on samples from the same resource.

> **Note**
>
> The calculation is limited to meters with the same interval.

Example configuration:

```
transformers:
    - name: "arithmetic"
      parameters:
        target:
          name: "memory_util"
          unit: "%"
          type: "gauge"
          expr: "100 * $(memory.usage) / $(memory)"
```

To demonstrate the use of metadata, here is the implementation of a silly meter that shows average CPU time per core:

```
transformers:
    - name: "arithmetic"
      parameters:
        target:
          name: "avg_cpu_per_core"
          unit: "ns"
          type: "cumulative"
          expr: "$(cpu) / ($(cpu).resource_metadata.cpu_number or 1)"
```

> **Note**
>
> Expression evaluation gracefully handles NaNs and exceptions. In such a case it does not create a new sample but only logs a warning.

Delta transformer

This transformer calculates the change between two sample datapoints of a resource. It can be configured to capture only the positive growth deltas.

Example configuration:

```
transformers:
    - name: "delta"
      parameters:
        target:
            name: "cpu.delta"
        growth_only: True
```

Meter definitions

The Telemetry service collects a subset of the meters by filtering notifications emitted by other OpenStack services. Starting with the Liberty release, you can find the meter definitions in a separate configuration file, called `ceilometer/meter/data/meter.yaml`. This enables operators/administrators to add new meters to Telemetry project by updating the `meter.yaml` file without any need for additional code changes.

> **Note**
>
> The `meter.yaml` file should be modified with care. Unless intended do not remove any existing meter definitions from the file. Also, the collected meters can differ in some cases from what is referenced in the documentation.

A standard meter definition looks like:

```
---
metric:
  - name: 'meter name'
    event_type: 'event name'
    type: 'type of meter eg: gauge, cumulative or delta'
    unit: 'name of unit eg: MB'
    volume: 'path to a measurable value eg: $.payload.size'
    resource_id: 'path to resource id eg: $.payload.id'
    project_id: 'path to project id eg: $.payload.owner'
```

The definition above shows a simple meter definition with some fields, from which `name`, `event_type`, `type`, `unit`, and `volume` are required. If there is a match on the event type, samples are generated for the meter.

If you take a look at the `meter.yaml` file, it contains the sample definitions for all the meters that Telemetry is collecting from notifications. The value of each field is specified by using json path in order to find the right value from the notification message. In order to be able to specify the right field you need to be aware of the format of the consumed notification. The values that need to be searched in the notification message are set with a json path starting with `$.` For instance, if you need the `size` information from the payload you can define it like `$.payload.size`.

A notification message may contain multiple meters. You can use `*` in the meter definition to capture all the meters and generate samples respectively. You can use wild cards as shown in the following example:

```
---
metric:
  - name: $.payload.measurements.[*].metric.[*].name
```

```
    event_type: 'event_name.*'
    type: 'delta'
    unit: $.payload.measurements.[*].metric.[*].unit
    volume: payload.measurements.[*].result
    resource_id: $.payload.target
    user_id: $.payload.initiator.id
    project_id: $.payload.initiator.project_id
```

In the above example, the `name` field is a json path with matching a list of meter names defined in the notification message.

You can even use complex operations on json paths. In the following example, `volume` and `resource_id` fields perform an arithmetic and string concatenation:

```
---
metric:
- name: 'compute.node.cpu.idle.percent'
  event_type: 'compute.metrics.update'
  type: 'gauge'
  unit: 'percent'
  volume: payload.metrics[?(@.name='cpu.idle.percent')].value * 100
  resource_id: $.payload.host + "_" + $.payload.nodename
```

You can use the `timedelta` plug-in to evaluate the difference in seconds between two `datetime` fields from one notification.

```
---
metric:
- name: 'compute.instance.booting.time'
  event_type: 'compute.instance.create.end'
 type: 'gauge'
 unit: 'sec'
 volume:
   fields: [$.payload.created_at, $.payload.launched_at]
   plugin: 'timedelta'
 project_id: $.payload.tenant_id
 resource_id: $.payload.instance_id
```

You will find some existence meters in the `meter.yaml`. These meters have a `volume` as `1` and are at the bottom of the yaml file with a note suggesting that these will be removed in Mitaka release.

For example, the meter definition for existence meters is as follows:

```
---
metric:
  - name: 'meter name'
    type: 'delta'
    unit: 'volume'
```

```
    volume: 1
    event_type:
        - 'event type'
    resource_id: $.payload.volume_id
    user_id: $.payload.user_id
    project_id: $.payload.tenant_id
```

These meters are not loaded by default. To load these meters, flip the disable_non_metric_meters option in the `ceilometer.conf` file.

Block Storage audit script setup to get notifications

If you want to collect OpenStack Block Storage notification on demand, you can use **cinder-volume-usage-audit** from OpenStack Block Storage. This script becomes available when you install OpenStack Block Storage, so you can use it without any specific settings and you don't need to authenticate to access the data. To use it, you must run this command in the following format:

```
$ cinder-volume-usage-audit \
  --start_time='YYYY-MM-DD HH:MM:SS' --end_time='YYYY-MM-DD HH:MM:SS'
--send_actions
```

This script outputs what volumes or snapshots were created, deleted, or exists in a given period of time and some information about these volumes or snapshots. Information about the existence and size of volumes and snapshots is store in the Telemetry service. This data is also stored as an event which is the recommended usage as it provides better indexing of data.

Using this script via cron you can get notifications periodically, for example, every 5 minutes:

```
*/5 * * * * /path/to/cinder-volume-usage-audit --send_actions
```

Storing samples

The Telemetry service has a separate service that is responsible for persisting the data that comes from the pollsters or is received as notifications. The data can be stored in a file or a database back end, for which the list of supported databases can be found in *Supported databases*. The data can also be sent to an external data store by using an HTTP dispatcher.

The `ceilometer-collector` service receives the data as messages from the message bus of the configured AMQP service. It sends these datapoints without any modification to the configured target. The service has to run on a host machine from which it has access to the configured dispatcher.

> **Note**
>
> Multiple dispatchers can be configured for Telemetry at one time.

Multiple `ceilometer-collector` processes can be run at a time. It is also supported to start multiple worker threads per collector process. The `collector_workers` configuration option has to be modified in the Collector section of the `ceilometer.conf` configuration file.

Database dispatcher

When the database dispatcher is configured as data store, you have the option to set a `time_to_live` option (ttl) for samples. By default the time to live value for samples is set to -1, which means that they are kept in the database forever.

The time to live value is specified in seconds. Each sample has a time stamp, and the `ttl` value indicates that a sample will be deleted from the database when the number of seconds has elapsed since that sample reading was stamped. For example, if the time to live is set to 600, all samples older than 600 seconds will be purged from the database.

Certain databases support native TTL expiration. In cases where this is not possible, a command-line script, which you can use for this purpose is `ceilometer-expirer`. You can run it in a cron job, which helps to keep your database in a consistent state.

The level of support differs in case of the configured back end:

Database	TTL value support	Note
MongoDB	Yes	MongoDB has native TTL support for deleting samples that are older than the configured ttl value.
SQL-based back ends	Yes	`ceilometer-expirer` has to be used for deleting samples and its related data from the database.
HBase	No	Telemetry's HBase support does not include native TTL nor `ceilometer-expirer` support.
DB2 NoSQL	No	DB2 NoSQL does not have native TTL nor `ceilometer-expirer` support.

HTTP dispatcher

The Telemetry service supports sending samples to an external HTTP target. The samples

are sent without any modification. To set this option as the collector's target, the `dispatcher` has to be changed to `http` in the `ceilometer.conf` configuration file. For the list of options that you need to set, see the see the dispatcher_http section in the OpenStack Configuration Reference.

File dispatcher

You can store samples in a file by setting the `dispatcher` option in the `ceilometer.conf` file. For the list of configuration options, see the dispatcher_file section in the OpenStack Configuration Reference.

Gnocchi dispatcher

The Telemetry service supports sending the metering data to Gnocchi back end through the gnocchi dispatcher. To set this option as the target, change the `dispatcher` to `gnocchi` in the `ceilometer.conf` configuration file.

For the list of options that you need to set, see the dispatcher_gnocchi section in the OpenStack Configuration Reference.

Data retrieval

The Telemetry service offers several mechanisms from which the persisted data can be accessed. As described in *System architecture* and in *Data collection*, the collected information can be stored in one or more database back ends, which are hidden by the Telemetry RESTful API.

> **Note**
>
> It is highly recommended not to access the database directly and read or modify any data in it. The API layer hides all the changes in the actual database schema and provides a standard interface to expose the samples, alarms and so forth.

Telemetry v2 API

The Telemetry service provides a RESTful API, from which the collected samples and all the related information can be retrieved, like the list of meters, alarm definitions and so forth.

The Telemetry API URL can be retrieved from the service catalog provided by OpenStack Identity, which is populated during the installation process. The API access needs a valid token and proper permission to retrieve data, as described in *Users, roles, and tenants*.

Further information about the available API endpoints can be found in the Telemetry API

Reference.

Query

The API provides some additional functionalities, like querying the collected data set. For the samples and alarms API endpoints, both simple and complex query styles are available, whereas for the other endpoints only simple queries are supported.

After validating the query parameters, the processing is done on the database side in the case of most database back ends in order to achieve better performance.

Simple query

Many of the API endpoints accept a query filter argument, which should be a list of data structures that consist of the following items:

- `field`
- `op`
- `value`
- `type`

Regardless of the endpoint on which the filter is applied on, it will always target the fields of the Sample type.

Several fields of the API endpoints accept shorter names than the ones defined in the reference. The API will do the transformation internally and return the output with the fields that are listed in the API reference. The fields are the following:

- `project_id`: project
- `resource_id`: resource
- `user_id`: user

When a filter argument contains multiple constraints of the above form, a logical `AND` relation between them is implied.

Complex query

The filter expressions of the complex query feature operate on the fields of `Sample`, `Alarm` and `AlarmChange` types. The following comparison operators are supported:

- `=`
- `!=`
- `<`
- `<=`
- `>`
- `>=`

The following logical operators can be used:

- and
- or
- not

> **Note**
>
> The not operator has different behavior in MongoDB and in the SQLAlchemy-based database engines. If the not operator is applied on a non existent metadata field then the result depends on the database engine. In case of MongoDB, it will return every sample as the not operator is evaluated true for every sample where the given field does not exist. On the other hand the SQL-based database engine will return an empty result because of the underlying join operation.

Complex query supports specifying a list of orderby expressions. This means that the result of the query can be ordered based on the field names provided in this list. When multiple keys are defined for the ordering, these will be applied sequentially in the order of the specification. The second expression will be applied on the groups for which the values of the first expression are the same. The ordering can be ascending or descending.

The number of returned items can be bounded using the limit option.

The filter, orderby and limit fields are optional.

> **Note**
>
> As opposed to the simple query, complex query is available via a separate API endpoint. For more information see the Telemetry v2 Web API Reference.

Statistics

The sample data can be used in various ways for several purposes, like billing or profiling. In external systems the data is often used in the form of aggregated statistics. The Telemetry API provides several built-in functions to make some basic calculations available without any additional coding.

Telemetry supports the following statistics and aggregation functions:

avg
: Average of the sample volumes over each period.

cardinality
: Count of distinct values in each period identified by a key specified as the parameter of this aggregate function. The supported parameter values are:

- project_id
- resource_id
- user_id

> **Note**
>
> The aggregate.param option is required.

count
Number of samples in each period.

max
Maximum of the sample volumes in each period.

min
Minimum of the sample volumes in each period.

stddev
Standard deviation of the sample volumes in each period.

sum
Sum of the sample volumes over each period.

The simple query and the statistics functionality can be used together in a single API request.

Telemetry command-line client and SDK

The Telemetry service provides a command-line client, with which the collected data is available just as the alarm definition and retrieval options. The client uses the Telemetry RESTful API in order to execute the requested operations.

To be able to use the **ceilometer** command, the python-ceilometerclient package needs to be installed and configured properly. For details about the installation process, see the Telemetry chapter in the OpenStack Installation Guide.

> **Note**
>
> The Telemetry service captures the user-visible resource usage data. Therefore the database will not contain any data without the existence of these resources, like VM images in the OpenStack Image service.

Similarly to other OpenStack command-line clients, the ceilometer client uses OpenStack Identity for authentication. The proper credentials and *--auth_url* parameter have to be defined via command line parameters or environment variables.

This section provides some examples without the aim of completeness. These commands can be used for instance for validating an installation of Telemetry.

To retrieve the list of collected meters, the following command should be used:

```
$ ceilometer meter-list
+------------------------+------------+------+--------------------------------------+----------------------------------+----------------------------------+
| Name                   | Type       | Unit | Resource ID                          | User ID                          | Project ID                       |
+------------------------+------------+------+--------------------------------------+----------------------------------+----------------------------------+
| cpu                    | cumulative | ns   | bb52e52b-1e42-4751-b3ac-45c52d83ba07     | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| cpu                    | cumulative | ns   | c8d2e153-a48f-4cec-9e93-86e7ac6d4b0b     | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| cpu_util               | gauge      | %    | bb52e52b-1e42-4751-b3ac-45c52d83ba07     | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| cpu_util               | gauge      | %    | c8d2e153-a48f-4cec-9e93-86e7ac6d4b0b     | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| disk.device.read.bytes | cumulative | B    | bb52e52b-1e42-4751-b3ac-45c52d83ba07-hdd | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| disk.device.read.bytes | cumulative | B    | bb52e52b-1e42-4751-b3ac-45c52d83ba07-vda | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| disk.device.read.bytes | cumulative | B    | c8d2e153-a48f-4cec-9e93-86e7ac6d4b0b-hdd | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| disk.device.read.bytes | cumulative | B    | c8d2e153-a48f-4cec-9e93-86e7ac6d4b0b-vda | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| ...                    |            |      |                                      |                                  |                                  |
+------------------------+------------+------+--------------------------------------+----------------------------------+----------------------------------+
```

The **ceilometer** command was run with `admin` rights, which means that all the data is accessible in the database. For more information about access right see *Users, roles, and tenants*. As it can be seen in the above example, there are two VM instances existing in the system, as there are VM instance related meters on the top of the result list. The existence of these meters does not indicate that these instances are running at the time of the request. The result contains the currently collected meters per resource, in an ascending order based on the name of the meter.

Samples are collected for each meter that is present in the list of meters, except in case of instances that are not running or deleted from the OpenStack Compute database. If an instance no longer exists and there is a `time_to_live` value set in the `ceilometer.conf` configuration file, then a group of samples are deleted in each expiration cycle. When the last sample is deleted for a meter, the database can be cleaned up by running ceilometer-expirer and the meter will not be present in the list above anymore. For more information about the expiration procedure see *Storing samples*.

The Telemetry API supports simple query on the meter endpoint. The query functionality has the following syntax:

```
--query <field1><operator1><value1>;...;<field_n><operator_n><value_n>
```

The following command needs to be invoked to request the meters of one VM instance:

```
$ ceilometer meter-list --query resource=bb52e52b-1e42-4751-b3ac-45c52d83ba07
+------------------------+------------+-----------+--------------------------------------+----------------------------------+----------------------------------+
| Name                   | Type       | Unit      | Resource ID                          | User ID                          | Project ID                       |
+------------------------+------------+-----------+--------------------------------------+----------------------------------+----------------------------------+
| cpu                    | cumulative | ns        | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| cpu_util               | gauge      | %         | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| cpu_l3_cache           | gauge      | B         | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| disk.ephemeral.size    | gauge      | GB        | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| disk.read.bytes        | cumulative | B         | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| disk.read.bytes.rate   | gauge      | B/s       | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| disk.read.requests     | cumulative | request   | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| disk.read.requests.rate| gauge      | request/s | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| disk.root.size         | gauge      | GB        | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| disk.write.bytes       | cumulative | B         | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| disk.write.bytes.rate  | gauge      | B/s       | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| disk.write.requests    | cumulative | request   | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| disk.write.requests.rate| gauge     | request/s | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| instance               | gauge      | instance  | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| instance:m1.tiny       | gauge      | instance  | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| memory                 | gauge      | MB        | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
| vcpus                  | gauge      | vcpu      | bb52e52b-1e42-4751-b3ac-45c52d83ba07 | b6e62aad26174382bc3781c12fe413c8 | cbfa8e3dfab64a27a87c8e24ecd5c60f |
```

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

As it was described above, the whole set of samples can be retrieved that are stored for a meter or filtering the result set by using one of the available query types. The request for all the samples of the `cpu` meter without any additional filtering looks like the following:

```
$ ceilometer sample-list --meter cpu
+--------------------------------------+-------+------------+------------+------+---------------------+
| Resource ID                          | Meter | Type       | Volume     | Unit | Timestamp           |
+--------------------------------------+-------+------------+------------+------+---------------------+
| c8d2e153-a48f-4cec-9e93-86e7ac6d4b0b | cpu   | cumulative | 5.4863e+11 | ns   | 2014-08-31T11:17:03 |
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | cpu   | cumulative | 5.7848e+11 | ns   | 2014-08-31T11:17:03 |
| c8d2e153-a48f-4cec-9e93-86e7ac6d4b0b | cpu   | cumulative | 5.4811e+11 | ns   | 2014-08-31T11:07:05 |
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | cpu   | cumulative | 5.7797e+11 | ns   | 2014-08-31T11:07:05 |
| c8d2e153-a48f-4cec-9e93-86e7ac6d4b0b | cpu   | cumulative | 5.3589e+11 | ns   | 2014-08-31T10:27:19 |
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | cpu   | cumulative | 5.6397e+11 | ns   | 2014-08-31T10:27:19 |
| ...                                  |       |            |            |      |                     |
+--------------------------------------+-------+------------+------------+------+---------------------+
```

The result set of the request contains the samples for both instances ordered by the timestamp field in the default descending order.

The simple query makes it possible to retrieve only a subset of the collected samples. The following command can be executed to request the `cpu` samples of only one of the VM instances:

```
$ ceilometer sample-list --meter cpu --query resource=bb52e52b-1e42-4751-
  b3ac-45c52d83ba07
+--------------------------------------+------+------------+------------+------+---------------------+
| Resource ID                          | Name | Type       | Volume     | Unit | Timestamp           |
+--------------------------------------+------+------------+------------+------+---------------------+
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | cpu  | cumulative | 5.7906e+11 | ns   | 2014-08-31T11:27:08 |
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | cpu  | cumulative | 5.7848e+11 | ns   | 2014-08-31T11:17:03 |
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | cpu  | cumulative | 5.7797e+11 | ns   | 2014-08-31T11:07:05 |
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | cpu  | cumulative | 5.6397e+11 | ns   | 2014-08-31T10:27:19 |
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | cpu  | cumulative | 5.6207e+11 | ns   | 2014-08-31T10:17:03 |
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | cpu  | cumulative | 5.3831e+11 | ns   | 2014-08-31T08:41:57 |
| ...                                  |      |            |            |      |                     |
+--------------------------------------+------+------------+------------+------+---------------------+
```

As it can be seen on the output above, the result set contains samples for only one instance of the two.

The **ceilometer query-samples** command is used to execute rich queries. This command accepts the following parameters:

`--filter`
: Contains the filter expression for the query in the form of: `{complex_op: [{simple_op: {field_name: value}}]}`.

`--orderby`
: Contains the list of `orderby` expressions in the form of: `[{field_name: direction}, {field_name: direction}]`.

`--limit`
: Specifies the maximum number of samples to return.

For more information about complex queries see *Complex query*.

As the complex query functionality provides the possibility of using complex operators, it is possible to retrieve a subset of samples for a given VM instance. To request for the first six samples for the `cpu` and `disk.read.bytes` meters, the following command should be invoked:

```
$ ceilometer query-samples --filter '{"and": \
  [{"=":{"resource":"bb52e52b-1e42-4751-b3ac-45c52d83ba07"}},{"or":[{"=":{"counter_name":"cpu"}}, \
  {"=":{"counter_name":"disk.read.bytes"}}]}]}' --orderby '[{"timestamp":"asc"}]' --limit 6
+--------------------------------------+-----------------+------------+------------+------+---------------------+
| Resource ID                          | Meter           | Type       | Volume     | Unit | Timestamp           |
+--------------------------------------+-----------------+------------+------------+------+---------------------+
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | disk.read.bytes | cumulative | 385334.0   | B    | 2014-08-30T13:00:46 |
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | cpu             | cumulative | 1.2132e+11 | ns   | 2014-08-30T13:00:47 |
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | cpu             | cumulative | 1.4295e+11 | ns   | 2014-08-30T13:10:51 |
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | disk.read.bytes | cumulative | 601438.0   | B    | 2014-08-30T13:10:51 |
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | disk.read.bytes | cumulative | 601438.0   | B    | 2014-08-30T13:20:33 |
| bb52e52b-1e42-4751-b3ac-45c52d83ba07 | cpu             | cumulative | 1.4795e+11 | ns   | 2014-08-30T13:20:34 |
+--------------------------------------+-----------------+------------+------------+------+---------------------+
```

Ceilometer also captures data as events, which represents the state of a resource. Refer to `/telemetry-events` for more information regarding Events.

To retrieve a list of recent events that occurred in the system, the following command can be executed:

```
$ ceilometer event-list
+--------------------------------------+--------------+----------------------------+-------------------------------------------------------------------+
| Message ID                           | Event Type   | Generated                  | Traits                                                            |
+--------------------------------------+--------------+----------------------------+-------------------------------------------------------------------+
| dfdb87b6-92c6-4d40-b9b5-ba308f304c13 | image.create | 2015-09-24T22:17:39.498888 | +---------+--------+-----------------+                              |
|                                      |              |                            | |   name  |  type  |      value      |                              |
|                                      |              |                            | +---------+--------+-----------------+                              |
|                                      |              |                            | | service | string | image.localhost |                              |
|                                      |              |                            | +---------+--------+-----------------+                              |
| 84054bc6-2ae6-4b93-b5e7-06964f151cef | image.prepare | 2015-09-24T22:17:39.594192 | +---------+--------+-----------------+                              |
|                                      |              |                            | |   name  |  type  |      value      |                              |
|                                      |              |                            | +---------+--------+-----------------+                              |
|                                      |              |                            | | service | string | image.localhost |                              |
|                                      |              |                            | +---------+--------+-----------------+                              |
| 2ec99c2c-08ee-4079-bf80-27d4a073ded6 | image.update | 2015-09-24T22:17:39.578336 | +-------------+--------+-----------------------------------------+ |
|                                      |              |                            | |     name    |  type  |                value                | |
|                                      |              |                            | +-------------+--------+-----------------------------------------+ |
|                                      |              |                            | |  created_at | string |         2015-09-24T22:17:39Z         | |
|                                      |              |                            | |     name    | string |    cirros-0.3.4-x86_64-uec-kernel    | |
|                                      |              |                            | |  project_id | string |   56ffddea5b4f423496444ea36c31be23   | |
|                                      |              |                            | | resource_id | string | 86eb8273-edd7-4483-a07c-002ff1c5657d | |
|                                      |              |                            | |   service   | string |           image.localhost            | |
|                                      |              |                            | |    status   | string |                saving                | |
|                                      |              |                            | |   user_id   | string |   56ffddea5b4f423496444ea36c31be23   | |
|                                      |              |                            | +-------------+--------+-----------------------------------------+ |
+--------------------------------------+--------------+----------------------------+-------------------------------------------------------------------+
```

> **Note**
>
> In Liberty, the data returned corresponds to the role and user. Non-admin users will only return events that are scoped to them. Admin users will return all events related to the project they administer as well as all unscoped events.

Similar to querying meters, additional filter parameters can be given to retrieve specific events:

```
$ ceilometer event-list -q 'event_type=compute.instance.exists; \
  instance_type=m1.tiny'
+--------------------------------------+----------------------+----------------------------+-------------------------------------------------------------------------------+
| Message ID                           | Event Type           | Generated                  | Traits                                                                        |
+--------------------------------------+----------------------+----------------------------+-------------------------------------------------------------------------------+
| 134a2ab3-6051-496c-b82f-10a3c367439a | compute.instance.exists | 2015-09-25T03:00:02.152041 | +----------------------+---------+--------------------------------------+ |
|                                      |                      |                            | |         name         |  type   |                value                 | |
|                                      |                      |                            | +----------------------+---------+--------------------------------------+ |
|                                      |                      |                            | | audit_period_beginning | datetime |          2015-09-25T02:00:00          | |
|                                      |                      |                            | |  audit_period_ending   | datetime |          2015-09-25T03:00:00          | |
|                                      |                      |                            | |       disk_gb          | integer  |                  1                    | |
|                                      |                      |                            | |     ephemeral_gb       | integer  |                  0                    | |
|                                      |                      |                            | |         host           |  string  |         localhost.localdomain         | |
|                                      |                      |                            | |     instance_id        |  string  |  2115f189-c7f1-4228-97bc-d742600839f2 | |
|                                      |                      |                            | |    instance_type       |  string  |               m1.tiny                 | |
|                                      |                      |                            | |   instance_type_id     | integer  |                  2                    | |
|                                      |                      |                            | |     launched_at        | datetime |          2015-09-24T22:24:56          | |
|                                      |                      |                            | |      memory_mb         | integer  |                 512                   | |
|                                      |                      |                            | |      project_id        |  string  |    56ffddea5b4f423496444ea36c31be23   | |
|                                      |                      |                            | |      request_id        |  string  | req-c6292b21-bf98-4a1d-b40c-cebba4d09a67 | |
|                                      |                      |                            | |       root_gb          | integer  |                  1                    | |
|                                      |                      |                            | |       service          |  string  |               compute                 | |
|                                      |                      |                            | |        state           |  string  |                active                 | |
|                                      |                      |                            | |      tenant_id         |  string  |    56ffddea5b4f423496444ea36c31be23   | |
|                                      |                      |                            | |       user_id          |  string  |    0b3d725756f94923b9d0c4db864d06a9   | |
|                                      |                      |                            | |        vcpus           | integer  |                  1                    | |
|                                      |                      |                            | +----------------------+---------+--------------------------------------+ |
+--------------------------------------+----------------------+----------------------------+-------------------------------------------------------------------------------+
```

> **Note**
>
> As of the Liberty release, the number of items returned will be restricted to the value defined by `default_api_return_limit` in the `ceilometer.conf` configuration file. Alternatively, the value can be set per query by passing the `limit` option in the request.

Telemetry Python bindings

The command-line client library provides python bindings in order to use the Telemetry Python API directly from python programs.

The first step in setting up the client is to create a client instance with the proper credentials:

```
>>> import ceilometerclient.client
>>> cclient = ceilometerclient.client.get_client(VERSION, username=USERNAME,
password=PASSWORD, tenant_name=PROJECT_NAME, auth_url=AUTH_URL)
```

The `VERSION` parameter can be `1` or `2`, specifying the API version to be used.

The method calls look like the following:

```
>>> cclient.meters.list()
 [<Meter ...>, ...]

>>> cclient.samples.list()
 [<Sample ...>, ...]
```

For further details about the python-ceilometerclient package, see the Python bindings to the OpenStack Ceilometer API reference.

Publishers

The Telemetry service provides several transport methods to forward the data collected to the `ceilometer-collector` service or to an external system. The consumers of this data are widely different, like monitoring systems, for which data loss is acceptable and billing systems, which require reliable data transportation. Telemetry provides methods to fulfill the requirements of both kind of systems, as it is described below.

The publisher component makes it possible to persist the data into storage through the message bus or to send it to one or more external consumers. One chain can contain multiple publishers.

To solve the above mentioned problem, the notion of multi-publisher can be configured for each datapoint within the Telemetry service, allowing the same technical meter or event to be published multiple times to multiple destinations, each potentially using a different transport.

Publishers can be specified in the `publishers` section for each pipeline (for further details about pipelines see *Data collection and processing*) that is defined in the pipeline.yaml file.

The following publisher types are supported:

notifier
: It can be specified in the form of `notifier://?option1=value1&option2=value2`. It emits data over AMQP using oslo.messaging. This is the recommended method of publishing.

rpc
: It can be specified in the form of `rpc://?option1=value1&option2=value2`. It emits metering data over lossy AMQP. This method is synchronous and may experience performance issues. This publisher is deprecated in Liberty in favor of the notifier publisher.

udp
: It can be specified in the form of `udp://<host>:<port>/`. It emits metering data for over UDP.

file
: It can be specified in the form of `file://path?option1=value1&option2=value2`. This publisher records metering data into a file.

> **Note**
>
> If a file name and location is not specified, this publisher does not log any meters, instead it logs a warning message in the configured log file for Telemetry.

kafka

It can be specified in the form of: `kafka://kafka_broker_ip: kafka_broker_port?topic=kafka_topic &option1=value1`.

This publisher sends metering data to a kafka broker.

> **Note**
>
> If the topic parameter is missing, this publisher brings out metering data under a topic name, `ceilometer`. When the port number is not specified, this publisher uses 9092 as the broker's port.

The following options are available for `rpc` and `notifier`. The policy option can be used by `kafka` publisher:

`per_meter_topic`
The value of it is 1. It is used for publishing the samples on additional `metering_topic.sample_name` topic queue besides the default `metering_topic` queue.

`policy`

It is used for configuring the behavior for the case, when the publisher fails to send the samples, where the possible predefined values are the following:

default
Used for waiting and blocking until the samples have been sent.

drop
Used for dropping the samples which are failed to be sent.

queue
Used for creating an in-memory queue and retrying to send the samples on the queue on the next samples publishing period (the queue length can be configured with `max_queue_length`, where 1024 is the default value).

The following option is additionally available for the `notifier` publisher:

`topic`
The topic name of queue to publish to. Setting this will override the default topic defined by `metering_topic` and `event_topic` options. This option can be used to support multiple consumers. Support for this feature was added in Kilo.

The following options are available for the `file` publisher:

max_bytes
When this option is greater than zero, it will cause a rollover. When the size is about to be exceeded, the file is closed and a new file is silently opened for output. If its value is zero, rollover never occurs.

backup_count
If this value is non-zero, an extension will be appended to the filename of the old log, as '.1', '.2', and so forth until the specified value is reached. The file that is written and contains the newest data is always the one that is specified without any extensions.

The default publisher is `notifier`, without any additional options specified. A sample `publishers` section in the `/etc/ceilometer/pipeline.yaml` looks like the following:

```
publishers:
    - udp://10.0.0.2:1234
    - rpc://?per_meter_topic=1 (deprecated in Liberty)
    - notifier://?policy=drop&max_queue_length=512&topic=custom_target
```

Alarms

Alarms provide user-oriented Monitoring-as-a-Service for resources running on OpenStack. This type of monitoring ensures you can automatically scale in or out a group of instances through the Orchestration service, but you can also use alarms for general-purpose awareness of your cloud resources' health.

These alarms follow a tri-state model:

ok
The rule governing the alarm has been evaluated as `False`.

alarm
The rule governing the alarm have been evaluated as `True`.

insufficient data
There are not enough datapoints available in the evaluation periods to meaningfully determine the alarm state.

Alarm definitions

The definition of an alarm provides the rules that govern when a state transition should occur, and the actions to be taken thereon. The nature of these rules depend on the alarm type.

Threshold rule alarms

For conventional threshold-oriented alarms, state transitions are governed by:

- A static threshold value with a comparison operator such as greater than or less than.
- A statistic selection to aggregate the data.
- A sliding time window to indicate how far back into the recent past you want to look.

Combination rule alarms

The Telemetry service also supports the concept of a meta-alarm, which aggregates over the current state of a set of underlying basic alarms combined via a logical operator (AND or OR).

Alarm dimensioning

A key associated concept is the notion of *dimensioning* which defines the set of matching meters that feed into an alarm evaluation. Recall that meters are per-resource-instance, so in the simplest case an alarm might be defined over a particular meter applied to all resources visible to a particular user. More useful however would be the option to explicitly select which specific resources you are interested in alarming on.

At one extreme you might have narrowly dimensioned alarms where this selection would have only a single target (identified by resource ID). At the other extreme, you could have widely dimensioned alarms where this selection identifies many resources over which the statistic is aggregated. For example all instances booted from a particular image or all instances with matching user metadata (the latter is how the Orchestration service identifies autoscaling groups).

Alarm evaluation

Alarms are evaluated by the `alarm-evaluator` service on a periodic basis, defaulting to once every minute.

Alarm actions

Any state transition of individual alarm (to `ok`, `alarm`, or `insufficient data`) may have one or more actions associated with it. These actions effectively send a signal to a consumer that the state transition has occurred, and provide some additional context. This includes the new and previous states, with some reason data describing the disposition with respect to the threshold, the number of datapoints involved and most recent of these. State transitions are detected by the `alarm-evaluator`, whereas the `alarm-notifier` effects the actual notification action.

Webhooks

These are the *de facto* notification type used by Telemetry alarming and simply involve an HTTP POST request being sent to an endpoint, with a request body containing a description of the state transition encoded as a JSON fragment.

Log actions

These are a lightweight alternative to webhooks, whereby the state transition is simply logged by the `alarm-notifier`, and are intended primarily for testing purposes.

Workload partitioning

The alarm evaluation process uses the same mechanism for workload partitioning as the central and compute agents. The Tooz library provides the coordination within the groups of service instances. For further information about this approach, see the section called *Support for HA deployment of the central and compute agent services*.

To use this workload partitioning solution set the `evaluation_service` option to `default`. For more information, see the alarm section in the OpenStack Configuration Reference.

Using alarms

Alarm creation

An example of creating a threshold-oriented alarm, based on an upper bound on the CPU utilization for a particular instance:

```
$ ceilometer alarm-threshold-create --name cpu_hi \
  --description 'instance running hot' \
  --meter-name cpu_util --threshold 70.0 \
  --comparison-operator gt --statistic avg \
  --period 600 --evaluation-periods 3 \
  --alarm-action 'log://' \
  --query resource_id=INSTANCE_ID
```

This creates an alarm that will fire when the average CPU utilization for an individual instance exceeds 70% for three consecutive 10 minute periods. The notification in this case is simply a log message, though it could alternatively be a webhook URL.

> **Note**
>
> Alarm names must be unique for the alarms associated with an individual project. Administrator can limit the maximum resulting actions for three different states, and the ability for a normal user to create `log://` and `test://` notifiers is disabled. This

> prevents unintentional consumption of disk and memory resources by the Telemetry service.

The sliding time window over which the alarm is evaluated is 30 minutes in this example. This window is not clamped to wall-clock time boundaries, rather it's anchored on the current time for each evaluation cycle, and continually creeps forward as each evaluation cycle rolls around (by default, this occurs every minute).

The period length is set to 600s in this case to reflect the out-of-the-box default cadence for collection of the associated meter. This period matching illustrates an important general principal to keep in mind for alarms:

> **Note**
>
> The alarm period should be a whole number multiple (1 or more) of the interval configured in the pipeline corresponding to the target meter.

Otherwise the alarm will tend to flit in and out of the `insufficient data` state due to the mismatch between the actual frequency of datapoints in the metering store and the statistics queries used to compare against the alarm threshold. If a shorter alarm period is needed, then the corresponding interval should be adjusted in the `pipeline.yaml` file.

Other notable alarm attributes that may be set on creation, or via a subsequent update, include:

state
: The initial alarm state (defaults to `insufficient data`).

description
: A free-text description of the alarm (defaults to a synopsis of the alarm rule).

enabled
: True if evaluation and actioning is to be enabled for this alarm (defaults to `True`).

repeat-actions
: True if actions should be repeatedly notified while the alarm remains in the target state (defaults to `False`).

ok-action
: An action to invoke when the alarm state transitions to `ok`.

insufficient-data-action
: An action to invoke when the alarm state transitions to `insufficient data`.

time-constraint
: Used to restrict evaluation of the alarm to certain times of the day or days of the week (expressed as `cron` expression with an optional timezone).

An example of creating a combination alarm, based on the combined state of two underlying alarms:

```
$ ceilometer alarm-combination-create --name meta \
  --alarm_ids ALARM_ID1 \
  --alarm_ids ALARM_ID2 \
  --operator or \
  --alarm-action 'http://example.org/notify'
```

This creates an alarm that will fire when either one of two underlying alarms transition into the alarm state. The notification in this case is a webhook call. Any number of underlying alarms can be combined in this way, using either `and` or `or`.

Alarm retrieval

You can display all your alarms via (some attributes are omitted for brevity):

```
$ ceilometer alarm-list
+----------+--------+-------------------+---------------------------------+
| Alarm ID | Name   | State             | Alarm condition                 |
+----------+--------+-------------------+---------------------------------+
| ALARM_ID | cpu_hi | insufficient data | cpu_util > 70.0 during 3 x 600s |
+----------+--------+-------------------+---------------------------------+
```

In this case, the state is reported as `insufficient data` which could indicate that:

- meters have not yet been gathered about this instance over the evaluation window into the recent past (for example a brand-new instance)
- *or*, that the identified instance is not visible to the user/tenant owning the alarm
- *or*, simply that an alarm evaluation cycle hasn't kicked off since the alarm was created (by default, alarms are evaluated once per minute).

> **Note**
>
> The visibility of alarms depends on the role and project associated with the user issuing the query:
>
> - admin users see *all* alarms, regardless of the owner
> - non-admin users see only the alarms associated with their project (as per the normal tenant segregation in OpenStack)

Alarm update

Once the state of the alarm has settled down, we might decide that we set that bar too low with 70%, in which case the threshold (or most any other alarm attribute) can be updated thusly:

```
$ ceilometer alarm-update --threshold 75 ALARM_ID
```

The change will take effect from the next evaluation cycle, which by default occurs every minute.

Most alarm attributes can be changed in this way, but there is also a convenient short-cut for getting and setting the alarm state:

```
$ ceilometer alarm-state-get ALARM_ID
$ ceilometer alarm-state-set --state ok -a ALARM_ID
```

Over time the state of the alarm may change often, especially if the threshold is chosen to be close to the trending value of the statistic. You can follow the history of an alarm over its lifecycle via the audit API:

```
$ ceilometer alarm-history ALARM_ID
+------------------+-----------+---------------------------------------+
| Type             | Timestamp | Detail                                |
+------------------+-----------+---------------------------------------+
| creation         | time0     | name: cpu_hi                          |
|                  |           | description: instance running hot     |
|                  |           | type: threshold                       |
|                  |           | rule: cpu_util > 70.0 during 3 x 600s |
| state transition | time1     | state: ok                             |
| rule change      | time2     | rule: cpu_util > 75.0 during 3 x 600s |
+------------------+-----------+---------------------------------------+
```

Alarm deletion

An alarm that is no longer required can be disabled so that it is no longer actively evaluated:

```
$ ceilometer alarm-update --enabled False -a ALARM_ID
```

or even deleted permanently (an irreversible step):

```
$ ceilometer alarm-delete ALARM_ID
```

> **Note**
>
> By default, alarm history is retained for deleted alarms.

Measurements

The Telemetry service collects meters within an OpenStack deployment. This section provides a brief summary about meters format and origin and also contains the list of available meters.

Telemetry collects meters by polling the infrastructure elements and also by consuming the notifications emitted by other OpenStack services. For more information about the

polling mechanism and notifications see *Data collection*. There are several meters which are collected by polling and by consuming. The origin for each meter is listed in the tables below.

> **Note**
>
> You may need to configure Telemetry or other OpenStack services in order to be able to collect all the samples you need. For further information about configuration requirements see the Telemetry chapter in the OpenStack Installation Guide. Also check the Telemetry manual installation description.

Telemetry uses the following meter types:

Type	Description
Cumulative	Increasing over time (instance hours)
Delta	Changing over time (bandwidth)
Gauge	Discrete items (floating IPs, image uploads) and fluctuating values (disk I/O)

Telemetry provides the possibility to store metadata for samples. This metadata can be extended for OpenStack Compute and OpenStack Object Storage.

In order to add additional metadata information to OpenStack Compute you have two options to choose from. The first one is to specify them when you boot up a new instance. The additional information will be stored with the sample in the form of `resource_metadata.user_metadata.*`. The new field should be defined by using the prefix `metering.`. The modified boot command look like the following:

```
$ nova boot --meta metering.custom_metadata=a_value my_vm
```

The other option is to set the `reserved_metadata_keys` to the list of metadata keys that you would like to be included in `resource_metadata` of the instance related samples that are collected for OpenStack Compute. This option is included in the `DEFAULT` section of the `ceilometer.conf` configuration file.

You might also specify headers whose values will be stored along with the sample data of OpenStack Object Storage. The additional information is also stored under `resource_metadata`. The format of the new field is `resource_metadata.http_header_$name`, where `$name` is the name of the header with `-` replaced by `_`.

For specifying the new header, you need to set `metadata_headers` option under the

`[filter:ceilometer]` section in `proxy-server.conf` under the `swift` folder. You can use this additional data for instance to distinguish external and internal users.

Measurements are grouped by services which are polled by Telemetry or emit notifications that this service consumes.

> **Note**
>
> The Telemetry service supports storing notifications as events. This functionality was added later, therefore the list of meters still contains existence type and other event related items. The proper way of using Telemetry is to configure it to use the event store and turn off the collection of the event related meters. For further information about events see Events section in the Telemetry documentation. For further information about how to turn on and off meters see *Pipeline configuration*. Please also note that currently no migration is available to move the already existing event type samples to the event store.

OpenStack Compute

The following meters are collected for OpenStack Compute:

Name	Type	Unit	Resource	Origin	Support	Note
Meters added in the Icehouse release or earlier						
instance	Gauge	instance	instance ID	Notification, Pollster	Libvirt, Hyper-V, vSphere	Existence of instance
instance:<type>	Gauge	instance	instance ID	Notification, Pollster	Libvirt, Hyper-V, vSphere	Existence of instance <type> (OpenStack types)
memory	Gauge	MB	instance ID	Notification	Libvirt, Hyper-V	Volume of RAM allocated to the instance
memory.usage	Gauge	MB	instance ID	Pollster	vSphere	Volume of RAM used by the instance from the amount of its allocated memory
cpu	Cumulative	ns	instance ID	Pollster	Libvirt, Hyper-V	CPU time used
cpu_util	Gauge	%	instance ID	Pollster	vSphere	Average CPU utilization
vcpus	Gauge	vcpu	instance ID	Notification	Libvirt, Hyper-V	Number of virtual CPUs allocated to

Name	Type	Unit	Resource	Origin	Support	Note
						the instance
disk.read.requests	Cumulative	request	instance ID	Pollster	Libvirt, Hyper-V	Number of read requests
disk.read.requests.rate	Gauge	request/s	instance ID	Pollster	Libvirt, Hyper-V, vSphere	Average rate of read requests
disk.write.requests	Cumulative	request	instance ID	Pollster	Libvirt, Hyper-V	Number of write requests
disk.write.requests.rate	Gauge	request/s	instance ID	Pollster	Libvirt, Hyper-V, vSphere	Average rate of write requests
disk.read.bytes	Cumulative	B	instance ID	Pollster	Libvirt, Hyper-V	Volume of reads
disk.read.bytes.rate	Gauge	B/s	instance ID	Pollster	Libvirt, Hyper-V, vSphere	Average rate of reads
disk.write.bytes	Cumulative	B	instance ID	Pollster	Libvirt, Hyper-V	Volume of writes
disk.write.bytes.rate	Gauge	B/s	instance ID	Pollster	Libvirt, Hyper-V, vSphere	Average rate of writes
disk.root.size	Gauge	GB	instance ID	Notification	Libvirt, Hyper-V	Size of root disk
disk.ephemeral.size	Gauge	GB	instance ID	Notification	Libvirt, Hyper-V	Size of ephemeral disk
network.incoming.bytes	Cumulative	B	interface ID	Pollster	Libvirt, Hyper-V	Number of incoming bytes
network.incoming.bytes.rate	Gauge	B/s	interface ID	Pollster	Libvirt, Hyper-V, vSphere	Average rate of incoming bytes
network.outgoing.bytes	Cumulative	B	interface ID	Pollster	Libvirt, Hyper-V	Number of outgoing bytes
network.outgoing.bytes.rate	Gauge	B/s	interface ID	Pollster	Libvirt, Hyper-V, vSphere	Average rate of outgoing bytes
network.incoming.packets	Cumulative	packet	interface ID	Pollster	Libvirt, Hyper-V	Number of incoming packets
network.incoming.packets.rate	Gauge	packet/s	interface ID	Pollster	Libvirt, Hyper-V, vSphere	Average rate of incoming packets
network.outgoing.packets	Cumulative	packet	interface ID	Pollster	Libvirt, Hyper-V	Number of outgoing packets
network.outgoing.packets.rate	Gauge	packet/s	interface ID	Pollster	Libvirt, Hyper-V, vSphere	Average rate of outgoing packets
Meters added or hypervisor support changed in the Juno release						

Name	Type	Unit	Resource	Origin	Support	Note
instance	Gauge	instance	instance ID	Notification, Pollster	Libvirt, Hyper-V, vSphere, XenAPI	Existence of instance
instance:<type>	Gauge	instance	instance ID	Notification, Pollster	Libvirt, Hyper-V, vSphere, XenAPI	Existence of instance <type> (OpenStack types)
memory.usage	Gauge	MB	instance ID	Pollster	vSphere, XenAPI	Volume of RAM used by the instance from the amount of its allocated memory
cpu_util	Gauge	%	instance ID	Pollster	vSphere, XenAPI	Average CPU utilization
disk.read.bytes.rate	Gauge	B/s	instance ID	Pollster	Libvirt, Hyper-V, vSphere, XenAPI	Average rate of reads
disk.write.bytes.rate	Gauge	B/s	instance ID	Pollster	Libvirt, Hyper-V, vSphere, XenAPI	Average rate of writes
disk.device.read.requests	Cumulative	request	disk ID	Pollster	Libvirt, Hyper-V	Number of read requests
disk.device.read.requests.rate	Gauge	request/s	disk ID	Pollster	Libvirt, Hyper-V, vSphere	Average rate of read requests
disk.device.write.requests	Cumulative	request	disk ID	Pollster	Libvirt, Hyper-V	Number of write requests
disk.device.write.requests.rate	Gauge	request/s	disk ID	Pollster	Libvirt, Hyper-V, vSphere	Average rate of write requests
disk.device.read.bytes	Cumulative	B	disk ID	Pollster	Libvirt, Hyper-V	Volume of reads
disk.device.read.bytes .rate	Gauge	B/s	disk ID	Pollster	Libvirt, Hyper-V, vSphere	Average rate of reads
disk.device.write.bytes	Cumulative	B	disk ID	Pollster	Libvirt, Hyper-V	Volume of writes
disk.device.write.bytes .rate	Gauge	B/s	disk ID	Pollster	Libvirt, Hyper-V, vSphere	Average rate of writes
network.incoming.bytes.rate	Gauge	B/s	interface ID	Pollster	Libvirt, Hyper-V, vSphere, XenAPI	Average rate of incoming bytes

Name	Type	Unit	Resource	Origin	Support	Note
network.outgoing.bytes.rate	Gauge	B/s	interface ID	Pollster	Libvirt, Hyper-V, vSphere, XenAPI	Average rate of outgoing bytes
network.incoming.packets.rate	Gauge	packet/s	interface ID	Pollster	Libvirt, Hyper-V, vSphere, XenAPI	Average rate of incoming packets
network.outgoing.packets.rate	Gauge	packet/s	interface ID	Pollster	Libvirt, Hyper-V, vSphere, XenAPI	Average rate of outgoing packets
Meters added or hypervisor support changed in the Kilo release						
memory.usage	Gauge	MB	instance ID	Pollster	Libvirt, Hyper-V, vSphere, XenAPI	Volume of RAM used by the instance from the amount of its allocated memory
memory.resident	Gauge	MB	instance ID	Pollster	Libvirt	Volume of RAM used by the instance on the physical machine
disk.latency	Gauge	ms	instance ID	Pollster	Hyper-V	Average disk latency
disk.iops	Gauge	count/s	instance ID	Pollster	Hyper-V	Average disk iops
disk.device.latency	Gauge	ms	disk ID	Pollster	Hyper-V	Average disk latency per device
disk.device.iops	Gauge	count/s	disk ID	Pollster	Hyper-V	Average disk iops per device
disk.capacity	Gauge	B	instance ID	Pollster	Libvirt	The amount of disk that the instance can see
disk.allocation	Gauge	B	instance ID	Pollster	Libvirt	The amount of disk occupied by the instance on the host machine
disk.usage	Gauge	B	instance ID	Pollster	Libvirt	The physical size in bytes of the image container on the host

Name	Type	Unit	Resource	Origin	Support	Note
disk.device.capacity	Gauge	B	disk ID	Pollster	Libvirt	The amount of disk per device that the instance can see
disk.device.allocation	Gauge	B	disk ID	Pollster	Libvirt	The amount of disk per device occupied by the instance on the host machine
disk.device.usage	Gauge	B	disk ID	Pollster	Libvirt	The physical size in bytes of the image container on the host per device
Meters deprecated in the Kilo release						
instance:<type>	Gauge	instance	instance ID	Notification, Pollster	Libvirt, Hyper-V, vSphere, XenAPI	Existence of instance <type> (OpenStack types)
Meters added in the Liberty release						
cpu.delta	Delta	ns	instance ID	Pollster	Libvirt, Hyper-V	CPU time used since previous datapoint
Meters added in the Newton release						
cpu_l3_cache	Gauge	B	instance ID	Pollster	Libvirt	L3 cache used by the instance

> **Note**
>
> The `instance:<type>` meter can be replaced by using extra parameters in both the samples and statistics queries. Sample queries look like:
>
> statistics: ceilometer statistics -m instance -g resource_metadata.instance_type
> samples: ceilometer sample-list -m instance -q metadata.instance_type=<value>

The Telemetry service supports to create new meters by using transformers. For more details about transformers see *Transformers*. Among the meters gathered from libvirt and Hyper-V there are a few ones which are generated from other meters. The list of meters that are created by using the `rate_of_change` transformer from the above table is the following:

- cpu_util
- disk.read.requests.rate
- disk.write.requests.rate
- disk.read.bytes.rate
- disk.write.bytes.rate
- disk.device.read.requests.rate
- disk.device.write.requests.rate
- disk.device.read.bytes.rate
- disk.device.write.bytes.rate
- network.incoming.bytes.rate
- network.outgoing.bytes.rate
- network.incoming.packets.rate
- network.outgoing.packets.rate

> **Note**
>
> To enable the libvirt `memory.usage` support, you need to install libvirt version 1.1.1+, QEMU version 1.5+, and you also need to prepare suitable balloon driver in the image. It is applicable particularly for Windows guests, most modern Linux distributions already have it built in. Telemetry is not able to fetch the `memory.usage` samples without the image balloon driver.

OpenStack Compute is capable of collecting `CPU` related meters from the compute host machines. In order to use that you need to set the `compute_monitors` option to `ComputeDriverCPUMonitor` in the `nova.conf` configuration file. For further information see the Compute configuration section in the Compute chapter of the OpenStack Configuration Reference.

The following host machine related meters are collected for OpenStack Compute:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Icehouse release or earlier					
compute.node.cpu.frequency	Gauge	MHz	host ID	Notification	CPU frequency
compute.node.cpu.kernel.time	Cumulative	ns	host ID	Notification	CPU kernel time
compute.node.cpu.idle.time	Cumulative	ns	host ID	Notification	CPU idle time
compute.node.cpu.user.time	Cumulative	ns	host ID	Notification	CPU user mode time
compute.node.cpu.iowait.time	Cumulative	ns	host ID	Notification	CPU I/O wait time

Name	Type	Unit	Resource	Origin	Note
compute.node.cpu.kernel.percent	Gauge	%	host ID	Notification	CPU kernel percentage
compute.node.cpu.idle.percent	Gauge	%	host ID	Notification	CPU idle percentage
compute.node.cpu.user.percent	Gauge	%	host ID	Notification	CPU user mode percentage
compute.node.cpu.iowait.percent	Gauge	%	host ID	Notification	CPU I/O wait percentage
compute.node.cpu.percent	Gauge	%	host ID	Notification	CPU utilization

Bare metal service

Telemetry captures notifications that are emitted by the Bare metal service. The source of the notifications are IPMI sensors that collect data from the host machine.

> **Note**
>
> The sensor data is not available in the Bare metal service by default. To enable the meters and configure this module to emit notifications about the measured values see the Installation Guide for the Bare metal service.

The following meters are recorded for the Bare metal service:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Juno release					
hardware.ipmi.fan	Gauge	RPM	fan sensor	Notification	Fan rounds per minute (RPM)
hardware.ipmi.fan	Gauge	RPM	fan sensor	Notification	Fan rounds per minute (RPM)
hardware.ipmi.temperature	Gauge	C	temperature sensor	Notification	Temperate reading from sensor
hardware.ipmi.current	Gauge	W	current sensor	Notification	Current reading from sensor
hardware.ipmi.voltage	Gauge	V	voltage sensor	Notification	Voltage reading from sensor

IPMI based meters

Another way of gathering IPMI based data is to use IPMI sensors independently from the Bare metal service's components. Same meters as *Bare metal service* could be fetched except that origin is `Pollster` instead of `Notification`.

You need to deploy the ceilometer-agent-ipmi on each IPMI-capable node in order to poll local sensor data. For further information about the IPMI agent see *IPMI agent*.

> **Warning**
>
> To avoid duplication of metering data and unnecessary load on the IPMI interface, do not deploy the IPMI agent on nodes that are managed by the Bare metal service and keep the `conductor.send_sensor_data` option set to `False` in the `ironic.conf` configuration file.

Besides generic IPMI sensor data, the following Intel Node Manager meters are recorded from capable platform:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Juno release					
hardware.ipmi.node.power	Gauge	W	host ID	Pollster	Current power of the system
hardware.ipmi.node.temperature	Gauge	C	host ID	Pollster	Current temperature of the system
Meters added in the Kilo release					
hardware.ipmi.node.inlet_temperature	Gauge	C	host ID	Pollster	Inlet temperature of the system
hardware.ipmi.node.outlet_temperature	Gauge	C	host ID	Pollster	Outlet temperature of the system
hardware.ipmi.node.airflow	Gauge	CFM	host ID	Pollster	Volumetric airflow of the system, expressed as 1/10th of CFM
hardware.ipmi.node.cups	Gauge	CUPS	host ID	Pollster	CUPS(Compute Usage Per Second) index data of the system

Name	Type	Unit	Resource	Origin	Note
hardware.ipmi.node.cpu_util	Gauge	%	host ID	Pollster	CPU CUPS utilization of the system
hardware.ipmi.node.mem_util	Gauge	%	host ID	Pollster	Memory CUPS utilization of the system
hardware.ipmi.node.io_util	Gauge	%	host ID	Pollster	IO CUPS utilization of the system

Meters renamed in the Kilo release	
Original Name	**New Name**
hardware.ipmi.node.temperature	hardware.ipmi.node.inlet_temperature

SNMP based meters

Telemetry supports gathering SNMP based generic host meters. In order to be able to collect this data you need to run snmpd on each target host.

The following meters are available about the host machines by using SNMP:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Kilo release					
hardware.cpu.load.1min	Gauge	process	host ID	Pollster	CPU load in the past 1 minute
hardware.cpu.load.5min	Gauge	process	host ID	Pollster	CPU load in the past 5 minutes
hardware.cpu.load.10min	Gauge	process	host ID	Pollster	CPU load in the past 10 minutes
hardware.disk.size.total	Gauge	KB	disk ID	Pollster	Total disk size
hardware.disk.size.used	Gauge	KB	disk ID	Pollster	Used disk size
hardware.memory.total	Gauge	KB	host ID	Pollster	Total physical memory size
hardware.memory.used	Gauge	KB	host ID	Pollster	Used physical memory size
hardware.memory.buffer	Gauge	KB	host ID	Pollster	Physical memory buffer size

Name	Type	Unit	Resource	Origin	Note
hardware.memory.cached	Gauge	KB	host ID	Pollster	Cached physical memory size
hardware.memory.swap.total	Gauge	KB	host ID	Pollster	Total swap space size
hardware.memory.swap.avail	Gauge	KB	host ID	Pollster	Available swap space size
hardware.network.incoming.bytes	Cumulative	B	interface ID	Pollster	Bytes received by network interface
hardware.network.outgoing.bytes	Cumulative	B	interface ID	Pollster	Bytes sent by network interface
hardware.network.outgoing.errors	Cumulative	packet	interface ID	Pollster	Sending error of network interface
hardware.network.ip.incoming.datagrams	Cumulative	datagrams	host ID	Pollster	Number of received datagrams
hardware.network.ip.outgoing.datagrams	Cumulative	datagrams	host ID	Pollster	Number of sent datagrams
hardware.system_stats.io.incoming.blocks	Cumulative	blocks	host ID	Pollster	Aggregated number of blocks received to block device
hardware.system_stats.io.outgoing.blocks	Cumulative	blocks	host ID	Pollster	Aggregated number of blocks sent to block device
hardware.system_stats.cpu.idle	Gauge	%	host ID	Pollster	CPU idle percentage

OpenStack Image service

The following meters are collected for OpenStack Image service:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Icehouse release or earlier					
image	Gauge	image	image ID	Notification, Pollster	Existence of the image
image.size	Gauge	image	image ID	Notification,	Size of the

Name	Type	Unit	Resource	Origin	Note
				Pollster	uploaded image
image.update	Delta	image	image ID	Notification	Number of updates on the image
image.upload	Delta	image	image ID	Notification	Number of uploads on the image
image.delete	Delta	image	image ID	Notification	Number of deletes on the image
image.download	Delta	B	image ID	Notification	Image is downloaded
image.serve	Delta	B	image ID	Notification	Image is served out

OpenStack Block Storage

The following meters are collected for OpenStack Block Storage:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Icehouse release or earlier					
volume	Gauge	volume	volume ID	Notification	Existence of the volume
volume.size	Gauge	GB	volume ID	Notification	Size of the volume
Meters added in the Juno release					
snapshot	Gauge	snapshot	snapshot ID	Notification	Existence of the snapshot
snapshot.size	Gauge	GB	snapshot ID	Notification	Size of the snapshot
Meters added in the Kilo release					
volume.create.(start|end)	Delta	volume	volume ID	Notification	Creation of the volume
volume.delete.(start|end)	Delta	volume	volume ID	Notification	Deletion of the volume
volume.update.(start|end)	Delta	volume	volume ID	Notification	Update the name or description of the volume
volume.resize.(start|end)	Delta	volume	volume ID	Notification	Update the size

Name	Type	Unit	Resource	Origin	Note
					of the volume
volume.attach.(start|end)	Delta	volume	volume ID	Notification	Attaching the volume to an instance
volume.detach.(start|end)	Delta	volume	volume ID	Notification	Detaching the volume from an instance
snapshot.create.(start|end)	Delta	snapshot	snapshot ID	Notification	Creation of the snapshot
snapshot.delete.(start|end)	Delta	snapshot	snapshot ID	Notification	Deletion of the snapshot
volume.backup.create.(start|end)	Delta	volume	backup ID	Notification	Creation of the volume backup
volume.backup.delete.(start|end)	Delta	volume	backup ID	Notification	Deletion of the volume backup
volume.backup.restore.(start|end)	Delta	volume	backup ID	Notification	Restoration of the volume backup

OpenStack Object Storage

The following meters are collected for OpenStack Object Storage:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Icehouse release or earlier					
storage.objects	Gauge	object	storage ID	Pollster	Number of objects
storage.objects.size	Gauge	B	storage ID	Pollster	Total size of stored objects
storage.objects.containers	Gauge	container	storage ID	Pollster	Number of containers
storage.objects.incoming.bytes	Delta	B	storage ID	Notification	Number of incoming bytes
storage.objects.outgoing.bytes	Delta	B	storage ID	Notification	Number of outgoing bytes
storage.api.request	Delta	request	storage ID	Notification	Number of API requests against

Name	Type	Unit	Resource	Origin	Note
					OpenStack Object Storage
storage.containers.objects	Gauge	object	storage ID/container	Pollster	Number of objects in container
storage.containers.objects.size	Gauge	B	storage ID/container	Pollster	Total size of stored objects in container

Ceph Object Storage

In order to gather meters from Ceph, you have to install and configure the Ceph Object Gateway (radosgw) as it is described in the Installation Manual. You have to enable usage logging in order to get the related meters from Ceph. You will also need an `admin` user with `users`, `buckets`, `metadata` and `usage caps` configured.

In order to access Ceph from Telemetry, you need to specify a `service group` for `radosgw` in the `ceilometer.conf` configuration file along with `access_key` and `secret_key` of the `admin` user mentioned above.

The following meters are collected for Ceph Object Storage:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Kilo release					
radosgw.objects	Gauge	object	storage ID	Pollster	Number of objects
radosgw.objects.size	Gauge	B	storage ID	Pollster	Total size of stored objects
radosgw.objects.containers	Gauge	container	storage ID	Pollster	Number of containers
radosgw.api.request	Gauge	request	storage ID	Pollster	Number of API requests against Ceph Object Gateway (radosgw)
radosgw.containers.objects	Gauge	object	storage ID/container	Pollster	Number of objects in container

Name	Type	Unit	Resource	Origin	Note
radosgw.containers.objects.size	Gauge	B	storage ID/container	Pollster	Total size of stored objects in container

Note

The usage related information may not be updated right after an upload or download, because the Ceph Object Gateway needs time to update the usage properties. For instance, the default configuration needs approximately 30 minutes to generate the usage logs.

OpenStack Identity

The following meters are collected for OpenStack Identity:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Juno release					
identity.authenticate.success	Delta	user	user ID	Notification	User successfully authenticated
identity.authenticate.pending	Delta	user	user ID	Notification	User pending authentication
identity.authenticate.failure	Delta	user	user ID	Notification	User failed to authenticate
identity.user.created	Delta	user	user ID	Notification	User is created
identity.user.deleted	Delta	user	user ID	Notification	User is deleted
identity.user.updated	Delta	user	user ID	Notification	User is updated
identity.group.created	Delta	group	group ID	Notification	Group is created
identity.group.deleted	Delta	group	group ID	Notification	Group is deleted
identity.group.updated	Delta	group	group ID	Notification	Group is updated
identity.role.created	Delta	role	role ID	Notification	Role is created
identity.role.deleted	Delta	role	role ID	Notification	Role is deleted
identity.role.updated	Delta	role	role ID	Notification	Role is updated
identity.project.created	Delta	project	project ID	Notification	Project is created
identity.project.deleted	Delta	project	project ID	Notification	Project is deleted
identity.project.updated	Delta	project	project ID	Notification	Project is updated
identity.trust.created	Delta	trust	trust ID	Notification	Trust is created
identity.trust.deleted	Delta	trust	trust ID	Notification	Trust is deleted
Meters added in the Kilo release					

Name	Type	Unit	Resource	Origin	Note
identity.role_assignment.created	Delta	role_assignment	role ID	Notification	Role is added to an actor on a target
identity.role_assignment.deleted	Delta	role_assignment	role ID	Notification	Role is removed from an actor on a target

OpenStack Networking

The following meters are collected for OpenStack Networking:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Icehouse release or earlier					
network	Gauge	network	network ID	Notification	Existence of network
network.create	Delta	network	network ID	Notification	Creation requests for this network
network.update	Delta	network	network ID	Notification	Update requests for this network
subnet	Gauge	subnet	subnet ID	Notification	Existence of subnet
subnet.create	Delta	subnet	subnet ID	Notification	Creation requests for this subnet
subnet.update	Delta	subnet	subnet ID	Notification	Update requests for this subnet
port	Gauge	port	port ID	Notification	Existence of port
port.create	Delta	port	port ID	Notification	Creation requests for this port
port.update	Delta	port	port ID	Notification	Update requests for this port
router	Gauge	router	router ID	Notification	Existence of router
router.create	Delta	router	router ID	Notification	Creation requests for this router
router.update	Delta	router	router ID	Notification	Update requests for this router
ip.floating	Gauge	ip	ip ID	Notification,	Existence of IP

Name	Type	Unit	Resource	Origin	Note
				Pollster	
ip.floating.create	Delta	ip	ip ID	Notification	Creation requests for this IP
ip.floating.update	Delta	ip	ip ID	Notification	Update requests for this IP
bandwidth	Delta	B	label ID	Notification	Bytes through this l3 metering label

SDN controllers

The following meters are collected for SDN:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Icehouse release or earlier					
switch	Gauge	switch	switch ID	Pollster	Existence of switch
switch.port	Gauge	port	switch ID	Pollster	Existence of port
switch.port.receive.packets	Cumulative	packet	switch ID	Pollster	Packets received on port
switch.port.transmit.packets	Cumulative	packet	switch ID	Pollster	Packets transmitted on port
switch.port.receive.bytes	Cumulative	B	switch ID	Pollster	Bytes received on port
switch.port.transmit.bytes	Cumulative	B	switch ID	Pollster	Bytes transmitted on port
switch.port.receive.drops	Cumulative	packet	switch ID	Pollster	Drops received on port
switch.port.transmit.drops	Cumulative	packet	switch ID	Pollster	Drops transmitted on port
switch.port.receive.errors	Cumulative	packet	switch ID	Pollster	Errors received on port
switch.port.transmit.errors	Cumulative	packet	switch ID	Pollster	Errors transmitted on port
switch.port.receive.frame_error	Cumulative	packet	switch ID	Pollster	Frame alignment errors received

Name	Type	Unit	Resource	Origin	Note
					on port
switch.port.receive.overrun_error	Cumulative	packet	switch ID	Pollster	Overrun errors received on port
switch.port.receive.crc_error	Cumulative	packet	switch ID	Pollster	CRC errors received on port
switch.port.collision.count	Cumulative	count	switch ID	Pollster	Collisions on port
switch.table	Gauge	table	switch ID	Pollster	Duration of table
switch.table.active.entries	Gauge	entry	switch ID	Pollster	Active entries in table
switch.table.lookup.packets	Gauge	packet	switch ID	Pollster	Lookup packets for table
switch.table.matched.packets	Gauge	packet	switch ID	Pollster	Packets matches for table
switch.flow	Gauge	flow	switch ID	Pollster	Duration of flow
switch.flow.duration.seconds	Gauge	s	switch ID	Pollster	Duration of flow in seconds
switch.flow.duration.nanoseconds	Gauge	ns	switch ID	Pollster	Duration of flow in nanoseconds
switch.flow.packets	Cumulative	packet	switch ID	Pollster	Packets received
switch.flow.bytes	Cumulative	B	switch ID	Pollster	Bytes received

These meters are available for OpenFlow based switches. In order to enable these meters, each driver needs to be properly configured.

Load-Balancer-as-a-Service (LBaaS v1)

The following meters are collected for LBaaS v1:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Juno release					
network.services.lb.pool	Gauge	pool	pool ID	Notification, Pollster	Existence of a LB pool
network.services.lb.vip	Gauge	vip	vip ID	Notification, Pollster	Existence of a LB VIP
network.services.lb.member	Gauge	member	member ID	Notification, Pollster	Existence of a LB member
network.services.lb.health_monitor	Gauge	health_monitor	monitor ID	Notification, Pollster	Existence of a LB health probe

Name	Type	Unit	Resource	Origin	Note
network.services.lb.total.connections	Cumulative	connection	pool ID	Pollster	Total connections on a LB
network.services.lb.active.connections	Gauge	connection	pool ID	Pollster	Active connections on a LB
network.services.lb.incoming.bytes	Gauge	B	pool ID	Pollster	Number of incoming Bytes
network.services.lb.outgoing.bytes	Gauge	B	pool ID	Pollster	Number of outgoing Bytes
Meters added in the Kilo release					
network.services.lb.pool.create	Delta	pool	pool ID	Notification	LB pool was created
network.services.lb.pool.update	Delta	pool	pool ID	Notification	LB pool was updated
network.services.lb.vip.create	Delta	vip	vip ID	Notification	LB VIP was created
network.services.lb.vip.update	Delta	vip	vip ID	Notification	LB VIP was updated
network.services.lb.member.create	Delta	member	member ID	Notification	LB member was created
network.services.lb.member.update	Delta	member	member ID	Notification	LB member was updated
network.services.lb.health_monitor.create	Delta	health_monitor	monitor ID	Notification	LB health probe was created
network.services.lb.health_monitor.update	Delta	health_monitor	monitor ID	Notification	LB health probe was updated

Load-Balancer-as-a-Service (LBaaS v2)

The following meters are collected for LBaaS v2. They are added in Mitaka release:

Name	Type	Unit	Resource	Origin	Note
network.services.lb.pool	Gauge	pool	pool ID	Notification, Pollster	Existence of a LB pool
network.services.lb.listener	Gauge	listener	listener ID	Notification, Pollster	Existence of a LB listener
network.services.lb.member	Gauge	member	member ID	Notification, Pollster	Existence of a LB member

Name	Type	Unit	Resource	Origin	Note
network.services.lb.health_monitor	Gauge	health_monitor	monitor ID	Notification, Pollster	Existence of a LB health probe
network.services.lb.loadbalancer	Gauge	loadbalancer	loadbalancer ID	Notification, Pollster	Existence of a LB loadbalancer
network.services.lb.total.connections	Cumulative	connection	pool ID	Pollster	Total connections on a LB
network.services.lb.active.connections	Gauge	connection	pool ID	Pollster	Active connections on a LB
network.services.lb.incoming.bytes	Gauge	B	pool ID	Pollster	Number of incoming Bytes
network.services.lb.outgoing.bytes	Gauge	B	pool ID	Pollster	Number of outgoing Bytes
network.services.lb.pool.create	Delta	pool	pool ID	Notification	LB pool was created
network.services.lb.pool.update	Delta	pool	pool ID	Notification	LB pool was updated
network.services.lb.listener.create	Delta	listener	listener ID	Notification	LB listener was created
network.services.lb.listener.update	Delta	listener	listener ID	Notification	LB listener was updated
network.services.lb.member.create	Delta	member	member ID	Notification	LB member was created
network.services.lb.member.update	Delta	member	member ID	Notification	LB member was updated
network.services.lb.healthmonitor.create	Delta	health_monitor	monitor ID	Notification	LB health probe was created
network.services.lb.healthmonitor.update	Delta	health_monitor	monitor ID	Notification	LB health probe was updated
network.services.lb.loadbalancer.create	Delta	loadbalancer	loadbalancer ID	Notification	LB loadbalancer was created
network.services.lb.loadbalancer.update	Delta	loadbalancer	loadbalancer ID	Notification	LB loadbalancer was updated

> **Note**
>
> The above meters are experimental and may generate a large load against the Neutron APIs. The future enhancement will be implemented when Neutron supports the new APIs.

VPN-as-a-Service (VPNaaS)

The following meters are collected for VPNaaS:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Juno release					
network.services.vpn	Gauge	vpnservice	vpn ID	Notification, Pollster	Existence of a VPN
network.services.vpn.connections	Gauge	ipsec_site_connection	connection ID	Notification, Pollster	Existence of an IPSec connection
Meters added in the Kilo release					
network.services.vpn.create	Delta	vpnservice	vpn ID	Notification	VPN was created
network.services.vpn.update	Delta	vpnservice	vpn ID	Notification	VPN was updated
network.services.vpn.connections.create	Delta	ipsec_site_connection	connection ID	Notification	IPSec connection was created
network.services.vpn.connections.update	Delta	ipsec_site_connection	connection ID	Notification	IPSec connection was updated
network.services.vpn.ipsecpolicy	Gauge	ipsecpolicy	ipsecpolicy ID	Notification, Pollster	Existence of an IPSec policy
network.services.vpn.ipsecpolicy.create	Delta	ipsecpolicy	ipsecpolicy ID	Notification	IPSec policy was created
network.services.vpn.ipsecpolicy.update	Delta	ipsecpolicy	ipsecpolicy ID	Notification	IPSec policy was updated
network.services.vpn.ikepolicy	Gauge	ikepolicy	ikepolicy ID	Notification, Pollster	Existence of an Ike policy
network.services.vpn.ikepolicy.create	Delta	ikepolicy	ikepolicy ID	Notification	Ike policy

Name	Type	Unit	Resource	Origin	Note
					was created
network.services.vpn.ikepolicy.update	Delta	ikepolicy	ikepolicy ID	Notification	Ike policy was updated

Firewall-as-a-Service (FWaaS)

The following meters are collected for FWaaS:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Juno release					
network.services.firewall	Gauge	firewall	firewall ID	Notification, Pollster	Existence of a firewall
network.services.firewall.policy	Gauge	firewall_policy	firewall ID	Notification, Pollster	Existence of a firewall policy
Meters added in the Kilo release					
network.services.firewall.create	Delta	firewall	firewall ID	Notification	Firewall was created
network.services.firewall.update	Delta	firewall	firewall ID	Notification	Firewall was updated
network.services.firewall.policy.create	Delta	firewall_policy	policy ID	Notification	Firewall policy was created
network.services.firewall.policy.update	Delta	firewall_policy	policy ID	Notification	Firewall policy was updated
network.services.firewall.rule	Gauge	firewall_rule	rule ID	Notification	Existence of a firewall rule
network.services.firewall.rule.create	Delta	firewall_rule	rule ID	Notification	Firewall rule was created
network.services.firewall.rule.update	Delta	firewall_rule	rule ID	Notification	Firewall rule was updated

Orchestration service

The following meters are collected for the Orchestration service:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Icehouse release or earlier					
stack.create	Delta	stack	stack ID	Notification	Stack was successfully created

Name	Type	Unit	Resource	Origin	Note
stack.update	Delta	stack	stack ID	Notification	Stack was successfully updated
stack.delete	Delta	stack	stack ID	Notification	Stack was successfully deleted
stack.resume	Delta	stack	stack ID	Notification	Stack was successfully resumed
stack.suspend	Delta	stack	stack ID	Notification	Stack was successfully suspended

Data processing service for OpenStack

The following meters are collected for the Data processing service for OpenStack:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Juno release					
cluster.create	Delta	cluster	cluster ID	Notification	Cluster was successfully created
cluster.update	Delta	cluster	cluster ID	Notification	Cluster was successfully updated
cluster.delete	Delta	cluster	cluster ID	Notification	Cluster was successfully deleted

Key Value Store module

The following meters are collected for the Key Value Store module:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Kilo release					
magnetodb.table.create	Gauge	table	table ID	Notification	Table was successfully created
magnetodb.table.delete	Gauge	table	table ID	Notification	Table was successfully deleted
magnetodb.table.index.count	Gauge	index	table ID	Notification	Number of indices created in a table

Energy

The following energy related meters are available:

Name	Type	Unit	Resource	Origin	Note
Meters added in the Icehouse release or earlier					
energy	Cumulative	kWh	probe ID	Pollster	Amount of energy
power	Gauge	W	probe ID	Pollster	Power consumption

Events

In addition to meters, the Telemetry service collects events triggered within an OpenStack environment. This section provides a brief summary of the events format in the Telemetry service.

While a sample represents a single, numeric datapoint within a time-series, an event is a broader concept that represents the state of a resource at a point in time. The state may be described using various data types including non-numeric data such as an instance's flavor. In general, events represent any action made in the OpenStack system.

Event configuration

To enable the creation and storage of events in the Telemetry service `store_events` option needs to be set to `True`. For further configuration options, see the event section in the OpenStack Configuration Reference.

> **Note**
>
> It is advisable to set `disable_non_metric_meters` to `True` when enabling events in the Telemetry service. The Telemetry service historically represented events as metering data, which may create duplication of data if both events and non-metric meters are enabled.

Event structure

Events captured by the Telemetry service are represented by five key attributes:

event_type
A dotted string defining what event occurred such as

`"compute.instance.resize.start"`.

message_id
A UUID for the event.

generated
A timestamp of when the event occurred in the system.

traits
A flat mapping of key-value pairs which describe the event. The event's traits contain most of the details of the event. Traits are typed, and can be strings, integers, floats, or datetimes.

raw
Mainly for auditing purpose, the full event message can be stored (unindexed) for future evaluation.

Event indexing

The general philosophy of notifications in OpenStack is to emit any and all data someone might need, and let the consumer filter out what they are not interested in. In order to make processing simpler and more efficient, the notifications are stored and processed within Ceilometer as events. The notification payload, which can be an arbitrarily complex JSON data structure, is converted to a flat set of key-value pairs. This conversion is specified by a config file.

> **Note**
>
> The event format is meant for efficient processing and querying. Storage of complete notifications for auditing purposes can be enabled by configuring `store_raw` option.

Event conversion

The conversion from notifications to events is driven by a configuration file defined by the `definitions_cfg_file` in the `ceilometer.conf` configuration file.

This includes descriptions of how to map fields in the notification body to Traits, and optional plug-ins for doing any programmatic translations (splitting a string, forcing case).

The mapping of notifications to events is defined per event_type, which can be wildcarded. Traits are added to events if the corresponding fields in the notification exist and are non-null.

> **Note**
>
> The default definition file included with the Telemetry service contains a list of known notifications and useful traits. The mappings provided can be modified to include more or less data according to user requirements.

If the definitions file is not present, a warning will be logged, but an empty set of definitions will be assumed. By default, any notifications that do not have a corresponding event definition in the definitions file will be converted to events with a set of minimal traits. This can be changed by setting the option `drop_unmatched_notifications` in the `ceilometer.conf` file. If this is set to `True`, any unmapped notifications will be dropped.

The basic set of traits (all are TEXT type) that will be added to all events if the notification has the relevant data are: service (notification's publisher), tenant_id, and request_id. These do not have to be specified in the event definition, they are automatically added, but their definitions can be overridden for a given event_type.

Event definitions format

The event definitions file is in YAML format. It consists of a list of event definitions, which are mappings. Order is significant, the list of definitions is scanned in reverse order to find a definition which matches the notification's event_type. That definition will be used to generate the event. The reverse ordering is done because it is common to want to have a more general wildcarded definition (such as `compute.instance.*`) with a set of traits common to all of those events, with a few more specific event definitions afterwards that have all of the above traits, plus a few more.

Each event definition is a mapping with two keys:

event_type
: This is a list (or a string, which will be taken as a 1 element list) of event_types this definition will handle. These can be wildcarded with unix shell glob syntax. An exclusion listing (starting with a `!`) will exclude any types listed from matching. If only exclusions are listed, the definition will match anything not matching the exclusions.

traits
: This is a mapping, the keys are the trait names, and the values are trait definitions.

Each trait definition is a mapping with the following keys:

fields
: A path specification for the field(s) in the notification you wish to extract for this trait. Specifications can be written to match multiple possible fields. By default the value will be the first such field. The paths can be specified with a dot syntax (`payload.host`). Square bracket syntax (`payload[host]`) is also supported. In either case, if the key for

the field you are looking for contains special characters, like `.`, it will need to be quoted (with double or single quotes): `payload.image_meta.'org.openstack__1__architecture'`. The syntax used for the field specification is a variant of JSONPath

type

(Optional) The data type for this trait. Valid options are: `text`, `int`, `float`, and `datetime`. Defaults to `text` if not specified.

plugin

(Optional) Used to execute simple programmatic conversions on the value in a notification field.

Troubleshoot Telemetry

Logging in Telemetry

The Telemetry service has similar log settings as the other OpenStack services. Multiple options are available to change the target of logging, the format of the log entries and the log levels.

The log settings can be changed in `ceilometer.conf`. The list of configuration options are listed in the logging configuration options table in the Telemetry section in the OpenStack Configuration Reference.

By default `stderr` is used as standard output for the log messages. It can be changed to either a log file or syslog. The `debug` and `verbose` options are also set to false in the default settings, the default log levels of the corresponding modules can be found in the table referred above.

Recommended order of starting services

As it can be seen in Bug 1355809, the wrong ordering of service startup can result in data loss.

When the services are started for the first time or in line with the message queue service restart, it takes time while the `ceilometer-collector` service establishes the connection and joins or rejoins to the configured exchanges. Therefore, if the `ceilometer-agent-compute`, `ceilometer-agent-central`, and the `ceilometer-agent-notification` services are started before the `ceilometer-collector` service, the `ceilometer-collector` service may lose some messages while connecting to the message queue service.

The possibility of this issue to happen is higher, when the polling interval is set to a

relatively short period. In order to avoid this situation, the recommended order of service startup is to start or restart the `ceilometer-collector` service after the message queue. All the other Telemetry services should be started or restarted after and the `ceilometer-agent-compute` should be the last in the sequence, as this component emits metering messages in order to send the samples to the collector.

Notification agent

In the Icehouse release of OpenStack a new service was introduced to be responsible for consuming notifications that are coming from other OpenStack services.

If the `ceilometer-agent-notification` service is not installed and started, samples originating from notifications will not be generated. In case of the lack of notification based samples, the state of this service and the log file of Telemetry should be checked first.

For the list of meters that are originated from notifications, see the Telemetry Measurements Reference.

Recommended `auth_url` to be used

When using the Telemetry command-line client, the credentials and the `os_auth_url` have to be set in order for the client to authenticate against OpenStack Identity. For further details about the credentials that have to be provided see the Telemetry Python API.

The service catalog provided by OpenStack Identity contains the URLs that are available for authentication. The URLs have different `ports`, based on whether the type of the given URL is `public`, `internal` or `admin`.

OpenStack Identity is about to change API version from v2 to v3. The `adminURL` endpoint (which is available via the port: `35357`) supports only the v3 version, while the other two supports both.

The Telemetry command line client is not adapted to the v3 version of the OpenStack Identity API. If the `adminURL` is used as `os_auth_url`, the **ceilometer** command results in the following error message:

```
$ ceilometer meter-list
  Unable to determine the Keystone version to authenticate with \
  using the given auth_url: http://10.0.2.15:35357/v2.0
```

Therefore when specifying the `os_auth_url` parameter on the command line or by using environment variable, use the `internalURL` or `publicURL`.

For more details check the bug report Bug 1351841.

Telemetry best practices

The following are some suggested best practices to follow when deploying and configuring the Telemetry service. The best practices are divided into data collection and storage.

Data collection

1. The Telemetry service collects a continuously growing set of data. Not all the data will be relevant for an administrator to monitor.
    - Based on your needs, you can edit the `pipeline.yaml` configuration file to include a selected number of meters while disregarding the rest.
    - By default, Telemetry service polls the service APIs every 10 minutes. You can change the polling interval on a per meter basis by editing the `pipeline.yaml` configuration file.

    > **Warning**
    >
    > If the polling interval is too short, it will likely cause increase of stored data and the stress on the service APIs.

    - Expand the configuration to have greater control over different meter intervals.

    > **Note**
    >
    > For more information, see the *Pipeline configuration*.

2. If you are using the Kilo version of Telemetry, you can delay or adjust polling requests by enabling the jitter support. This adds a random delay on how the polling agents send requests to the service APIs. To enable jitter, set `shuffle_time_before_polling_task` in the `ceilometer.conf` configuration file to an integer greater than 0.
3. If you are using Juno or later releases, based on the number of resources that will be polled, you can add additional central and compute agents as necessary. The agents are designed to scale horizontally.

> **Note**
>
> For more information see, *Support for HA deployment*.

4. If you are using Juno or later releases, use the `notifier://` publisher rather than `rpc://` as there is a certain level of overhead that comes with RPC.

> **Note**
>
> For more information on RPC overhead, see RPC overhead info.

Data storage

1. We recommend that you avoid open-ended queries. In order to get better performance you can use reasonable time ranges and/or other query constraints for retrieving measurements.

   For example, this open-ended query might return an unpredictable amount of data:

   ```
   $ ceilometer sample-list --meter cpu -q 'resource_id=INSTANCE_ID_1'
   ```

   Whereas, this well-formed query returns a more reasonable amount of data, hence better performance:

   ```
   $ ceilometer sample-list --meter cpu -q
   'resource_id=INSTANCE_ID_1;timestamp > 2015-05-01T00:00:00;timestamp <
   2015-06-01T00:00:00'
   ```

> **Note**
>
> As of the Liberty release, the number of items returned will be restricted to the value defined by `default_api_return_limit` in the `ceilometer.conf` configuration file. Alternatively, the value can be set per query by passing `limit` option in request.

2. You can install the API behind `mod_wsgi`, as it provides more settings to tweak, like `threads` and `processes` in case of `WSGIDaemon`.

> **Note**
>
> For more information on how to configure `mod_wsgi`, see the Telemetry Install Documentation.

3. The collection service provided by the Telemetry project is not intended to be an archival service. Set a Time to Live (TTL) value to expire data and minimize the database size. If you would like to keep your data for longer time period, you may consider storing it in a data warehouse outside of Telemetry.

> **Note**
>
> For more information on how to set the TTL, see *Storing samples*.

4. We recommend that you do not use SQLAlchemy back end prior to the Juno release, as it previously contained extraneous relationships to handle deprecated data models. This resulted in extremely poor query performance.
5. We recommend that you do not run MongoDB on the same node as the controller. Keep it on a separate node optimized for fast storage for better performance. Also it is advisable for the MongoDB node to have a lot of memory.

> **Note**
>
> For more information on how much memory you need, see MongoDB FAQ.

6. Use replica sets in MongoDB. Replica sets provide high availability through automatic failover. If your primary node fails, MongoDB will elect a secondary node to replace the primary node, and your cluster will remain functional.

   For more information on replica sets, see the MongoDB replica sets docs.
7. Use sharding in MongoDB. Sharding helps in storing data records across multiple machines and is the MongoDB's approach to meet the demands of data growth.

   For more information on sharding, see the MongoDB sharding docs.

Database

The Database service provides database management features.

Introduction

The Database service provides scalable and reliable cloud provisioning functionality for both relational and non-relational database engines. Users can quickly and easily use database features without the burden of handling complex administrative tasks. Cloud users and database administrators can provision and manage multiple database instances as needed.

The Database service provides resource isolation at high performance levels, and automates complex administrative tasks such as deployment, configuration, patching, backups, restores, and monitoring.

You can modify various cluster characteristics by editing the `/etc/trove/trove.conf` file. A comprehensive list of the Database service configuration options is described in the Database service chapter in the *Configuration Reference*.

Create a data store

An administrative user can create data stores for a variety of databases.

This section assumes you do not yet have a MySQL data store, and shows you how to create a MySQL data store and populate it with a MySQL 5.5 data store version.

To create a data store

1. **Create a trove image**

   Create an image for the type of database you want to use, for example, MySQL, MongoDB, Cassandra.

   This image must have the trove guest agent installed, and it must have the `trove-guestagent.conf` file configured to connect to your OpenStack environment. To configure `trove-guestagent.conf`, add the following lines to `trove-guestagent.conf` on the guest instance you are using to build your image:

   ```
   rabbit_host = controller
   rabbit_password = RABBIT_PASS
   nova_proxy_admin_user = admin
   nova_proxy_admin_pass = ADMIN_PASS
   nova_proxy_admin_tenant_name = service
   trove_auth_url = http://controller:35357/v2.0
   ```

   This example assumes you have created a MySQL 5.5 image called `mysql-5.5.qcow2`.

   > **Important**
   >
   > If you have a guest image that was created with an OpenStack version before Kilo, modify the guest agent init script for the guest image to read the configuration files from the directory `/etc/trove/conf.d`.
   >
   > For a backwards compatibility with pre-Kilo guest instances, set the database service configuration options `injected_config_location` to `/etc/trove` and `guest_info` to `/etc/guest_info`.

2. **Register image with Image service**

   You need to register your guest image with the Image service.

   In this example, you use the glance **image-create** command to register a `mysql-5.5.qcow2` image.

```
$ glance image-create --name mysql-5.5 --disk-format qcow2 --container-format bare --is-public True < mysql-5.5.qcow2
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | d41d8cd98f00b204e9800998ecf8427e     |
| container_format | bare                                 |
| created_at       | 2014-05-23T21:01:18                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | bb75f870-0c33-4907-8467-1367f8cb15b6 |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | mysql-5.5                            |
| owner            | 1448da1223124bb291f5ae8e9af4270d     |
| protected        | False                                |
| size             | 0                                    |
| status           | active                               |
| updated_at       | 2014-05-23T21:01:22                  |
| virtual_size     | None                                 |
+------------------+--------------------------------------+
```

3. **Create the data store**

   Create the data store that will house the new image. To do this, use the **trove-manage datastore_update** command.

   This example uses the following arguments:

Argument	Description	In this example:
config file	The configuration file to use.	*--config-file=/etc/trove/trove.conf*
name	Name you want to use for this data store.	`mysql`
default version	You can attach multiple versions/images to a data store. For example, you might have a MySQL 5.5 version and a MySQL 5.6 version. You can designate one version as the default, which the system uses if a user does not explicitly request a specific version.	`""` At this point, you do not yet have a default version, so pass in an empty string.

Example:

```
$ trove-manage --config-file=/etc/trove/trove.conf datastore_update mysql
""
```

4. **Add a version to the new data store**

   Now that you have a MySQL data store, you can add a version to it, using the **trove-manage datastore_version_update** command. The version indicates which guest image to use.

   This example uses the following arguments:

Argument	Description	In this example:
config file	The configuration file to use.	*--config-file=/etc/trove/trove.conf*
data store	The name of the data store you just created via `trove-manage` **datastore_update**.	`mysql`
version name	The name of the version you are adding to the data store.	`mysql-5.5`
data store manager	Which data store manager to use for this version. Typically, the data store manager is identified by one of the following strings, depending on the database: • cassandra • couchbase • couchdb • db2 • mariadb • mongodb • mysql • percona • postgresql • pxc • redis • vertica	`mysql`
glance ID	The ID of the guest image you just added to the Image service. You can get this ID by using the glance **image-show** IMAGE_NAME command.	bb75f870-0c33-4907-8467-1367f8cb15b6
packages	If you want to put additional	""

Argument	Description	In this example:
	packages on each guest that you create with this data store version, you can list the package names here.	In this example, the guest image already contains all the required packages, so leave this argument empty.
active	**Set this to either 1 or 0:** • `1` = active • `0` = disabled	1

Example:

```
$ trove-manage --config-file=/etc/trove/trove.conf
datastore_version_update mysql mysql-5.5 mysql GLANCE_ID "" 1
```

Optional. Set your new version as the default version. To do this, use the **trove-manage datastore_update** command again, this time specifying the version you just created.

```
$ trove-manage --config-file=/etc/trove/trove.conf datastore_update mysql
mysql-5.5
```

5. **Load validation rules for configuration groups**

> **Note**
>
> **Applies only to MySQL and Percona data stores**
>
> • If you just created a MySQL or Percona data store, then you need to load the appropriate validation rules, as described in this step.
>
> • If you just created a different data store, skip this step.

Background. You can manage database configuration tasks by using configuration groups. Configuration groups let you set configuration parameters, in bulk, on one or more databases.

When you set up a configuration group using the trove **configuration-create** command, this command compares the configuration values you are setting against a list of valid configuration values that are stored in the `validation-rules.json` file.

Operating System	Location of `validation-rules.json`	Notes
Ubuntu 14.04	`/usr/lib/python2.7/dist-packages/trove/templates/DATASTORE_NAME`	DATASTORE_NAME is the name of either the MySQL data store or the Percona data store. This is typically either `mysql` or `percona`.
RHEL 7, CentOS 7, Fedora 20, and Fedora 21	`/usr/lib/python2.7/site-packages/trove/templates/DATASTORE_NAME`	DATASTORE_NAME is the name of either the MySQL data store or the Percona data store. This is typically either `mysql` or `percona`.

Therefore, as part of creating a data store, you need to load the `validation-rules.json` file, using the **trove-manage db_load_datastore_config_parameters** command. This command takes the following arguments:

- Data store name
- Data store version
- Full path to the `validation-rules.json` file

This example loads the `validation-rules.json` file for a MySQL database on Ubuntu 14.04:

```
$ trove-manage db_load_datastore_config_parameters mysql mysql-5.5
/usr/lib/python2.7/dist-packages/trove/templates/mysql/validation-
rules.json
```

6. **Validate data store**

   To validate your new data store and version, start by listing the data stores on your system:

```
$ trove datastore-list
+--------------------------------------+--------------+
|                  id                  |     name     |
+--------------------------------------+--------------+
| 10000000-0000-0000-0000-000000000001 | Legacy MySQL |
| e5dc1da3-f080-4589-a4c2-eff7928f969a |    mysql     |
+--------------------------------------+--------------+
```

Take the ID of the MySQL data store and pass it in with the **datastore-version-list** command:

```
$ trove datastore-version-list DATASTORE_ID
+--------------------------------------+-----------+
|                  id                  |    name   |
+--------------------------------------+-----------+
| 36a6306b-efd8-4d83-9b75-8b30dd756381 | mysql-5.5 |
+--------------------------------------+-----------+
```

Data store classifications

The Database service supports a variety of both relational and non-relational database engines, but to a varying degree of support for each *data store*. The Database service project has defined several classifications that indicate the quality of support for each data store. Data stores also implement different extensions. An extension is called a *strategy* and is classified similar to data stores.

Valid classifications for a data store and a strategy are:

- Experimental
- Technical preview
- Stable

Each classification builds on the previous one. This means that a data store that meets the `technical preview` requirements must also meet all the requirements for `experimental`, and a data store that meets the `stable` requirements must also meet all the requirements for `technical preview`.

Requirements

- Experimental

  A data store is considered to be `experimental` if it meets these criteria:

  - It implements a basic subset of the Database service API including `create` and `delete`.
  - It has guest agent elements that allow guest agent creation.

- It has a definition of supported operating systems.
- It meets the other Documented Technical Requirements.

A strategy is considered `experimental` if:

- It meets the Documented Technical Requirements.

- Technical preview

  A data store is considered to be a `technical preview` if it meets the requirements of `experimental` and further:

  - It implements APIs required to plant and start the capabilities of the data store as defined in the Datastore Compatibility Matrix.

    > **Note**
    >
    > It is not required that the data store implements all features like resize, backup, replication, or clustering to meet this classification.

  - It provides a mechanism for building a guest image that allows you to exercise its capabilities.
  - It meets the other Documented Technical Requirements.

  > **Important**
  >
  > A strategy is not normally considered to be `technical preview`.

- Stable

  A data store or a strategy is considered `stable` if:

  - It meets the requirements of `technical preview`.
  - It meets the other Documented Technical Requirements.

Initial Classifications

The following table shows the current classification assignments for the different data stores.

Classification	Data store
Stable	MySQL
Technical Preview	Cassandra, MongoDB
Experimental	All others

Redis data store replication

Replication strategies are available for Redis with several commands located in the Redis data store manager:

- **create**
- **detach-replica**
- **eject-replica-source**
- **promote-to-replica-source**

Additional arguments for the **create** command include **–replica_of** and **–replica_count**.

Redis integration and unit tests

Unit tests and integration tests are also available for Redis.

1. Install redstack:

   ```
   $ ./redstack install

   .. note::

      Redstack is a development script used for integration
      testing and Database service development installations.
      Do not use Redstack in a production environment. For
      more information, see `the Database service
      developer docs
   <http://docs.openstack.org/developer/trove/dev/install.html#running-
   redstack-to-install-trove>`_
   ```

2. Start Redis:

   ```
   $ ./redstack kick-start redis
   ```

3. Run integration tests:

   ```
   $ ./redstack int-tests --group=replication
   ```

   You can run **–group=redis_supported** instead of **–group=replication** if needed.

Configure a cluster

An administrative user can configure various characteristics of a MongoDB cluster.

Query routers and config servers

Background. Each cluster includes at least one query router and one config server. Query routers and config servers count against your quota. When you delete a cluster, the system deletes the associated query router(s) and config server(s).

Configuration. By default, the system creates one query router and one config server per cluster. You can change this by editing the `/etc/trove/trove.conf` file. These settings are in the `mongodb` section of the file:

Setting	Valid values are:
num_config_servers_per_cluster	1 or 3
num_query_routers_per_cluster	1 or 3

Bare Metal

The Bare Metal service provides physical hardware management features.

Introduction

The Bare Metal service provides physical hardware as opposed to virtual machines. It also provides several reference drivers, which leverage common technologies like PXE and IPMI, to cover a wide range of hardware. The pluggable driver architecture also allows vendor-specific drivers to be added for improved performance or functionality not provided by reference drivers. The Bare Metal service makes physical servers as easy to provision as virtual machines in a cloud, which in turn will open up new avenues for enterprises and service providers.

System architecture

The Bare Metal service is composed of the following components:

1. An admin-only RESTful API service, by which privileged users, such as operators and other services within the cloud control plane, may interact with the managed bare-

metal servers.
2. A conductor service, which conducts all activity related to bare-metal deployments. Functionality is exposed via the API service. The Bare Metal service conductor and API service communicate via RPC.
3. Various drivers that support heterogeneous hardware, which enable features specific to unique hardware platforms and leverage divergent capabilities via a common API.
4. A message queue, which is a central hub for passing messages, such as RabbitMQ. It should use the same implementation as that of the Compute service.
5. A database for storing information about the resources. Among other things, this includes the state of the conductors, nodes (physical servers), and drivers.

When a user requests to boot an instance, the request is passed to the Compute service via the Compute service API and scheduler. The Compute service hands over this request to the Bare Metal service, where the request passes from the Bare Metal service API, to the conductor which will invoke a driver to successfully provision a physical server for the user.

Bare Metal deployment

1. PXE deploy process
2. Agent deploy process

Use Bare Metal

1. Install the Bare Metal service.
2. Setup the Bare Metal driver in the compute node's `nova.conf` file.
3. Setup TFTP folder and prepare PXE boot loader file.
4. Prepare the bare metal flavor.
5. Register the nodes with correct drivers.
6. Configure the driver information.
7. Register the ports information.
8. Use nova boot to kick off the bare metal provision.
9. Check nodes' provision state and power state.

Use multitenancy with Bare Metal service

Use multitenancy with Bare Metal service

Multitenancy allows creating a dedicated tenant network that extends the current Bare Metal (ironic) service capabilities of providing `flat` networks. Multitenancy works in conjunction with Networking (neutron) service to allow provisioning of a bare metal server onto the tenant network. Therefore, multiple tenants can get isolated instances after deployment.

Bare Metal service provides the `local_link_connection` information to the Networking service ML2 driver. The ML2 driver uses that information to plug the specified port to the tenant network.

`local_link_connection` fields

Field	Description
`switch_id`	Required. Identifies a switch and can be an LLDP-based MAC address or an OpenFlow-based `datapath_id`.
`port_id`	Required. Port ID on the switch, for example, Gig0/1.
`switch_info`	Optional. Used to distinguish different switch models or other vendor specific-identifier.

Configure Networking service ML2 driver

To enable the Networking service ML2 driver, edit the `/etc/neutron/plugins/ml2/ml2_conf.ini` file:

1. Add the name of your ML2 driver.
2. Add the vendor ML2 plugin configuration options.

```
[ml2]
...
mechanism_drivers = my_mechanism_driver

[my_vendor]
param_1 = ...
param_2 = ...
param_3 = ...
```

For more details, see Networking service mechanism drivers.

Configure Bare Metal service

After you configure the Networking service ML2 driver, configure Bare Metal service:

1. Edit the `/etc/ironic/ironic.conf` for the `ironic-conductor` service. Set the `network_interface` node field to a valid network driver that is used to switch, clean, and provision networks.

   ```
   [DEFAULT]
   ```

```
...
enabled_network_interfaces=flat,neutron

[neutron]
...
cleaning_network_uuid=$UUID
provisioning_network_uuid=$UUID
```

> **Warning**
>
> The `cleaning_network_uuid` and `provisioning_network_uuid` parameters are required for the `neutron` network interface. If they are not set, `ironic-conductor` fails to start.

2. Set `neutron` to use Networking service ML2 driver:

```
$ ironic node-create -n $NAME --network-interface neutron --driver
agent_ipmitool
```

3. Create a port with appropriate `local_link_connection` information. Set the `pxe_enabled` port attribute to `True` to create network ports for for the `pxe_enabled` ports only:

```
$ ironic --ironic-api-version latest port-create -a $HW_MAC_ADDRESS \
  -n $NODE_UUID -l switch_id=$SWITCH_MAC_ADDRESS \
  -l switch_info=$SWITCH_HOSTNAME -l port_id=$SWITCH_PORT --pxe-enabled true
```

Troubleshooting

No valid host found error

Problem

Sometimes `/var/log/nova/nova-conductor.log` contains the following error:

```
NoValidHost: No valid host was found. There are not enough hosts available.
```

The message `No valid host was found` means that the Compute service scheduler could not find a bare metal node suitable for booting the new instance.

This means there will be some mismatch between resources that the Compute service expects to find and resources that Bare Metal service advertised to the Compute service.

Solution

If you get this message, check the following:

1. Introspection should have succeeded for you before, or you should have entered the required bare-metal node properties manually. For each node in the **ironic node-list** command, use:

   ```
   $ ironic node-show <IRONIC-NODE-UUID>
   ```

   and make sure that `properties` JSON field has valid values for keys `cpus`, `cpu_arch`, `memory_mb` and `local_gb`.

2. The flavor in the Compute service that you are using does not exceed the bare-metal node properties above for a required number of nodes. Use:

   ```
   $ openstack flavor show FLAVOR
   ```

3. Make sure that enough nodes are in `available` state according to the **ironic node-list** command. Nodes in `manageable` state usually mean they have failed introspection.

4. Make sure nodes you are going to deploy to are not in maintenance mode. Use the **ironic node-list** command to check. A node automatically going to maintenance mode usually means the incorrect credentials for this node. Check them and then remove maintenance mode:

   ```
   $ ironic node-set-maintenance <IRONIC-NODE-UUID> off
   ```

5. It takes some time for nodes information to propagate from the Bare Metal service to the Compute service after introspection. Our tooling usually accounts for it, but if you did some steps manually there may be a period of time when nodes are not available to the Compute service yet. Check that the **nova hypervisor-stats** command correctly shows total amount of resources in your system.

Orchestration

Orchestration is an orchestration engine that provides the possibility to launch multiple composite cloud applications based on templates in the form of text files that can be treated like code. A native Heat Orchestration Template (HOT) format is evolving, but it also endeavors to provide compatibility with the AWS CloudFormation template format, so that many existing CloudFormation templates can be launched on OpenStack.

Introduction

The OpenStack Orchestration service, a tool for orchestrating clouds, automatically configures and deploys resources in stacks. The deployments can be simple, such as deploying WordPress on Ubuntu with an SQL back end, or complex, such as starting a server group that auto scales by starting and stopping using real-time CPU loading information from the Telemetry service.

Orchestration stacks are defined with templates, which are non-procedural documents. Templates describe tasks in terms of resources, parameters, inputs, constraints, and dependencies. When the Orchestration service was originally introduced, it worked with AWS CloudFormation templates, which are in the JSON format.

The Orchestration service also runs Heat Orchestration Template (HOT) templates that are written in YAML. YAML is a terse notation that loosely follows structural conventions (colons, returns, indentation) that are similar to Python or Ruby. Therefore, it is easier to write, parse, grep, generate with tools, and maintain source-code management systems.

Orchestration can be accessed through a CLI and RESTful queries. The Orchestration service provides both an OpenStack-native REST API and a CloudFormation-compatible Query API. The Orchestration service is also integrated with the OpenStack dashboard to perform stack functions through a web interface.

For more information about using the Orchestration service through the command line, see the OpenStack Command-Line Interface Reference.

Orchestration authorization model

The Orchestration authorization model defines the authorization process for requests during deferred operations. A common example is an auto-scaling group update. During the auto-scaling update operation, the Orchestration service requests resources of other components (such as servers from Compute or networks from Networking) to extend or reduce the capacity of an auto-scaling group.

The Orchestration service provides the following authorization models:

- Password authorization
- OpenStack Identity trusts authorization

Password authorization

The Orchestration service supports password authorization. Password authorization requires that a user pass a username and password to the Orchestration service. Encrypted password are stored in the database, and used for deferred operations.

Password authorization involves the following steps:

1. A user requests stack creation, by providing a token and username and password. The Dashboard or python-heatclient requests the token on the user's behalf.
2. If the stack contains any resources that require deferred operations, then the orchestration engine fails its validation checks if the user did not provide a valid username/password.
3. The username/password are encrypted and stored in the Orchestration database.
4. Orchestration creates a stack.
5. Later, the Orchestration service retrieves the credentials and requests another token on behalf of the user. The token is not limited in scope and provides access to all the roles of the stack owner.

OpenStack Identity trusts authorization

A trust is an OpenStack Identity extension that enables delegation, and optionally impersonation through the OpenStack Identity service. The key terminology is *trustor* (the user delegating) and *trustee* (the user being delegated to).

To create a trust, the *trustor* (in this case, the user creating the stack in the Orchestration service) provides the OpenStack Identity service with the following information:

- The ID of the *trustee* (who you want to delegate to, in this case, the Orchestration service user).
- The roles to be delegated. Configure roles through the `heat.conf` file. Ensure the configuration contains whatever roles are required to perform the deferred operations on the user's behalf. For example, launching an OpenStack Compute instance in response to an auto-scaling event.
- Whether to enable impersonation.

The OpenStack Identity service provides a *trust id*, which is consumed by *only* the trustee to obtain a *trust scoped token*. This token is limited in scope, such that the trustee has limited access to those roles delegated. In addition, the trustee has effective impersonation of the trustor user if it was selected when creating the trust. For more information, see the *Identity management* section.

Trusts authorization involves the following steps:

1. A user creates a stack through an API request (only the token is required).
2. The Orchestration service uses the token to create a trust between the stack owner (trustor) and the Orchestration service user (trustee). The service delegates a special role (or roles) as defined in the *trusts_delegated_roles* list in the Orchestration configuration file. By default, the Orchestration service sets all the roles from trustor available for trustee. Deployers might modify this list to reflect a local RBAC policy. For example, to ensure that the heat process can access only those services that are expected while impersonating a stack owner.
3. Orchestration stores the encrypted *trust id* in the Orchestration database.
4. When a deferred operation is required, the Orchestration service retrieves the *trust id* and requests a trust scoped token which enables the service user to impersonate the stack owner during the deferred operation. Impersonation is helpful, for example, so the service user can launch Compute instances on behalf of the stack owner in response to an auto-scaling event.

Authorization model configuration

Initially, the password authorization model was the default authorization model. Since the Kilo release, the Identity trusts authorization model is enabled for the Orchestration service by default.

To enable the password authorization model, change the following parameter in the `heat.conf` file:

```
deferred_auth_method=password
```

To enable the trusts authorization model, change the following parameter in the `heat.conf` file:

```
deferred_auth_method=trusts
```

To specify the trustor roles that it delegates to trustee during authorization, specify the `trusts_delegated_roles` parameter in the `heat.conf` file. If `trusts_delegated_roles` is not defined, then all the trustor roles are delegated to trustee.

> **Note**
>
> The trustor delegated roles must be pre-configured in the OpenStack Identity service before using them in the Orchestration service.

Stack domain users

Stack domain users allow the Orchestration service to authorize and start the following operations within booted virtual machines:

- Provide metadata to agents inside instances. Agents poll for changes and apply the configuration that is expressed in the metadata to the instance.
- Detect when an action is complete. Typically, software configuration on a virtual machine after it is booted. Compute moves the VM state to “Active” as soon as it creates it, not when the Orchestration service has fully configured it.
- Provide application level status or meters from inside the instance. For example, allow auto-scaling actions to be performed in response to some measure of performance or quality of service.

The Orchestration service provides APIs that enable all of these operations, but all of those APIs require authentication. For example, credentials to access the instance that the agent is running upon. The heat-cfntools agents use signed requests, which require an ec2 key pair created through Identity. The key pair is then used to sign requests to the Orchestration CloudFormation and CloudWatch compatible APIs, which are authenticated through signature validation. Signature validation uses the Identity ec2tokens extension.

Stack domain users encapsulate all stack-defined users (users who are created as a result of data that is contained in an Orchestration template) in a separate domain. The separate domain is created specifically to contain data related to the Orchestration stacks only. A user is created, which is the *domain admin*, and Orchestration uses the *domain admin* to manage the lifecycle of the users in the stack *user domain*.

Stack domain users configuration

To configure stack domain user, the Orchestration service completes the following tasks:

1. A special OpenStack Identity service domain is created. For example, a domain that is called `heat` and the ID is set with the `stack_user_domain` option in the `heat.conf` file.
2. A user with sufficient permissions to create and delete projects and users in the `heat` domain is created.
3. The username and password for the domain admin user is set in the `heat.conf` file (`stack_domain_admin` and `stack_domain_admin_password`). This user administers *stack domain users* on behalf of stack owners, so they no longer need to be administrators themselves. The risk of this escalation path is limited because the `heat_domain_admin` is only given administrative permission for the `heat` domain.

To set up stack domain users, complete the following steps:

1. Create the domain:

`$OS_TOKEN` refers to a token. For example, the service admin token or some other valid token for a user with sufficient roles to create users and domains. `$KS_ENDPOINT_V3` refers to the v3 OpenStack Identity endpoint (for example, `http://keystone_address:5000/v3` where *keystone_address* is the IP address or resolvable name for the Identity service).

```
$ openstack --os-token $OS_TOKEN --os-url=$KS_ENDPOINT_V3 --os-\
  identity-api-version=3 domain create heat --description "Owns \
  users and projects created by heat"
```

The domain ID is returned by this command, and is referred to as `$HEAT_DOMAIN_ID` below.

2. Create the user:

```
$ openstack --os-token $OS_TOKEN --os-url=$KS_ENDPOINT_V3 --os-\
  identity-api-version=3 user create --password $PASSWORD --domain \
  $HEAT_DOMAIN_ID heat_domain_admin --description "Manages users \
  and projects created by heat"
```

The user ID is returned by this command and is referred to as `$DOMAIN_ADMIN_ID` below.

3. Make the user a domain admin:

```
$ openstack --os-token $OS_TOKEN --os-url=$KS_ENDPOINT_V3 --os-\
  identity-api-version=3 role add --user $DOMAIN_ADMIN_ID --domain \
  $HEAT_DOMAIN_ID admin
```

Then you must add the domain ID, username and password from these steps to the `heat.conf` file:

```
stack_domain_admin_password = password
stack_domain_admin = heat_domain_admin
stack_user_domain = domain id returned from domain create above
```

Usage workflow

The following steps are run during stack creation:

1. Orchestration creates a new *stack domain project* in the `heat` domain if the stack contains any resources that require creation of a *stack domain user*.
2. For any resources that require a user, the Orchestration service creates the user in the *stack domain project*. The *stack domain project* is associated with the Orchestration stack in the Orchestration database, but is separate and unrelated (from an authentication perspective) to the stack owners project. The users who are created in the stack domain are still assigned the `heat_stack_user` role, so the API surface they can access is limited

through the `policy.json` file. For more information, see *OpenStack Identity documentation*.

3. When API requests are processed, the Orchestration service performs an internal lookup, and allows stack details for a given stack to be retrieved. Details are retrieved from the database for both the stack owner's project (the default API path to the stack) and the stack domain project, subject to the `policy.json` restrictions.

This means there are now two paths that can result in the same data being retrieved through the Orchestration API. The following example is for resource-metadata:

```
GET v1/{stack_owner_project_id}/stacks/{stack_name}/\
{stack_id}/resources/{resource_name}/metadata
```

or:

```
GET v1/{stack_domain_project_id}/stacks/{stack_name}/\
{stack_id}/resources/{resource_name}/metadata
```

The stack owner uses the former (via `openstack stack resource metadata STACK RESOURCE`), and any agents in the instance use the latter.

OpenStack command-line clients

Overview

Each OpenStack project provides a command-line client, which enables you to access the project API through easy-to-use commands. For example, the Compute service provides a nova command-line client.

You can run the commands from the command line, or include the commands within scripts to automate tasks. If you provide OpenStack credentials, such as your user name and password, you can run these commands on any computer.

Internally, each command uses cURL command-line tools, which embed API requests. OpenStack APIs are RESTful APIs, and use the HTTP protocol. They include methods, URIs, media types, and response codes.

OpenStack APIs are open-source Python clients, and can run on Linux or Mac OS X systems. On some client commands, you can specify a debug parameter to show the underlying API request for the command. This is a good way to become familiar with the OpenStack API calls.

As a cloud end user, you can use the OpenStack dashboard to provision your own resources within the limits set by administrators. You can modify the examples provided in this section to create other types and sizes of server instances.

The following table lists the command-line client for each OpenStack service with its package name and description.

OpenStack services and clients

Service	Client	Package	Description
Command-line client	**openstack**	**python-openstackclient**	**Common client for the OpenStack project.**
Application Catalog service	murano	python-muranoclient	Creates and manages applications.
Bare Metal service	ironic	python-ironicclient	manages and provisions physical machines.
Block Storage service	cinder	python-cinderclient	Creates and manages volumes.
Clustering service	senlin	python-senlinclient	Creates and

Service	Client	Package	Description
			manages clustering services.
Compute service	nova	python-novaclient	Creates and manages images, instances, and flavors.
Container Infrastructure Management service	magnum	python-magnumclient	Creates and manages containers.
Data Processing service	sahara	python-saharaclient	Creates and manages Hadoop clusters on OpenStack.
Database service	trove	python-troveclient	Creates and manages databases.
Deployment service	fuel	python-fuelclient	Plans deployments.
DNS service	designate	python-designateclient	Creates and manages self service authoritative DNS.
Identity service	keystone	python-keystoneclient	Creates and manages users, tenants, roles, endpoints, and credentials.
Image service	glance	python-glanceclient	Creates and manages images.
Key Manager service	barbican	python-barbicanclient	Creates and manages keys.
Monitoring	monasca	python-monascaclient	Monitoring solution.
Networking service	neutron	python-neutronclient	Configures networks for guest servers.
Object Storage service	swift	python-swiftclient	Gathers statistics, lists items, updates metadata, and

Service	Client	Package	Description
			uploads, downloads, and deletes files stored by the Object Storage service. Gains access to an Object Storage installation for ad hoc processing.
Orchestration service	heat	python-heatclient	Launches stacks from templates, views details of running stacks including events and resources, and updates and deletes stacks.
Rating service	cloudkitty	python-cloudkittyclient	Rating service.
Shared File Systems service	manila	python-manilaclient	Creates and manages shared file systems.
Telemetry service	ceilometer	python-ceilometerclient	Creates and collects measurements across OpenStack.
Telemetry v3	gnocchi	python-gnocchiclient	Creates and collects measurements across OpenStack.
Workflow service	mistral	python-mistralclient	Workflow service for OpenStack cloud.

Install the OpenStack command-line clients

Install the prerequisite software and the Python package for each OpenStack client.

Install the prerequisite software

Most Linux distributions include packaged versions of the command-line clients that you can install directly, see Installing_from_packages.

If you need to install the source package for the command-line package, the following table lists the software needed to run the command-line clients, and provides installation instructions as needed.

OpenStack command-line clients prerequisites

Prerequisite	Description
Python 2.7 or later	Currently, the clients do not support Python 3.
setuptools package	Installed by default on Mac OS X. Many Linux distributions provide packages to make setuptools easy to install. Search your package manager for setuptools to find an installation package. If you cannot find one, download the setuptools package directly from https://pypi.python.org/pypi/setuptools. The recommended way to install setuptools on Microsoft Windows is to follow the documentation provided on the setuptools website (https://pypi.python.org/pypi/setuptools). Another option is to use the unofficial binary installer maintained by Christoph Gohlke (http://www.lfd.uci.edu/~gohlke/pythonlibs/#setuptools).
pip package	To install the clients on a Linux, Mac OS X, or Microsoft Windows system, use pip. It is easy to use, ensures that you get the latest version of the clients from the Python Package Index, and lets you update or remove the packages later on. Since the installation process compiles source files, this requires the related Python development package for your operating system and distribution. Install pip through the package manager for your system: **MacOS** `# easy_install pip` **Microsoft Windows**

Prerequisite	Description

	Ensure that the `C:\Python27\Scripts` directory is defined in the `PATH` environment variable, and use the `easy_install` command from the setuptools package:
	
	`C:\>easy_install pip`
	
	Another option is to use the unofficial binary installer provided by Christoph Gohlke (http://www.lfd.uci.edu/~gohlke/pythonlibs/#pip).
	
	Ubuntu or Debian
	
	`# apt install python-dev python-pip`
	
	Note that extra dependencies may be required, per operating system, depending on the package being installed, such as is the case with Tempest.
	
	Red Hat Enterprise Linux, CentOS, or Fedora
	
	A packaged version enables you to use yum to install the package:
	
	`# yum install python-devel python-pip`
	
	There are also packaged versions of the clients available in RDO that enable yum to install the clients as described in Installing_from_packages.
	
	SUSE Linux Enterprise Server
	
	A packaged version available in the Open Build Service enables you to use YaST or zypper to install the package.
	
	First, add the Open Build Service repository:
	
	`# zypper addrepo -f obs://Cloud:OpenStack:Mitaka/SLE_12_SP1 Mitaka`
	
	Then install pip and use it to manage client installation:
	
	`# zypper install python-devel python-pip`
	
	There are also packaged versions of the clients available that enable zypper to install the clients as described in Installing_from_packages.
	
	openSUSE
	
	You can install pip and use it to manage client installation:
	
	`# zypper install python-devel python-pip`
	
	There are also packaged versions of the clients available that enable zypper to install the clients as described in Installing_from_packages.

Install the OpenStack client

The following example shows the command for installing the OpenStack client with `pip`, which supports multiple services.

```
# pip install python-openstackclient
```

The following individual clients are deprecated in favor of a common client. Instead of installing and learning all these clients, we recommend installing and using the OpenStack client. You may need to install an individual project's client because coverage is not yet sufficient in the OpenStack client. If you need to install an individual client's project, replace the `<project>` name in this `pip install` command using the list below.

```
# pip install python-<project>client
```

- `barbican` - Key Manager Service API
- `ceilometer` - Telemetry API
- `cinder` - Block Storage API and extensions
- `cloudkitty` - Rating service API
- `designate` - DNS service API
- `fuel` - Deployment service API
- `glance` - Image service API
- `gnocchi` - Telemetry API v3
- `heat` - Orchestration API
- `keystone` - Identity service API and extensions
- `magnum` - Container Infrastructure Management service API
- `manila` - Shared file systems API
- `mistral` - Workflow service API
- `monasca` - Monitoring API
- `murano` - Application catalog API
- `neutron` - Networking API
- `nova` - Compute API and extensions
- `sahara` - Data Processing API
- `senlin` - Clustering service API
- `swift` - Object Storage API
- `trove` - Database service API

While you can install the `keystone` client for interacting with version 2.0 of the service's API, you should use the `openstack` client for all Identity interactions. Identity API v2 is deprecated in the Mitaka release.

Installing with pip

Use pip to install the OpenStack clients on a Linux, Mac OS X, or Microsoft Windows system. It is easy to use and ensures that you get the latest version of the client from the Python Package Index. Also, pip enables you to update or remove a package.

Install each client separately by using the following command:

- For Mac OS X or Linux:

  ```
  # pip install python-PROJECTclient
  ```

- For Microsoft Windows:

  ```
  C:\>pip install python-PROJECTclient
  ```

Installing from packages

RDO, openSUSE, SUSE Linux Enterprise, Debian, and Ubuntu have client packages that can be installed without `pip`.

- On Red Hat Enterprise Linux, CentOS, or Fedora, use `yum` to install the clients from the packaged versions available in RDO:

  ```
  # yum install python-PROJECTclient
  ```

- For Ubuntu or Debian, use `apt-get` to install the clients from the packaged versions:

  ```
  # apt-get install python-PROJECTclient
  ```

- For openSUSE, use `zypper` to install the clients from the distribution packages service:

  ```
  # zypper install python-PROJECTclient
  ```

- For SUSE Linux Enterprise Server, use `zypper` to install the clients from the distribution packages in the Open Build Service. First, add the Open Build Service repository:

  ```
  # zypper addrepo -f obs://Cloud:OpenStack:Mitaka/SLE_12_SP1 Mitaka
  ```

  Then you can install the packages:

  ```
  # zypper install python-PROJECTclient
  ```

Upgrade or remove clients

To upgrade a client, add the *--upgrade* option to the **pip install** command:

```
# pip install --upgrade python-PROJECTclient
```

To remove the client, run the **pip uninstall** command:

```
# pip uninstall python-PROJECTclient
```

What's next

Before you can run client commands, you must create and source the `PROJECT-openrc.sh` file to set environment variables. See *Set environment variables using the OpenStack RC file*.

Discover the version number for a client

Run the following command to discover the version number for a client:

```
$ PROJECT --version
```

For example, to see the version number for the `openstack` client, run the following command:

```
$ openstack --version
2.2.0
```

Set environment variables using the OpenStack RC file

To set the required environment variables for the OpenStack command-line clients, you must create an environment file called an OpenStack rc file, or `openrc.sh` file. If your OpenStack installation provides it, you can download the file from the OpenStack dashboard as an administrative user or any other user. This project-specific environment file contains the credentials that all OpenStack services use.

When you source the file, environment variables are set for your current shell. The variables enable the OpenStack client commands to communicate with the OpenStack services that run in the cloud.

Note

Defining environment variables using an environment file is not a common practice on Microsoft Windows. Environment variables are usually defined in the *Advanced > System Properties* dialog box. One method for using these scripts as-is on Windows is to install Git for Windows and using Git Bash to source the environment variables and to run all CLI commands.

Download and source the OpenStack RC file

- Log in to the dashboard and from the drop-down list select the project for which you want to download the OpenStack RC file.
- On the *Project* tab, open the *Compute* tab and click *Access & Security*.
- On the *API Access* tab, click *Download OpenStack RC File* and save the file. The filename will be of the form `PROJECT-openrc.sh` where `PROJECT` is the name of the project for which you downloaded the file.
- Copy the `PROJECT-openrc.sh` file to the computer from which you want to run OpenStack commands.

  For example, copy the file to the computer from which you want to upload an image with a `glance` client command.
- On any shell from which you want to run OpenStack commands, source the `PROJECT-openrc.sh` file for the respective project.

  In the following example, the `demo-openrc.sh` file is sourced for the demo project:

  ```
  $ . demo-openrc.sh
  ```
- When you are prompted for an OpenStack password, enter the password for the user who downloaded the `PROJECT-openrc.sh` file.

Create and source the OpenStack RC file

Alternatively, you can create the `PROJECT-openrc.sh` file from scratch, if you cannot download the file from the dashboard.

- In a text editor, create a file named `PROJECT-openrc.sh` and add the following authentication information:

  ```
  export OS_USERNAME=username
  export OS_PASSWORD=password
  export OS_TENANT_NAME=projectName
  ```

```
export OS_AUTH_URL=https://identityHost:portNumber/v2.0
# The following lines can be omitted
export OS_TENANT_ID=tenantIDString
export OS_REGION_NAME=regionName
export OS_CACERT=/path/to/cacertFile
```

- On any shell from which you want to run OpenStack commands, source the `PROJECT-openrc.sh` file for the respective project. In this example, you source the `admin-openrc.sh` file for the admin project:

  ```
  $ . admin-openrc.sh
  ```

> **Note**
>
> You are not prompted for the password with this method. The password lives in clear text format in the `PROJECT-openrc.sh` file. Restrict the permissions on this file to avoid security problems. You can also remove the `OS_PASSWORD` variable from the file, and use the *--password* parameter with OpenStack client commands instead.
>
> **Note**
>
> You must set the `OS_CACERT` environment variable when using the https protocol in the `OS_AUTH_URL` environment setting because the verification process for the TLS (HTTPS) server certificate uses the one indicated in the environment. This certificate will be used when verifying the TLS (HTTPS) server certificate.

Override environment variable values

When you run OpenStack client commands, you can override some environment variable settings by using the options that are listed at the end of the `help` output of the various client commands. For example, you can override the `OS_PASSWORD` setting in the `PROJECT-openrc.sh` file by specifying a password on a **openstack** command, as follows:

```
$ openstack --os-password PASSWORD server list
```

Where `PASSWORD` is your password.

A user specifies their username and password credentials to interact with OpenStack, using any client command. These credentials can be specified using various mechanisms, namely, the environment variable or command-line argument. It is not safe to specify the password using either of these methods.

For example, when you specify your password using the command-line client with the *--os-password* argument, anyone with access to your computer can view it in plain text

with the ps field.

To avoid storing the password in plain text, you can prompt for the OpenStack password interactively.

Manage projects, users, and roles

As an administrator, you manage projects, users, and roles. Projects are organizational units in the cloud to which you can assign users. Projects are also known as *tenants* or *accounts*. Users can be members of one or more projects. Roles define which actions users can perform. You assign roles to user-project pairs.

You can define actions for OpenStack service roles in the `/etc/PROJECT/policy.json` files. For example, define actions for Compute service roles in the `/etc/nova/policy.json` file.

You can manage projects, users, and roles independently from each other.

During cloud set up, the operator defines at least one project, user, and role.

You can add, update, and delete projects and users, assign users to one or more projects, and change or remove the assignment. To enable or temporarily disable a project or user, update that project or user. You can also change quotas at the project level.

Before you can delete a user account, you must remove the user account from its primary project.

Before you can run client commands, you must download and source an OpenStack RC file. See Download and source the OpenStack RC file.

Projects

A project is a group of zero or more users. In Compute, a project owns virtual machines. In Object Storage, a project owns containers. Users can be associated with more than one project. Each project and user pairing can have a role associated with it.

List projects

List all projects with their ID, name, and whether they are enabled or disabled:

```
$ openstack project list
+----------------------------------+--------------------+
| id                               | name               |
+----------------------------------+--------------------+
| f7ac731cc11f40efbc03a9f9e1d1d21f | admin              |
| c150ab41f0d9443f8874e32e725a4cc8 | alt_demo           |
| a9debfe41a6d4d09a677da737b907d5e | demo               |
| 9208739195a34c628c58c95d157917d7 | invisible_to_admin |
```

```
| 3943a53dc92a49b2827fae94363851e1 | service           |
| 80cab5e1f02045abad92a2864cfd76cb | test_project      |
+----------------------------------+-------------------+
```

Create a project

Create a project named `new-project`:

```
$ openstack project create --description 'my new project' new-project
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | my new project                   |
| enabled     | True                             |
| id          | 1a4a0618b306462c9830f876b0bd6af2 |
| name        | new-project                      |
+-------------+----------------------------------+
```

Update a project

Specify the project ID to update a project. You can update the name, description, and enabled status of a project.

- To temporarily disable a project:

  ```
  $ openstack project set PROJECT_ID --disable
  ```

- To enable a disabled project:

  ```
  $ openstack project set PROJECT_ID --enable
  ```

- To update the name of a project:

  ```
  $ openstack project set PROJECT_ID --name project-new
  ```

- To verify your changes, show information for the updated project:

  ```
  $ openstack project show PROJECT_ID
  +-------------+----------------------------------+
  | Field       | Value                            |
  +-------------+----------------------------------+
  | description | my new project                   |
  | enabled     | True                             |
  | id          | 1a4a0618b306462c9830f876b0bd6af2 |
  | name        | project-new                      |
  +-------------+----------------------------------+
  ```

Delete a project

Specify the project ID to delete a project:

```
$ openstack project delete PROJECT_ID
```

Users

List users

List all users:

```
$ openstack user list
+----------------------------------+----------+
| id                               | name     |
+----------------------------------+----------+
| 352b37f5c89144d4ad0534139266d51f | admin    |
| 86c0de739bcb4802b8dc786921355813 | demo     |
| 32ec34aae8ea432e8af560a1cec0e881 | glance   |
| 7047fcb7908e420cb36e13bbd72c972c | nova     |
+----------------------------------+----------+
```

Create a user

To create a user, you must specify a name. Optionally, you can specify a tenant ID, password, and email address. It is recommended that you include the tenant ID and password because the user cannot log in to the dashboard without this information.

Create the `new-user` user:

```
$ openstack user create --project new-project --password PASSWORD new-user
+----------+----------------------------------+
| Field    | Value                            |
+----------+----------------------------------+
| email    |                                  |
| enabled  | True                             |
| id       | 6e5140962b424cb9814fb172889d3be2 |
| name     | new-user                         |
| tenantId | new-project                      |
+----------+----------------------------------+
```

Update a user

You can update the name, email address, and enabled status for a user.

- To temporarily disable a user account:

```
$ openstack user set USER_NAME --disable
```

If you disable a user account, the user cannot log in to the dashboard. However, data for the user account is maintained, so you can enable the user at any time.

- To enable a disabled user account:

```
$ openstack user set USER_NAME --enable
```

- To change the name and description for a user account:

```
$ openstack user set USER_NAME --name user-new --email new-
user@example.com
User has been updated.
```

Delete a user

Delete a specified user account:

```
$ openstack user delete USER_NAME
```

Roles and role assignments

List available roles

List the available roles:

```
$ openstack role list
+----------------------------------+---------------+
| id                               | name          |
+----------------------------------+---------------+
| 71ccc37d41c8491c975ae72676db687f | Member        |
| 149f50a1fe684bfa88dae76a48d26ef7 | ResellerAdmin |
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_      |
| 6ecf391421604da985db2f141e46a7c8 | admin         |
| deb4fffd123c4d02a907c2c74559dccf | anotherrole   |
+----------------------------------+---------------+
```

Create a role

Users can be members of multiple projects. To assign users to multiple projects, define a role and assign that role to a user-project pair.

Create the `new-role` role:

```
$ openstack role create new-role
+--------+----------------------------------+
| Field  | Value                            |
```

```
+--------+----------------------------------+
| id     | bef1f95537914b1295da6aa038ef4de6 |
| name   | new-role                         |
+--------+----------------------------------+
```

Assign a role

To assign a user to a project, you must assign the role to a user-project pair. To do this, you need the user, role, and project IDs.

- List users and note the user ID you want to assign to the role:

```
$ openstack user list
+----------------------------------+----------+---------+---------------------+
| id                               | name     | enabled | email               |
+----------------------------------+----------+---------+---------------------+
| 352b37f5c89144d4ad0534139266d51f | admin    | True    | admin@example.com   |
| 981422ec906d4842b2fc2a8658a5b534 | alt_demo | True    | alt_demo@example.com |
| 036e22a764ae497992f5fb8e9fd79896 | cinder   | True    | cinder@example.com  |
| 86c0de739bcb4802b8dc786921355813 | demo     | True    | demo@example.com    |
| 32ec34aae8ea432e8af560a1cec0e881 | glance   | True    | glance@example.com  |
| 7047fcb7908e420cb36e13bbd72c972c | nova     | True    | nova@example.com    |
+----------------------------------+----------+---------+---------------------+
```

- List role IDs and note the role ID you want to assign:

```
$ openstack role list
+----------------------------------+----------------+
| id                               | name           |
+----------------------------------+----------------+
| 71ccc37d41c8491c975ae72676db687f | Member         |
| 149f50a1fe684bfa88dae76a48d26ef7 | ResellerAdmin  |
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_       |
| 6ecf391421604da985db2f141e46a7c8 | admin          |
| deb4fffd123c4d02a907c2c74559dccf | anotherrole    |
| bef1f95537914b1295da6aa038ef4de6 | new-role       |
+----------------------------------+----------------+
```

- List projects and note the project ID you want to assign to the role:

```
$ openstack project list
+----------------------------------+---------------------+---------+
| id                               | name                | enabled |
+----------------------------------+---------------------+---------+
| f7ac731cc11f40efbc03a9f9e1d1d21f | admin               |   True  |
| c150ab41f0d9443f8874e32e725a4cc8 | alt_demo            |   True  |
| a9debfe41a6d4d09a677da737b907d5e | demo                |   True  |
| 9208739195a34c628c58c95d157917d7 | invisible_to_admin  |   True  |
| caa9b4ce7d5c4225aa25d6ff8b35c31f | new-user            |   True  |
```

```
| 1a4a0618b306462c9830f876b0bd6af2 | project-new        |   True  |
| 3943a53dc92a49b2827fae94363851e1 | service            |   True  |
| 80cab5e1f02045abad92a2864cfd76cb | test_project       |   True  |
+----------------------------------+--------------------+---------+
```

- Assign a role to a user-project pair:

  ```
  $ openstack role add --user USER_NAME --project TENANT_ID ROLE_NAME
  ```

  For example, assign the `new-role` role to the `demo` and `test-project` pair:

  ```
  $ openstack role add --user demo --project test-project new-role
  ```

- Verify the role assignment:

  ```
  $ openstack role list --user USER_NAME --project TENANT_ID
  +--------------+----------+--------------------------+--------------+
  | id           | name     | user_id                  | tenant_id    |
  +--------------+----------+--------------------------+--------------+
  | bef1f9553... | new-role | 86c0de739bcb4802b21355... | 80cab5e1f... |
  +--------------+----------+--------------------------+--------------+
  ```

View role details

View details for a specified role:

```
$ openstack role show ROLE_NAME
+----------+----------------------------------+
| Field    | Value                            |
+----------+----------------------------------+
| id       | bef1f95537914b1295da6aa038ef4de6 |
| name     | new-role                         |
+----------+----------------------------------+
```

Remove a role

Remove a role from a user-project pair:

- Run the **openstack role remove** command:

  ```
  $ openstack role remove --user USER_NAME --project TENANT_ID ROLE_NAME
  ```

- Verify the role removal:

  ```
  $ openstack role list --user USER_NAME --project TENANT_ID
  ```

  If the role was removed, the command output omits the removed role.

Manage project security

Security groups are sets of IP filter rules that are applied to all project instances, which define networking access to the instance. Group rules are project specific; project members can edit the default rules for their group and add new rule sets.

All projects have a `default` security group which is applied to any instance that has no other defined security group. Unless you change the default, this security group denies all incoming traffic and allows only outgoing traffic to your instance.

You can use the `allow_same_net_traffic` option in the `/etc/nova/nova.conf` file to globally control whether the rules apply to hosts which share a network.

If set to:

- `True` (default), hosts on the same subnet are not filtered and are allowed to pass all types of traffic between them. On a flat network, this allows all instances from all projects unfiltered communication. With VLAN networking, this allows access between instances within the same project. You can also simulate this setting by configuring the default security group to allow all traffic from the subnet.
- `False`, security groups are enforced for all connections.

Additionally, the number of maximum rules per security group is controlled by the `security_group_rules` and the number of allowed security groups per project is controlled by the `security_groups` quota (see *Manage quotas*).

List and view current security groups

From the command-line you can get a list of security groups for the project, using the **nova** command:

- Ensure your system variables are set for the user and tenant for which you are checking security group rules. For example:

```
export OS_USERNAME=demo00
export OS_TENANT_NAME=tenant01
```

- Output security groups, as follows:

```
$ nova secgroup-list
+---------+-------------+
| Name    | Description |
+---------+-------------+
| default | default     |
| open    | all ports   |
+---------+-------------+
```

- View the details of a group, as follows:

  ```
  $ nova secgroup-list-rules groupName
  ```

  For example:

  ```
  $ nova secgroup-list-rules open
  +-------------+-----------+---------+-----------+--------------+
  | IP Protocol | From Port | To Port | IP Range  | Source Group |
  +-------------+-----------+---------+-----------+--------------+
  | icmp        | -1        | 255     | 0.0.0.0/0 |              |
  | tcp         | 1         | 65535   | 0.0.0.0/0 |              |
  | udp         | 1         | 65535   | 0.0.0.0/0 |              |
  +-------------+-----------+---------+-----------+--------------+
  ```

  These rules are allow type rules as the default is deny. The first column is the IP protocol (one of icmp, tcp, or udp). The second and third columns specify the affected port range. The third column specifies the IP range in CIDR format. This example shows the full port range for all protocols allowed from all IPs.

Create a security group

When adding a new security group, you should pick a descriptive but brief name. This name shows up in brief descriptions of the instances that use it where the longer description field often does not. For example, seeing that an instance is using security group “http” is much easier to understand than “bobs_group” or “secgrp1”.

- Ensure your system variables are set for the user and tenant for which you are creating security group rules.
- Add the new security group, as follows:

  ```
  $ nova secgroup-create GroupName Description
  ```

  For example:

  ```
  $ nova secgroup-create global_http "Allows Web traffic anywhere on the Internet."
  +--------------------------------------+-------------+-------------------------------------------+
  | Id                                   | Name        | Description                               |
  +--------------------------------------+-------------+-------------------------------------------+
  | 1578a08c-5139-4f3e-9012-86bd9dd9f23b | global_http | Allows Web traffic anywhere on the Internet. |
  +--------------------------------------+-------------+-------------------------------------------+
  ```

- Add a new group rule, as follows:

  ```
  $ nova secgroup-add-rule secGroupName ip-protocol from-port to-port CIDR
  ```

  The arguments are positional, and the `from-port` and `to-port` arguments specify

the local port range connections are allowed to access, not the source and destination ports of the connection. For example:

```
$ nova secgroup-add-rule global_http tcp 80 80 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| tcp         | 80        | 80      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

You can create complex rule sets by creating additional rules. For example, if you want to pass both HTTP and HTTPS traffic, run:

```
$ nova secgroup-add-rule global_http tcp 443 443 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| tcp         | 443       | 443     | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

Despite only outputting the newly added rule, this operation is additive (both rules are created and enforced).

- View all rules for the new security group, as follows:

```
$ nova secgroup-list-rules global_http
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| tcp         | 80        | 80      | 0.0.0.0/0 |              |
| tcp         | 443       | 443     | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

Delete a security group

- Ensure your system variables are set for the user and tenant for which you are deleting a security group.
- Delete the new security group, as follows:

```
$ nova secgroup-delete GroupName
```

For example:

```
$ nova secgroup-delete global_http
```

Create security group rules for a cluster of instances

Source Groups are a special, dynamic way of defining the CIDR of allowed sources. The user specifies a Source Group (Security Group name), and all the user's other Instances using the specified Source Group are selected dynamically. This alleviates the need for individual rules to allow each new member of the cluster.

- Make sure to set the system variables for the user and tenant for which you are creating a security group rule.
- Add a source group, as follows:

  ```
  $ nova secgroup-add-group-rule secGroupName source-group ip-protocol
  from-port to-port
  ```

  For example:

  ```
  $ nova secgroup-add-group-rule cluster global_http tcp 22 22
  ```

  The `cluster` rule allows SSH access from any other instance that uses the `global_http` group.

Manage services

Create and manage services and service users

The Identity service enables you to define services, as follows:

- Service catalog template. The Identity service acts as a service catalog of endpoints for other OpenStack services. The `/etc/keystone/default_catalog.templates` template file defines the endpoints for services. When the Identity service uses a template file back end, any changes that are made to the endpoints are cached. These changes do not persist when you restart the service or reboot the machine.
- An SQL back end for the catalog service. When the Identity service is online, you must add the services to the catalog. When you deploy a system for production,

use the SQL back end.

The `auth_token` middleware supports the use of either a shared secret or users for each service.

To authenticate users against the Identity service, you must create a service user for each OpenStack service. For example, create a service user for the Compute, Block Storage, and Networking services.

To configure the OpenStack services with service users, create a project for all services and create users for each service. Assign the admin role to each service user and project pair. This role enables users to validate tokens and authenticate and authorize other user requests.

Create a service

- List the available services:

```
$ openstack service list
+----------------------------------+----------+------------+
| ID                               | Name     | Type       |
+----------------------------------+----------+------------+
| 9816f1faaa7c4842b90fb4821cd09223 | cinder   | volume     |
| 1250f64f31e34dcd9a93d35a075ddbe1 | cinderv2 | volumev2   |
| da8cf9f8546b4a428c43d5e032fe4afc | ec2      | ec2        |
| 5f105eeb55924b7290c8675ad7e294ae | glance   | image      |
| dcaa566e912e4c0e900dc86804e3dde0 | keystone | identity   |
| 4a715cfbc3664e9ebf388534ff2be76a | nova     | compute    |
| 1aed4a6cf7274297ba4026cf5d5e96c5 | novav21  | computev21 |
| bed063c790634c979778551f66c8ede9 | neutron  | network    |
| 6feb2e0b98874d88bee221974770e372 |    s3    |    s3      |
+----------------------------------+----------+------------+
```

- To create a service, run this command:

```
$ openstack service create --name SERVICE_NAME --description
SERVICE_DESCRIPTION SERVICE_TYPE
```

The arguments are:

- `service_name`: the unique name of the new service.
- `service_type`: the service type, such as `identity`, `compute`, `network`, `image`, `object-store` or any other service identifier string.
- `service_description`: the description of the service.

For example, to create a `swift` service of type `object-store`, run this command:

```
$ openstack service create --name swift --description "object store
service" object-store
```

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | object store service             |
| enabled     | True                             |
| id          | 84c23f4b942c44c38b9c42c5e517cd9a |
| name        | swift                            |
| type        | object-store                     |
+-------------+----------------------------------+
```

- To get details for a service, run this command:

  ```
  $ openstack service show SERVICE_TYPE|SERVICE_NAME|SERVICE_ID
  ```

  For example:

  ```
  $ openstack service show object-store
  +-------------+----------------------------------+
  | Field       | Value                            |
  +-------------+----------------------------------+
  | description | object store service             |
  | enabled     | True                             |
  | id          | 84c23f4b942c44c38b9c42c5e517cd9a |
  | name        | swift                            |
  | type        | object-store                     |
  +-------------+----------------------------------+
  ```

Create service users

- Create a project for the service users. Typically, this project is named `service`, but choose any name you like:

  ```
  $ openstack project create service
  +-------------+----------------------------------+
  | Field       | Value                            |
  +-------------+----------------------------------+
  | description | None                             |
  | enabled     | True                             |
  | id          | 3e9f3f5399624b2db548d7f871bd5322 |
  | name        | service                          |
  +-------------+----------------------------------+
  ```

- Create service users for the relevant services for your deployment.
- Assign the admin role to the user-project pair.

  ```
  $ openstack role add --project service --user SERVICE_USER_NAME admin
  ```

```
+-------+----------------------------------+
| Field | Value                            |
+-------+----------------------------------+
| id    | 233109e756c1465292f31e7662b429b1 |
| name  | admin                            |
+-------+----------------------------------+
```

Delete a service

To delete a specified service, specify its ID.

```
$ openstack service delete SERVICE_TYPE|SERVICE_NAME|SERVICE_ID
```

For example:

```
$ openstack service delete object-store
```

Manage Compute services

You can enable and disable Compute services. The following examples disable and enable the `nova-compute` service.

- List the Compute services:

```
$ nova service-list
+------------------+----------+----------+---------+-------+----------------------------+-----------------+
| Binary           | Host     | Zone     | Status  | State | Updated_at                 | Disabled Reason |
+------------------+----------+----------+---------+-------+----------------------------+-----------------+
| nova-conductor   | devstack | internal | enabled | up    | 2013-10-16T00:56:08.000000 | None            |
| nova-cert        | devstack | internal | enabled | up    | 2013-10-16T00:56:09.000000 | None            |
| nova-compute     | devstack | nova     | enabled | up    | 2013-10-16T00:56:07.000000 | None            |
| nova-network     | devstack | internal | enabled | up    | 2013-10-16T00:56:06.000000 | None            |
| nova-scheduler   | devstack | internal | enabled | up    | 2013-10-16T00:56:04.000000 | None            |
| nova-consoleauth | devstack | internal | enabled | up    | 2013-10-16T00:56:07.000000 | None            |
+------------------+----------+----------+---------+-------+----------------------------+-----------------+
```

- Disable a nova service:

```
$ nova service-disable localhost.localdomain nova-compute --reason 'trial
log'
+-----------+--------------+-----------+--------------------+
| Host      | Binary       | Status    | Disabled Reason    |
+-----------+--------------+-----------+--------------------+
| devstack | nova-compute | disabled | Trial log          |
+-----------+--------------+-----------+--------------------+
```

- Check the service list:

```
$ nova service-list
+------------------+----------+----------+---------+-------+----------------------------+-----------------+
| Binary           | Host     | Zone     | Status  | State | Updated_at                 | Disabled Reason |
+------------------+----------+----------+---------+-------+----------------------------+-----------------+
```

```
| nova-conductor   | devstack | internal | enabled  | up    | 2013-10-16T00:56:48.000000 | None            |
| nova-cert        | devstack | internal | enabled  | up    | 2013-10-16T00:56:49.000000 | None            |
| nova-compute     | devstack | nova     | disabled | up    | 2013-10-16T00:56:47.000000 | Trial log       |
| nova-network     | devstack | internal | enabled  | up    | 2013-10-16T00:56:51.000000 | None            |
| nova-scheduler   | devstack | internal | enabled  | up    | 2013-10-16T00:56:44.000000 | None            |
| nova-consoleauth | devstack | internal | enabled  | up    | 2013-10-16T00:56:47.000000 | None            |
+------------------+----------+----------+---------+-------+----------------------------+-----------------+
```

- Enable the service:

```
$ nova service-enable localhost.localdomain nova-compute
+----------+--------------+---------+
| Host     | Binary       | Status  |
+----------+--------------+---------+
| devstack | nova-compute | enabled |
+----------+--------------+---------+
```

- Check the service list:

```
$ nova service-list
+------------------+----------+----------+---------+-------+----------------------------+-----------------+
| Binary           | Host     | Zone     | Status  | State | Updated_at                 | Disabled Reason |
+------------------+----------+----------+---------+-------+----------------------------+-----------------+
| nova-conductor   | devstack | internal | enabled | up    | 2013-10-16T00:57:08.000000 | None            |
| nova-cert        | devstack | internal | enabled | up    | 2013-10-16T00:57:09.000000 | None            |
| nova-compute     | devstack | nova     | enabled | up    | 2013-10-16T00:57:07.000000 | None            |
| nova-network     | devstack | internal | enabled | up    | 2013-10-16T00:57:11.000000 | None            |
| nova-scheduler   | devstack | internal | enabled | up    | 2013-10-16T00:57:14.000000 | None            |
| nova-consoleauth | devstack | internal | enabled | up    | 2013-10-16T00:57:07.000000 | None            |
+------------------+----------+----------+---------+-------+----------------------------+-----------------+
```

Manage images

The cloud operator assigns roles to users. Roles determine who can upload and manage images. The operator might restrict image upload and management to only cloud administrators or operators.

You can upload images through the `glance` client or the Image service API. You can use the `nova` client for the image management. The latter provides mechanisms to list and delete images, set and delete image metadata, and create images of a running instance or snapshot and backup types.

After you upload an image, you cannot change it.

For details about image creation, see the Virtual Machine Image Guide.

List or get details for images (glance)

To get a list of images and to get further details about a single image, use **openstack image list** and **openstack image show** commands.

```
$ openstack image list
```

```
+----------+--------------------------------+-------------+------------------+----------+--------+
| ID       | Name                           | Disk Format | Container Format | Size     | Status |
+----------+--------------------------------+-------------+------------------+----------+--------+
| 397e7... | cirros-0.3.2-x86_64-uec        | ami         | ami              | 25165824 | active |
| df430... | cirros-0.3.2-x86_64-uec-kernel | aki         | aki              | 4955792  | active |
| 3cf85... | cirros-0.3.2-x86_64-uec-ramdisk| ari         | ari              | 3714968  | active |
| 7e514... | myCirrosImage                  | ami         | ami              | 14221312 | active |
+----------+--------------------------------+-------------+------------------+----------+--------+
```

```
$ openstack image show myCirrosImage
```

```
+---------------------------------------+--------------------------------------+
| Property                              | Value                                |
+---------------------------------------+--------------------------------------+
| Property 'base_image_ref'             | 397e713c-b95b-4186-ad46-6126863ea0a9 |
| Property 'image_location'             | snapshot                             |
| Property 'image_state'                | available                            |
| Property 'image_type'                 | snapshot                             |
| Property 'instance_type_ephemeral_gb' | 0                                    |
| Property 'instance_type_flavorid'     | 2                                    |
| Property 'instance_type_id'           | 5                                    |
| Property 'instance_type_memory_mb'    | 2048                                 |
| Property 'instance_type_name'         | m1.small                             |
| Property 'instance_type_root_gb'      | 20                                   |
| Property 'instance_type_rxtx_factor'  | 1                                    |
| Property 'instance_type_swap'         | 0                                    |
| Property 'instance_type_vcpu_weight'  | None                                 |
| Property 'instance_type_vcpus'        | 1                                    |
| Property 'instance_uuid'              | 84c6e57d-a6b1-44b6-81eb-fcb36afd31b5 |
| Property 'kernel_id'                  | df430cc2-3406-4061-b635-a51c16e488ac |
| Property 'owner_id'                   | 66265572db174a7aa66eba661f58eb9e     |
| Property 'ramdisk_id'                 | 3cf852bd-2332-48f4-9ae4-7d926d50945e |
| Property 'user_id'                    | 376744b5910b4b4da7d8e6cb483b06a8     |
| checksum                              | 8e4838effa1969ad591655d6485c7ba8     |
| container_format                      | ami                                  |
| created_at                            | 2013-07-22T19:45:58                  |
| deleted                               | False                                |
| disk_format                           | ami                                  |
| id                                    | 7e5142af-1253-4634-bcc6-89482c5f2e8a |
| is_public                             | False                                |
| min_disk                              | 0                                    |
| min_ram                               | 0                                    |
| name                                  | myCirrosImage                        |
| owner                                 | 66265572db174a7aa66eba661f58eb9e     |
| protected                             | False                                |
| size                                  | 14221312                             |
| status                                | active                               |
| updated_at                            | 2013-07-22T19:46:42                  |
+---------------------------------------+--------------------------------------+
```

When viewing a list of images, you can also use `grep` to filter the list, as follows:

```
$ openstack image list | grep 'cirros'
```

```
| 397e713c-b95b-4186-ad46-612... | cirros-0.3.2-x86_64-uec         | ami | ami | 25165824 | active |
| df430cc2-3406-4061-b635-a51... | cirros-0.3.2-x86_64-uec-kernel  | aki | aki | 4955792  | active |
| 3cf852bd-2332-48f4-9ae4-7d9... | cirros-0.3.2-x86_64-uec-ramdisk | ari | ari | 3714968  | active |
```

> **Note**
>
> To store location metadata for images, which enables direct file access for a client, update the `/etc/glance/glance-api.conf` file with the following statements:
>
> - `show_multiple_locations = True`
> - `filesystem_store_metadata_file = filePath`
>
> where filePath points to a JSON file that defines the mount point for OpenStack images on your system and a unique ID. For example:
>
> `[{ "id": "2d9bb53f-70ea-4066-a68b-67960eaae673", "mountpoint": "/var/lib/glance/images/" }]`

After you restart the Image service, you can use the following syntax to view the image's location information:

```
$ openstack --os-image-api-version 2 image show imageID
```

For example, using the image ID shown above, you would issue the command as follows:

```
$ openstack --os-image-api-version 2 image show 2d9bb53f-70ea-4066-a68b-
67960eaae673
```

Create or update an image (glance)

To create an image, use **openstack image create**:

```
$ openstack image create imageName
```

To update an image by name or ID, use **openstack image set**:

```
$ openstack image set imageName
```

The following list explains the optional arguments that you can use with the `create` and `set` commands to modify image properties. For more information, refer to Image service chapter in the OpenStack Command-Line Interface Reference.

`--name NAME`
The name of the image.

`--disk-format DISK_FORMAT`
The disk format of the image. Acceptable formats are ami, ari, aki, vhd, vmdk, raw, qcow2, vdi, and iso.

`--container-format CONTAINER_FORMAT`
The container format of the image. Acceptable formats are ami, ari, aki, bare, docker, and ovf.

--owner TENANT_ID --size SIZE
The tenant who should own the image. The size of image data, in bytes.

--min-disk DISK_GB
The minimum size of the disk needed to boot the image, in gigabytes.

--min-ram DISK_RAM
The minimum amount of RAM needed to boot the image, in megabytes.

--location IMAGE_URL
The URL where the data for this image resides. This option is only available in V1 API. When using it, you also need to set `--os-image-api-version`. For example, if the image data is stored in swift, you could specify `--os-image-api-version 1 --location swift://account:key@example.com/container/obj`.

--file FILE
Local file that contains the disk image to be uploaded during the update. Alternatively, you can pass images to the client through stdin.

--checksum CHECKSUM
Hash of image data to use for verification.

--copy-from IMAGE_URL
Similar to *--location* in usage, but indicates that the image server should immediately copy the data and store it in its configured image store.

--is-public [True|False]
Makes an image accessible for all the tenants (admin-only by default).

--is-protected [True|False]
Prevents an image from being deleted.

--property KEY=VALUE
Arbitrary property to associate with image. This option can be used multiple times.

--purge-props
Deletes all image properties that are not explicitly set in the update request. Otherwise, those properties not referenced are preserved.

--human-readable
Prints the image size in a human-friendly format.

The following example shows the command that you would use to upload a CentOS 6.3 image in qcow2 format and configure it for public access:

```
$ openstack image create --name centos63-image --disk-format qcow2 \
  --container-format bare --is-public True --file ./centos63.qcow2
```

The following example shows how to update an existing image with a properties that describe the disk bus, the CD-ROM bus, and the VIF model:

> **Note**
>
> When you use OpenStack with VMware vCenter Server, you need to specify the `vmware_disktype` and `vmware_adaptertype` properties with **glance image-create**. Also, we recommend that you set the `hypervisor_type="vmware"` property. For more information, see Images with VMware vSphere in the OpenStack Configuration Reference.

```
$ openstack image set \
    --property hw_disk_bus=scsi \
    --property hw_cdrom_bus=ide \
    --property hw_vif_model=e1000 \
    f16-x86_64-openstack-sda
```

Currently the libvirt virtualization tool determines the disk, CD-ROM, and VIF device models based on the configured hypervisor type (`libvirt_type` in `/etc/nova/nova.conf` file). For the sake of optimal performance, libvirt defaults to using virtio for both disk and VIF (NIC) models. The disadvantage of this approach is that it is not possible to run operating systems that lack virtio drivers, for example, BSD, Solaris, and older versions of Linux and Windows.

If you specify a disk or CD-ROM bus model that is not supported, see the Disk_and_CD-ROM_bus_model_values_table. If you specify a VIF model that is not supported, the instance fails to launch. See the VIF_model_values_table.

The valid model values depend on the `libvirt_type` setting, as shown in the following tables.

Disk and CD-ROM bus model values

libvirt_type setting	Supported model values
qemu or kvm	• ide • scsi • virtio
xen	• ide • xen

VIF model values

libvirt_type setting	Supported model values
qemu or kvm	• e1000 • ne2k_pci • pcnet • rtl8139

libvirt_type setting	Supported model values
	• virtio
xen	• e1000 • netfront • ne2k_pci • pcnet • rtl8139
vmware	• VirtualE1000 • VirtualPCNet32 • VirtualVmxnet

Note

By default, hardware properties are retrieved from the image properties. However, if this information is not available, the `libosinfo` database provides an alternative source for these values.

If the guest operating system is not in the database, or if the use of `libosinfo` is disabled, the default system values are used.

Users can set the operating system ID or a `short-id` in image properties. For example:

```
$ openstack image set --property short-id=fedora23 \ name-of-my-fedora-image
```

Alternatively, users can set `id` to a URL:

```
$ glance image set \
  --property id=http://fedoraproject.org/fedora/23 \
  ID-of-my-fedora-image
```

Create an image from ISO image

You can upload ISO images to the Image service (glance). You can subsequently boot an ISO image using Compute.

In the Image service, run the following command:

```
$ openstack image create ISO_IMAGE --file IMAGE.iso \
  --disk-format iso --container-format bare
```

Optionally, to confirm the upload in Image service, run:

```
$ openstack image list
```

Troubleshoot image creation

If you encounter problems in creating an image in the Image service or Compute, the following information may help you troubleshoot the creation process.

- Ensure that the version of qemu you are using is version 0.14 or later. Earlier versions of qemu result in an `unknown option -s` error message in the `nova-compute.log` file.
- Examine the `/var/log/nova/nova-api.log` and `/var/log/nova/nova-compute.log` log files for error messages.

Manage volumes

A volume is a detachable block storage device, similar to a USB hard drive. You can attach a volume to only one instance. To create and manage volumes, you use a combination of `nova` and `cinder` client commands.

Migrate a volume

As an administrator, you can migrate a volume with its data from one location to another in a manner that is transparent to users and workloads. You can migrate only detached volumes with no snapshots.

Possible use cases for data migration include:

- Bring down a physical storage device for maintenance without disrupting workloads.
- Modify the properties of a volume.
- Free up space in a thinly-provisioned back end.

Migrate a volume with the **cinder migrate** command, as shown in the following example:

```
$ cinder migrate volumeID destinationHost --force-host-copy True|False
```

In this example, *--force-host-copy True* forces the generic host-based migration mechanism and bypasses any driver optimizations.

> **Note**
>
> If the volume has snapshots, the specified host destination cannot accept the volume. If the user is not an administrator, the migration fails.

Create a volume

This example creates a `my-new-volume` volume based on an image.

- List images, and note the ID of the image that you want to use for your volume:

```
$ nova image-list
+--------------------+--------------------------------+--------+--------------------------+
| ID                 | Name                           | Status | Server                   |
+--------------------+--------------------------------+--------+--------------------------+
| 397e713c-b95b-4186... | cirros-0.3.2-x86_64-uec         | ACTIVE |                          |
| df430cc2-3406-4061... | cirros-0.3.2-x86_64-uec-kernel  | ACTIVE |                          |
| 3cf852bd-2332-48f4... | cirros-0.3.2-x86_64-uec-ramdisk | ACTIVE |                          |
| 7e5142af-1253-4634... | myCirrosImage                   | ACTIVE | 84c6e57d-a6b1-44b6-81... |
| 89bcd424-9d15-4723... | mysnapshot                      | ACTIVE | f51ebd07-c33d-4951-87... |
+--------------------+--------------------------------+--------+--------------------------+
```

- List the availability zones, and note the ID of the availability zone in which you want to create your volume:

```
$ cinder availability-zone-list
+------+-----------+
| Name |   Status  |
+------+-----------+
| nova | available |
+------+-----------+
```

- Create a volume with 8 gibibytes (GiB) of space, and specify the availability zone and image:

```
$ cinder create 8 --display-name my-new-volume \
  --image-id 397e713c-b95b-4186-ad46-6126863ea0a9 \
  --availability-zone nova
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2013-07-25T17:02:12.472269      |
| display_description |                 None                 |
|     display_name    |             my-new-volume            |
|          id         | 573e024d-5235-49ce-8332-be1576d323f8 |
|       image_id      | 397e713c-b95b-4186-ad46-6126863ea0a9 |
|       metadata      |                  {}                  |
|         size        |                  8                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |              creating                |
|     volume_type     |                 None                 |
+---------------------+--------------------------------------+
```

- To verify that your volume was created successfully, list the available volumes:

```
$ cinder list
+------------------+-----------+-----------------+------+-------------+----------+-------------+
|    ID            |   Status  |   Display Name  | Size | Volume Type | Bootable | Attached to |
+------------------+-----------+-----------------+------+-------------+----------+-------------+
| 573e024d-523... | available |  my-new-volume  |  8   |     None    |   true   |             |
| bd7cf584-45d... | available | my-bootable-vol |  8   |     None    |   true   |             |
+------------------+-----------+-----------------+------+-------------+----------+-------------+
```

  If your volume was created successfully, its status is `available`. If its status is `error`, you might have exceeded your quota.

Create a volume from specified volume type

Cinder supports these three ways to specify `volume type` during volume creation.

- volume_type
- cinder_img_volume_type (via glance image metadata)
- default_volume_type (via cinder.conf)

volume_type

User can specify volume type when creating a volume.

```
$ cinder create --name <volume name> --volume-type <volume type> <size>
```

cinder_img_volume_type

If glance image has `cinder_img_volume_type` property, Cinder uses this parameter to specify `volume type` when creating a volume.

Choose glance image which has “cinder_img_volume_type” property and create a volume from the image.

```
$ glance image-list
+--------------------------------------+---------------------------------+
| ID                                   | Name                            |
+--------------------------------------+---------------------------------+
| a8701119-ca8d-4957-846c-9f4d27f251fa | cirros-0.3.4-x86_64-uec         |
| 6cf01154-0408-416a-b69c-b28b48c5d28a | cirros-0.3.4-x86_64-uec-kernel  |
| de457c7c-2038-435d-abed-5dfa6430e66e | cirros-0.3.4-x86_64-uec-ramdisk |
+--------------------------------------+---------------------------------+
```

```
$ glance image-show a8701119-ca8d-4957-846c-9f4d27f251fa
+------------------------+--------------------------------------+
| Property               | Value                                |
+------------------------+--------------------------------------+
| checksum               | eb9139e4942121f22bbc2afc0400b2a4     |
| cinder_img_volume_type | lvmdriver-1                          |
| container_format       | ami                                  |
| created_at             | 2016-02-07T19:39:13Z                 |
| disk_format            | ami                                  |
| id                     | a8701119-ca8d-4957-846c-9f4d27f251fa |
| kernel_id              | 6cf01154-0408-416a-b69c-b28b48c5d28a |
| min_disk               | 0                                    |
| min_ram                | 0                                    |
| name                   | cirros-0.3.4-x86_64-uec              |
| owner                  | 4c0dbc92040c41b1bdb3827653682952     |
| protected              | False                                |
| ramdisk_id             | de457c7c-2038-435d-abed-5dfa6430e66e |
| size                   | 25165824                             |
| status                 | active                               |
| tags                   | []                                   |
| updated_at             | 2016-02-22T23:01:54Z                 |
| virtual_size           | None                                 |
| visibility             | public                               |
+------------------------+--------------------------------------+
```

```
$ cinder create --name test --image-id a8701119-ca8d-4957-846c-9f4d27f251fa 1
+---------------------------------------+--------------------------------------+
|                Property               |                Value                 |
+---------------------------------------+--------------------------------------+
|              attachments              |                  []                  |
|           availability_zone           |                 nova                 |
|                bootable               |                false                 |
|          consistencygroup_id          |                 None                 |
|               created_at              |      2016-02-22T23:17:51.000000      |
|              description              |                 None                 |
|               encrypted               |                False                 |
|                   id                  | 123ad92f-8f4c-4639-ab10-3742a1d9b47c |
|               metadata                |                  {}                  |
|            migration_status           |                 None                 |
|              multiattach              |                False                 |
|                  name                 |                 test                 |
|         os-vol-host-attr:host         |                 None                 |
|     os-vol-mig-status-attr:migstat    |                 None                 |
|     os-vol-mig-status-attr:name_id    |                 None                 |
|      os-vol-tenant-attr:tenant_id     |   4c0dbc92040c41b1bdb3827653682952   |
|   os-volume-replication:driver_data   |                 None                 |
| os-volume-replication:extended_status |                 None                 |
|           replication_status          |               disabled               |
|                  size                 |                  1                   |
|              snapshot_id              |                 None                 |
|              source_volid             |                 None                 |
|                status                 |               creating               |
|              updated_at               |                 None                 |
|               user_id                 |   9a125f3d111e47e6a25f573853b32fd9   |
|              volume_type              |             lvmdriver-1              |
+---------------------------------------+--------------------------------------+
```

default_volume_type

If above parameters are not set, Cinder uses default_volume_type which is defined in cinder.conf during volume creation.

Example cinder.conf file configuration.

```
[default]
default_volume_type = lvmdriver-1
```

Attach a volume to an instance

- Attach your volume to a server, specifying the server ID and the volume ID:

```
$ nova volume-attach 84c6e57d-a6b1-44b6-81eb-fcb36afd31b5 \
  573e024d-5235-49ce-8332-be1576d323f8 /dev/vdb
+----------+--------------------------------------+
| Property | Value                                |
+----------+--------------------------------------+
| device   | /dev/vdb                             |
| serverId | 84c6e57d-a6b1-44b6-81eb-fcb36afd31b5 |
| id       | 573e024d-5235-49ce-8332-be1576d323f8 |
| volumeId | 573e024d-5235-49ce-8332-be1576d323f8 |
+----------+--------------------------------------+
```

  Note the ID of your volume.

- Show information for your volume:

```
$ cinder show 573e024d-5235-49ce-8332-be1576d323f8
```

  The output shows that the volume is attached to the server with ID `84c6e57d-a6b1-44b6-81eb-fcb36afd31b5`, is in the nova availability zone, and is bootable.

```
+------------------------------+----------------------------------------+
|           Property           |                Value                   |
+------------------------------+----------------------------------------+
|         attachments          |         [{u'device': u'/dev/vdb',      |
|                              |        u'server_id': u'84c6e57d-a      |
|                              |           u'id': u'573e024d-...        |
|                              |        u'volume_id': u'573e024d...     |
|      availability_zone       |                 nova                   |
|           bootable           |                 true                   |
|          created_at          |       2013-07-25T17:02:12.000000       |
|     display_description      |                 None                   |
|         display_name         |             my-new-volume              |
|              id              |   573e024d-5235-49ce-8332-be1576d323f8 |
|           metadata           |                 {}                     |
|    os-vol-host-attr:host     |               devstack                 |
| os-vol-tenant-attr:tenant_id |     66265572db174a7aa66eba661f58eb9e   |
|             size             |                 8                      |
|         snapshot_id          |                 None                   |
|         source_volid         |                 None                   |
|            status            |               in-use                   |
|    volume_image_metadata     |       {u'kernel_id': u'df430cc2...,    |
```

```
|                              |        u'image_id': u'397e713c...,       |
|                              |        u'ramdisk_id': u'3cf852bd...,     |
|                              |u'image_name': u'cirros-0.3.2-x86_64-uec'}|
|         volume_type          |                  None                    |
+------------------------------+------------------------------------------+
```

Resize a volume

- To resize your volume, you must first detach it from the server. To detach the volume from your server, pass the server ID and volume ID to the following command:

  ```
  $ nova volume-detach 84c6e57d-a6b1-44b6-81eb-fcb36afd31b5   573e024d-5235-49ce-8332-be1576d323f8
  ```

  The **nova volume-detach** command does not return any output.

- List volumes:

  ```
  $ cinder list
  +----------------+-----------+-----------------+------+-------------+----------+-------------+
  |       ID       |   Status  |   Display Name  | Size | Volume Type | Bootable | Attached to |
  +----------------+-----------+-----------------+------+-------------+----------+-------------+
  | 573e024d-52... | available |  my-new-volume  |  8   |     None    |   true   |             |
  | bd7cf584-45... | available | my-bootable-vol |  8   |     None    |   true   |             |
  +----------------+-----------+-----------------+------+-------------+----------+-------------+
  ```

  Note that the volume is now available.

- Resize the volume by passing the volume ID and the new size (a value greater than the old one) as parameters:

  ```
  $ cinder extend 573e024d-5235-49ce-8332-be1576d323f8 10
  ```

  The **cinder extend** command does not return any output.

Delete a volume

- To delete your volume, you must first detach it from the server. To detach the volume from your server and check for the list of existing volumes, see steps 1 and 2 in Resize_a_volume.

  Delete the volume using either the volume name or ID:

  ```
  $ cinder delete my-new-volume
  ```

  The **cinder delete** command does not return any output.

- List the volumes again, and note that the status of your volume is `deleting`:

  ```
  $ cinder list
  +----------------+-----------+-----------------+------+-------------+----------+-------------+
  ```

```
|        ID       |   Status  |   Display Name  | Size | Volume Type | Bootable | Attached to |
+-----------------+-----------+-----------------+------+-------------+----------+-------------+
| 573e024d-523... |  deleting |  my-new-volume  |  8   |     None    |   true   |             |
| bd7cf584-45d... | available | my-bootable-vol |  8   |     None    |   true   |             |
+-----------------+-----------+-----------------+------+-------------+----------+-------------+
```

When the volume is fully deleted, it disappears from the list of volumes:

```
$ cinder list
+-----------------+-----------+-----------------+------+-------------+----------+-------------+
|       ID        |   Status  |   Display Name  | Size | Volume Type | Bootable | Attached to |
+-----------------+-----------+-----------------+------+-------------+----------+-------------+
| bd7cf584-45d... | available | my-bootable-vol |  8   |     None    |   true   |             |
+-----------------+-----------+-----------------+------+-------------+----------+-------------+
```

Transfer a volume

You can transfer a volume from one owner to another by using the **cinder transfer*** commands. The volume donor, or original owner, creates a transfer request and sends the created transfer ID and authorization key to the volume recipient. The volume recipient, or new owner, accepts the transfer by using the ID and key.

> **Note**
>
> The procedure for volume transfer is intended for tenants (both the volume donor and recipient) within the same cloud.

Use cases include:

- Create a custom bootable volume or a volume with a large data set and transfer it to a customer.
- For bulk import of data to the cloud, the data ingress system creates a new Block Storage volume, copies data from the physical device, and transfers device ownership to the end user.

Create a volume transfer request

- While logged in as the volume donor, list the available volumes:

```
$ cinder list
+-----------------+-----------+--------------+------+-------------+----------+-------------+
|        ID       |   Status  | Display Name | Size | Volume Type | Bootable | Attached to |
+-----------------+-----------+--------------+------+-------------+----------+-------------+
| 72bfce9f-cac... |   error   |     None     |  1   |     None    |  false   |             |
| a1cdace0-08e... | available |     None     |  1   |     None    |  false   |             |
+-----------------+-----------+--------------+------+-------------+----------+-------------+
```

- As the volume donor, request a volume transfer authorization code for a specific volume:

```
$ cinder transfer-create volumeID
```

The volume must be in an `available` state or the request will be denied. If the transfer request is valid in the database (that is, it has not expired or been deleted), the volume is placed in an `awaiting-transfer` state. For example:

```
$ cinder transfer-create a1cdace0-08e4-4dc7-b9dc-457e9bcfe25f
```

The output shows the volume transfer ID in the `id` row and the authorization key.

```
+------------+--------------------------------------+
|  Property  |                Value                 |
+------------+--------------------------------------+
|  auth_key  |           b2c8e585cbc68a80           |
| created_at |      2013-10-14T15:20:10.121458      |
|     id     | 6e4e9aa4-bed5-4f94-8f76-df43232f44dc |
|    name    |                 None                 |
| volume_id  | a1cdace0-08e4-4dc7-b9dc-457e9bcfe25f |
+------------+--------------------------------------+
```

> **Note**
>
> Optionally, you can specify a name for the transfer by using the `--display-name displayName` parameter.
>
> **Note**
>
> While the `auth_key` property is visible in the output of `cinder transfer-create VOLUME_ID`, it will not be available in subsequent `cinder transfer-show TRANSFER_ID` commands.

- Send the volume transfer ID and authorization key to the new owner (for example, by email).
- View pending transfers:

```
$ cinder transfer-list
```

```
+--------------------------------------+--------------------------------------+------+
|                  ID                  |               VolumeID               | Name |
+--------------------------------------+--------------------------------------+------+
| 6e4e9aa4-bed5-4f94-8f76-df43232f44dc | a1cdace0-08e4-4dc7-b9dc-457e9bcfe25f | None |
+--------------------------------------+--------------------------------------+------+
```

- After the volume recipient, or new owner, accepts the transfer, you can see that the transfer is no longer available:

```
$ cinder transfer-list
+----+-----------+------+
| ID | Volume ID | Name |
+----+-----------+------+
+----+-----------+------+
```

Accept a volume transfer request

- As the volume recipient, you must first obtain the transfer ID and authorization key from the original owner.
- Accept the request:

  ```
  $ cinder transfer-accept transferID authKey
  ```

  For example:

  ```
  $ cinder transfer-accept 6e4e9aa4-bed5-4f94-8f76-df43232f44dc
  b2c8e585cbc68a80
  +-----------+--------------------------------------+
  |  Property |                Value                 |
  +-----------+--------------------------------------+
  |     id    | 6e4e9aa4-bed5-4f94-8f76-df43232f44dc |
  |    name   |                 None                 |
  | volume_id | a1cdace0-08e4-4dc7-b9dc-457e9bcfe25f |
  +-----------+--------------------------------------+
  ```

> **Note**
>
> If you do not have a sufficient quota for the transfer, the transfer is refused.

Delete a volume transfer

- List available volumes and their statuses:

  ```
  $ cinder list
  +-------------+-----------------+--------------+------+-------------+----------+-------------+
  |     ID      |      Status     | Display Name | Size | Volume Type | Bootable | Attached to |
  +-------------+-----------------+--------------+------+-------------+----------+-------------+
  | 72bfce9f... |      error      |     None     |  1   |     None    |  false   |             |
  | a1cdace0... |awaiting-transfer|     None     |  1   |     None    |  false   |             |
  +-------------+-----------------+--------------+------+-------------+----------+-------------+
  ```

- Find the matching transfer ID:

```
$ cinder transfer-list
+--------------------------------------+--------------------------------------+------+
|                  ID                  |              VolumeID                | Name |
+--------------------------------------+--------------------------------------+------+
| a6da6888-7cdf-4291-9c08-8c1f22426b8a | a1cdace0-08e4-4dc7-b9dc-457e9bcfe25f | None |
+--------------------------------------+--------------------------------------+------+
```

- Delete the volume:

```
$ cinder transfer-delete transferID
```

  For example:

```
$ cinder transfer-delete a6da6888-7cdf-4291-9c08-8c1f22426b8a
```

- Verify that transfer list is now empty and that the volume is again available for transfer:

```
$ cinder transfer-list
+----+-----------+------+
| ID | Volume ID | Name |
+----+-----------+------+
+----+-----------+------+
```

```
$ cinder list
+-----------------+-----------+--------------+------+-------------+----------+-------------+
|        ID       |   Status  | Display Name | Size | Volume Type | Bootable | Attached to |
+-----------------+-----------+--------------+------+-------------+----------+-------------+
| 72bfce9f-ca...  |   error   |     None     |  1   |     None    |  false   |             |
| a1cdace0-08...  | available |     None     |  1   |     None    |  false   |             |
+-----------------+-----------+--------------+------+-------------+----------+-------------+
```

Manage and unmanage a snapshot

A snapshot is a point in time version of a volume. As an administrator, you can manage and unmanage snapshots.

Manage a snapshot

Manage a snapshot with the **cinder snapshot-manage** command:

```
$ cinder snapshot-manage VOLUME_ID IDENTIFIER --id-type ID-TYPE \
  --name NAME --description DESCRIPTION --metadata METADATA
```

The arguments to be passed are:

VOLUME_ID
The ID of a volume that is the parent of the snapshot, and managed by the Block

Storage service.

IDENTIFIER

Name, ID, or other identifier for an existing snapshot.

--id-type

Type of back-end device the identifier provided. Is typically `source-name` or `source-id`. Defaults to `source-name`.

--name

Name of the snapshot. Defaults to `None`.

--description

Description of the snapshot. Defaults to `None`.

--metadata

Metadata key-value pairs. Defaults to `None`.

The following example manages the `my-snapshot-id` snapshot with the `my-volume-id` parent volume:

```
$ cinder snapshot-manage my-volume-id my-snapshot-id
```

Unmanage a snapshot

Unmanage a snapshot with the **cinder snapshot-unmanage** command:

```
$ cinder snapshot-umanage SNAPSHOT
```

The arguments to be passed are:

SNAPSHOT

Name or ID of the snapshot to unmanage.

The following example unmanages the `my-snapshot-id` image:

```
$ cinder snapshot-unmanage my-snapshot-id
```

Manage shares

A share is provided by file storage. You can give access to a share to instances. To create and manage shares, use `manila` client commands.

Migrate a share

As an administrator, you can migrate a share with its data from one location to another in a manner that is transparent to users and workloads.

Possible use cases for data migration include:

- Bring down a physical storage device for maintenance without disrupting workloads.
- Modify the properties of a share.
- Free up space in a thinly-provisioned back end.

Migrate a share with the **manila migrate** command, as shown in the following example:

```
$ manila migrate shareID destinationHost --force-host-copy True|False
```

In this example, *--force-host-copy True* forces the generic host-based migration mechanism and bypasses any driver optimizations. `destinationHost` is in this format `host#pool` which includes destination host and pool.

> **Note**
>
> If the user is not an administrator, the migration fails.

Manage flavors

In OpenStack, flavors define the compute, memory, and storage capacity of nova computing instances. To put it simply, a flavor is an available hardware configuration for a server. It defines the `size` of a virtual server that can be launched.

> **Note**
>
> Flavors can also determine on which compute host a flavor can be used to launch an instance. For information about customizing flavors, refer to *Flavors*.

A flavor consists of the following parameters:

Flavor ID
Unique ID (integer or UUID) for the new flavor. If specifying 'auto', a UUID will be automatically generated.

Name
Name for the new flavor.

VCPUs
Number of virtual CPUs to use.

Memory MB
Amount of RAM to use (in megabytes).

Root Disk GB
Amount of disk space (in gigabytes) to use for the root (/) partition.

Ephemeral Disk GB

Amount of disk space (in gigabytes) to use for the ephemeral partition. If unspecified, the value is `0` by default. Ephemeral disks offer machine local disk storage linked to the lifecycle of a VM instance. When a VM is terminated, all data on the ephemeral disk is lost. Ephemeral disks are not included in any snapshots.

Swap

Amount of swap space (in megabytes) to use. If unspecified, the value is `0` by default.

RXTX Factor

Optional property that allows servers with a different bandwidth be created with the RXTX Factor. The default value is `1.0`. That is, the new bandwidth is the same as that of the attached network. The RXTX Factor is available only for Xen or NSX based systems.

Is Public

Boolean value defines whether the flavor is available to all users. Defaults to `True`.

Extra Specs

Key and value pairs that define on which compute nodes a flavor can run. These pairs must match corresponding pairs on the compute nodes. It can be used to implement special resources, such as flavors that run on only compute nodes with GPU hardware.

The default flavors are:

Flavor	VCPUs	Disk (in GB)	RAM (in MB)
m1.tiny	1	1	512
m1.small	1	20	2048
m1.medium	2	40	4096
m1.large	4	80	8192
m1.xlarge	8	160	16384

You can create and manage flavors with the **nova flavor-*** commands provided by the `python-novaclient` package.

Create a flavor

- List flavors to show the ID and name, the amount of memory, the amount of disk space for the root partition and for the ephemeral partition, the swap, and the number of virtual CPUs for each flavor:

  ```
  $ openstack flavor list
  ```

- To create a flavor, specify a name, ID, RAM size, disk size, and the number of VCPUs for the flavor, as follows:

  ```
  $ openstack flavor create FLAVOR_NAME FLAVOR_ID RAM_IN_MB ROOT_DISK_IN_GB
  NUMBER_OF_VCPUS
  ```

  > **Note**
  >
  > Unique ID (integer or UUID) for the new flavor. If specifying 'auto', a UUID will be automatically generated.

  Here is an example with additional optional parameters filled in that creates a public `extra tiny` flavor that automatically gets an ID assigned, with 256 MB memory, no disk space, and one VCPU. The rxtx-factor indicates the slice of bandwidth that the instances with this flavor can use (through the Virtual Interface (vif) creation in the hypervisor):

  ```
  $ openstack flavor create --is-public true m1.extra_tiny auto 256 0 1
  --rxtx-factor .1
  ```

- If an individual user or group of users needs a custom flavor that you do not want other tenants to have access to, you can change the flavor's access to make it a private flavor. See Private Flavors in the OpenStack Operations Guide.

  For a list of optional parameters, run this command:

  ```
  $ openstack help flavor create
  ```

- After you create a flavor, assign it to a project by specifying the flavor name or ID and the tenant ID:

  ```
  $ nova flavor-access-add FLAVOR TENANT_ID
  ```

- In addition, you can set or unset `extra_spec` for the existing flavor. The `extra_spec` metadata keys can influence the instance directly when it is launched. If a flavor sets the `extra_spec key/value quota:vif_outbound_peak=65536`, the instance's outbound peak bandwidth I/O should be LTE 512 Mbps. There are

several aspects that can work for an instance including `CPU limits`, `Disk tuning`, `Bandwidth I/O`, `Watchdog behavior`, and `Random-number generator`. For information about supporting metadata keys, see Flavors.

For a list of optional parameters, run this command:

```
$ nova help flavor-key
```

Delete a flavor

Delete a specified flavor, as follows:

```
$ openstack flavor delete FLAVOR_ID
```

Manage the OpenStack environment

This section includes tasks specific to the OpenStack environment.

- Select hosts where instances are launched
- Consider NUMA topology when booting instances
- Evacuate instances
- Migrate a single instance to another compute host
- Configure SSH between compute nodes
- Manage IP addresses
    - List addresses for all projects
    - Bulk create floating IP addresses
    - Bulk delete floating IP addresses
- Launch and manage stacks using the CLI
- Show usage statistics for hosts and instances
    - Show host usage statistics
    - Show instance usage statistics

Select hosts where instances are launched

With the appropriate permissions, you can select which host instances are launched on and which roles can boot instances on this host.

- To select the host where instances are launched, use the *--availability-zone ZONE:HOST* parameter on the **openstack server create** command.

  For example:

  ```
  $ openstack server create --image IMAGE --flavor m1.tiny \
  ```

```
  --key-name KEY --availability-zone ZONE:HOST \
  --nic net-id=UUID SERVER
```

- To specify which roles can launch an instance on a specified host, enable the `create:forced_host` option in the `policy.json` file. By default, this option is enabled for only the admin role.
- To view the list of valid compute hosts, use the **openstack hypervisor list** command.

```
$ openstack hypervisor list
+----+---------------------+
| ID | Hypervisor Hostname |
+----+---------------------+
|  1 | server2             |
|  2 | server3             |
|  3 | server4             |
+----+---------------------+
```

Consider NUMA topology when booting instances

NUMA topology can exist on both the physical hardware of the host, and the virtual hardware of the instance. OpenStack Compute uses libvirt to tune instances to take advantage of NUMA topologies. The libvirt driver boot process looks at the NUMA topology field of both the instance and the host it is being booted on, and uses that information to generate an appropriate configuration.

If the host is NUMA capable, but the instance has not requested a NUMA topology, Compute attempts to pack the instance into a single cell. If this fails, though, Compute will not continue to try.

If the host is NUMA capable, and the instance has requested a specific NUMA topology, Compute will try to pin the vCPUs of different NUMA cells on the instance to the corresponding NUMA cells on the host. It will also expose the NUMA topology of the instance to the guest OS.

If you want Compute to pin a particular vCPU as part of this process, set the `vcpu_pin_set` parameter in the `nova.conf` configuration file. For more information about the `vcpu_pin_set` parameter, see the Configuration Reference Guide.

Evacuate instances

If a hardware malfunction or other error causes a cloud compute node to fail, you can evacuate instances to make them available again. You can optionally include the target host on the **nova evacuate** command. If you omit the host, the scheduler chooses the

target host.

To preserve user data on the server disk, configure shared storage on the target host. When you evacuate the instance, Compute detects whether shared storage is available on the target host. Also, you must validate that the current VM host is not operational. Otherwise, the evacuation fails.

- To find a host for the evacuated instance, list all hosts:

  ```
  $ nova host-list
  ```

- Evacuate the instance. You can use the *--password PWD* option to pass the instance password to the command. If you do not specify a password, the command generates and prints one after it finishes successfully. The following command evacuates a server from a failed host to HOST_B.

  ```
  $ nova evacuate EVACUATED_SERVER_NAME HOST_B
  ```

  The command rebuilds the instance from the original image or volume and returns a password. The command preserves the original configuration, which includes the instance ID, name, uid, IP address, and so on.

  ```
  +-----------+--------------+
  | Property  |    Value     |
  +-----------+--------------+
  | adminPass | kRAJpErnT4xZ |
  +-----------+--------------+
  ```

- To preserve the user disk data on the evacuated server, deploy Compute with a shared file system. To configure your system, see *Configure migrations*. The following example does not change the password.

  ```
  $ nova evacuate EVACUATED_SERVER_NAME HOST_B --on-shared-storage
  ```

Migrate a single instance to another compute host

When you want to move an instance from one compute host to another, you can use the **nova migrate** command. The scheduler chooses the destination compute host based on its settings. This process does not assume that the instance has shared storage available on the target host. If you are using SSH tunneling, you must ensure that each node is configured with SSH key authentication so that the Compute service can use SSH to move disks to other nodes. For more information, see *Configure SSH between compute nodes*.

- To list the VMs you want to migrate, run:

  ```
  $ nova list
  ```

- After selecting a VM from the list, run this command where *VM_ID* is set to the ID in the list returned in the previous step:

```
$ nova show VM_ID
```

- Use the **nova migrate** command.

```
$ nova migrate VM_ID
```

- To migrate an instance and watch the status, use this example script:

```
#!/bin/bash

# Provide usage
usage() {
echo "Usage: $0 VM_ID"
exit 1
}

[[ $# -eq 0 ]] && usage

# Migrate the VM to an alternate hypervisor
echo -n "Migrating instance to alternate host"
VM_ID=$1
nova migrate $VM_ID
VM_OUTPUT=`nova show $VM_ID`
VM_STATUS=`echo "$VM_OUTPUT" | grep status | awk '{print $4}'`
while [[ "$VM_STATUS" != "VERIFY_RESIZE" ]]; do
echo -n "."
sleep 2
VM_OUTPUT=`nova show $VM_ID`
VM_STATUS=`echo "$VM_OUTPUT" | grep status | awk '{print $4}'`
done
nova resize-confirm $VM_ID
echo " instance migrated and resized."
echo;

# Show the details for the VM
echo "Updated instance details:"
nova show $VM_ID

# Pause to allow users to examine VM details
read -p "Pausing, press <enter> to exit."
```

> **Note**
>
> If you see this error, it means you are either trying the command with the wrong credentials, such as a non-admin user, or the `policy.json` file prevents migration for your user:
>
> ```
> ERROR (Forbidden): Policy doesn't allow
> compute_extension:admin_actions:migrate to be performed. (HTTP 403)
> ```
>
> **Note**
>
> If you see an error similar to this message, SSH tunneling was not set up between the compute nodes:
>
> ```
> ProcessExecutionError: Unexpected error while running command.
>
> Stderr: u Host key verification failed.\r\n
> ```

The instance is booted from a new host, but preserves its configuration including its ID, name, any metadata, IP address, and other properties.

Configure SSH between compute nodes

If you are resizing or migrating an instance between hypervisors, you might encounter an SSH (Permission denied) error. Ensure that each node is configured with SSH key authentication so that the Compute service can use SSH to move disks to other nodes.

To share a key pair between compute nodes, complete the following steps:

- On the first node, obtain a key pair (public key and private key). Use the root key that is in the `/root/.ssh/id_rsa` and `/root/.ssh/id_ras.pub` directories or generate a new key pair.
- Run **setenforce 0** to put SELinux into permissive mode.
- Enable login abilities for the nova user:

  ```
  # usermod -s /bin/bash nova
  ```

  Switch to the nova account.

  ```
  # su nova
  ```

- As root, create the folder that is needed by SSH and place the private key that you obtained in step 1 into this folder:

```
mkdir -p /var/lib/nova/.ssh
cp <private key>  /var/lib/nova/.ssh/id_rsa
echo 'StrictHostKeyChecking no' >> /var/lib/nova/.ssh/config
chmod 600 /var/lib/nova/.ssh/id_rsa /var/lib/nova/.ssh/authorized_keys
```

- Repeat steps 2-4 on each node.

> **Note**
>
> The nodes must share the same key pair, so do not generate a new key pair for any subsequent nodes.

- From the first node, where you created the SSH key, run:

```
ssh-copy-id -i <pub key> nova@remote-host
```

  This command installs your public key in a remote machine's `authorized_keys` folder.

- Ensure that the nova user can now log in to each node without using a password:

```
# su nova
$ ssh *computeNodeAddress*
$ exit
```

- As root on each node, restart both libvirt and the Compute services:

```
# systemctl restart libvirtd.service
# systemctl restart openstack-nova-compute.service
```

Manage IP addresses

Each instance has a private, fixed IP address that is assigned when the instance is launched. In addition, an instance can have a public or floating IP address. Private IP addresses are used for communication between instances, and public IP addresses are used for communication with networks outside the cloud, including the Internet.

- By default, both administrative and end users can associate floating IP addresses with projects and instances. You can change user permissions for managing IP addresses by updating the `/etc/nova/policy.json` file. For basic floating-IP procedures, refer to the Allocate a floating address to an instance section in the OpenStack End User Guide.
- For details on creating public networks using OpenStack Networking (`neutron`), refer to *Advanced features through API extensions*. No floating IP addresses are created by default in OpenStack Networking.

As an administrator using legacy networking (`nova-network`), you can use the following bulk commands to list, create, and delete ranges of floating IP addresses. These addresses can then be associated with instances by end users.

List addresses for all projects

To list all floating IP addresses for all projects, run:

```
$ nova floating-ip-bulk-list
+------------+---------------+---------------+--------+-----------+
| project_id | address       | instance_uuid | pool   | interface |
+------------+---------------+---------------+--------+-----------+
| None       | 172.24.4.225  | None          | public | eth0      |
| None       | 172.24.4.226  | None          | public | eth0      |
| None       | 172.24.4.227  | None          | public | eth0      |
| None       | 172.24.4.228  | None          | public | eth0      |
| None       | 172.24.4.229  | None          | public | eth0      |
| None       | 172.24.4.230  | None          | public | eth0      |
| None       | 172.24.4.231  | None          | public | eth0      |
| None       | 172.24.4.232  | None          | public | eth0      |
| None       | 172.24.4.233  | None          | public | eth0      |
| None       | 172.24.4.234  | None          | public | eth0      |
| None       | 172.24.4.235  | None          | public | eth0      |
| None       | 172.24.4.236  | None          | public | eth0      |
| None       | 172.24.4.237  | None          | public | eth0      |
| None       | 172.24.4.238  | None          | public | eth0      |
| None       | 192.168.253.1 | None          | test   | eth0      |
| None       | 192.168.253.2 | None          | test   | eth0      |
| None       | 192.168.253.3 | None          | test   | eth0      |
| None       | 192.168.253.4 | None          | test   | eth0      |
| None       | 192.168.253.5 | None          | test   | eth0      |
| None       | 192.168.253.6 | None          | test   | eth0      |
+------------+---------------+---------------+--------+-----------+
```

Bulk create floating IP addresses

To create a range of floating IP addresses, run:

```
$ nova floating-ip-bulk-create [--pool POOL_NAME] [--interface INTERFACE]
RANGE_TO_CREATE
```

For example:

```
$ nova floating-ip-bulk-create --pool test 192.168.1.56/29
```

By default, `floating-ip-bulk-create` uses the `public` pool and `eth0` interface values.

> **Note**
>
> You should use a range of free IP addresses that is valid for your network. If you are not sure, at least try to avoid the DHCP address range:
>
> - Pick a small range (/29 gives an 8 address range, 6 of which will be usable).
>
> - Use **nmap** to check a range's availability. For example, 192.168.1.56/29 represents a small range of addresses (192.168.1.56-63, with 57-62 usable), and you could run the command **nmap -sn 192.168.1.56/29** to check whether the entire range is currently unused.

Bulk delete floating IP addresses

To delete a range of floating IP addresses, run:

```
$ nova floating-ip-bulk-delete RANGE_TO_DELETE
```

For example:

```
$ nova floating-ip-bulk-delete 192.168.1.56/29
```

Launch and manage stacks using the CLI

The Orchestration service provides a template-based orchestration engine. Administrators can use the orchestration engine to create and manage OpenStack cloud infrastructure resources. For example, an administrator can define storage, networking, instances, and applications to use as a repeatable running environment.

Templates are used to create stacks, which are collections of resources. For example, a stack might include instances, floating IPs, volumes, security groups, or users. The Orchestration service offers access to all OpenStack core services through a single modular template, with additional orchestration capabilities such as auto-scaling and basic high availability.

For information about:

- basic creation and deletion of Orchestration stacks, refer to the OpenStack End User Guide
- **openstack** CLI, see the OpenStack Command-Line Interface Reference

> **Note**
>
> The `heat` CLI is deprecated in favor of `python-openstackclient`. For a Python library, continue using `python-heatclient`.

As an administrator, you can also carry out stack functions on behalf of your users. For example, to resume, suspend, or delete a stack, run:

```
$ openstack stack resume STACK
$ openstack stack suspend STACK
$ openstack stack delete STACK
```

Manage quotas

To prevent system capacities from being exhausted without notification, you can set up quotas. Quotas are operational limits. For example, the number of gigabytes allowed for each tenant can be controlled so that cloud resources are optimized. Quotas can be enforced at both the tenant (or project) and the tenant-user level.

Using the command-line interface, you can manage quotas for the OpenStack Compute service, the OpenStack Block Storage service, and the OpenStack Networking service.

The cloud operator typically changes default values because a tenant requires more than ten volumes or 1 TB on a compute node.

> **Note**
>
> To view all tenants (projects), run:
>
> ```
> $ openstack project list
> ```

```
+----------------------------------+----------+
| ID                               | Name     |
+----------------------------------+----------+
| e66d97ac1b704897853412fc8450f7b9 | admin    |
| bf4a37b885fe46bd86e999e50adad1d3 | services |
| 21bd1c7c95234fd28f589b60903606fa | tenant01 |
| f599c5cd1cba4125ae3d7caed08e288c | tenant02 |
+----------------------------------+----------+
```

- To display all current users for a tenant, run:

```
$ openstack user list --project PROJECT_NAME
+----------------------------------+--------+
| ID                               | Name   |
+----------------------------------+--------+
| ea30aa434ab24a139b0e85125ec8a217 | demo00 |
| 4f8113c1d838467cad0c2f337b3dfded | demo01 |
+----------------------------------+--------+
```

Use `openstack quota show` *`PROJECT_NAME`* to list all quotas for a project.

Use `openstack quota set` *`PROJECT_NAME --parameters`* to set quota values.

- Manage Compute service quotas
  - View and update Compute quotas for a tenant (project)
  - View and update Compute quotas for a tenant user
- Manage Block Storage service quotas
  - View Block Storage quotas
  - Edit and update Block Storage service quotas
  - Remove a service
- Manage Networking service quotas
  - Basic quota configuration
  - Configure per-tenant quotas

Analyze log files

Use the swift command-line client for Object Storage to analyze log files.

The swift client is simple to use, scalable, and flexible.

Use the swift client *-o* or *-output* option to get short answers to questions about logs.

You can use the *-o* or *--output* option with a single object download to redirect the command output to a specific file or to STDOUT (`-`). The ability to redirect the output to STDOUT enables you to pipe (`|`) data without saving it to disk first.

Upload and analyze log files

- This example assumes that `logtest` directory contains the following log files.

```
2010-11-16-21_access.log
2010-11-16-22_access.log
2010-11-15-21_access.log
2010-11-15-22_access.log
```

Each file uses the following line format.

```
Nov 15 21:53:52 lucid64 proxy-server - 127.0.0.1 15/Nov/2010/22/53/52
DELETE
/v1/AUTH_cd4f57824deb4248a533f2c28bf156d3/2eefc05599d44df38a7f18b0b42ffed
d HTTP/1.0 204 - \
 - test%3Atester%2CAUTH_tkcdab3c6296e249d7b7e2454ee57266ff - - -
txaba5984c-aac7-460e-b04b-afc43f0c6571 - 0.0432
```

- Change into the `logtest` directory:

```
$ cd logtest
```

- Upload the log files into the `logtest` container:

```
$ swift -A http://swift-auth.com:11000/v1.0 -U test:tester -K testing
upload logtest *.log

2010-11-16-21_access.log
2010-11-16-22_access.log
2010-11-15-21_access.log
2010-11-15-22_access.log
```

- Get statistics for the account:

```
$ swift -A http://swift-auth.com:11000/v1.0 -U test:tester -K testing \
-q stat

Account: AUTH_cd4f57824deb4248a533f2c28bf156d3
Containers: 1
Objects: 4
Bytes: 5888268
```

- Get statistics for the `logtest` container:

```
$ swift -A http://swift-auth.com:11000/v1.0 -U test:tester -K testing \
stat logtest

Account: AUTH_cd4f57824deb4248a533f2c28bf156d3
Container: logtest
Objects: 4
Bytes: 5864468
Read ACL:
Write ACL:
```

- List all objects in the logtest container:

```
$ swift -A http:///swift-auth.com:11000/v1.0 -U test:tester -K testing \
```

```
list logtest

2010-11-15-21_access.log
2010-11-15-22_access.log
2010-11-16-21_access.log
2010-11-16-22_access.log
```

Download and analyze an object

This example uses the *-o* option and a hyphen (`-`) to get information about an object.

Use the **swift download** command to download the object. On this command, stream the output to `awk` to break down requests by return code and the date `2200 on November 16th, 2010`.

Using the log line format, find the request type in column 9 and the return code in column 12.

After `awk` processes the output, it pipes it to `sort` and `uniq -c` to sum up the number of occurrences for each request type and return code combination.

- Download an object:

```
$ swift -A http://swift-auth.com:11000/v1.0 -U test:tester -K testing \
     download -o - logtest 2010-11-16-22_access.log | awk '{ print \
     $9"-"$12}' | sort | uniq -c

805 DELETE-204
12 DELETE-404
2 DELETE-409
723 GET-200
142 GET-204
74 GET-206
80 GET-304
34 GET-401
5 GET-403
18 GET-404
166 GET-412
2 GET-416
50 HEAD-200
17 HEAD-204
20 HEAD-401
8 HEAD-404
30 POST-202
25 POST-204
22 POST-400
6 POST-404
842 PUT-201
```

```
2 PUT-202
32 PUT-400
4 PUT-403
4 PUT-404
2 PUT-411
6 PUT-412
6 PUT-413
2 PUT-422
8 PUT-499
```

- Discover how many PUT requests are in each log file.

  Use a bash for loop with awk and swift with the *-o* or *--output* option and a hyphen (`-`) to discover how many PUT requests are in each log file.

  Run the **swift list** command to list objects in the logtest container. Then, for each item in the list, run the **swift download -o -** command. Pipe the output into grep to filter the PUT requests. Finally, pipe into `wc -l` to count the lines.

```
$ for f in `swift -A http://swift-auth.com:11000/v1.0 -U test:tester \
 -K testing list logtest` ; \
        do  echo -ne "PUTS - " ; swift -A \
        http://swift-auth.com:11000/v1.0 -U test:tester \
        -K testing download -o -  logtest $f | grep PUT | wc -l ; \
    done

2010-11-15-21_access.log - PUTS - 402
2010-11-15-22_access.log - PUTS - 1091
2010-11-16-21_access.log - PUTS - 892
2010-11-16-22_access.log - PUTS - 910
```

- List the object names that begin with a specified string.
- Run the **swift list -p 2010-11-15** command to list objects in the logtest container that begin with the `2010-11-15` string.
- For each item in the list, run the **swift download -o -** command.
- Pipe the output to **grep** and **wc**. Use the **echo** command to display the object name.

```
$ for f in `swift -A http://swift-auth.com:11000/v1.0 -U test:tester \
 -K testing list -p 2010-11-15 logtest` ; \
        do  echo -ne "$f - PUTS - " ; swift -A \
        http://127.0.0.1:11000/v1.0 -U test:tester \
        -K testing download -o - logtest $f | grep PUT | wc -l ; \
      done

2010-11-15-21_access.log - PUTS - 402
2010-11-15-22_access.log - PUTS - 910
```

Manage Block Storage scheduling

As an administrative user, you have some control over which volume back end your volumes reside on. You can specify affinity or anti-affinity between two volumes. Affinity between volumes means that they are stored on the same back end, whereas anti-affinity means that they are stored on different back ends.

For information on how to set up multiple back ends for Cinder, refer to *Configure multiple-storage back ends*.

Example Usages

- Create a new volume on the same back end as Volume_A:

  ```
  $ cinder create --hint same_host=Volume_A-UUID SIZE
  ```

- Create a new volume on a different back end than Volume_A:

  ```
  $ cinder create --hint different_host=Volume_A-UUID SIZE
  ```

- Create a new volume on the same back end as Volume_A and Volume_B:

  ```
  $ cinder create --hint same_host=Volume_A-UUID --hint same_host=Volume_B-UUID SIZE
  ```

  Or:

  ```
  $ cinder create --hint same_host="[Volume_A-UUID, Volume_B-UUID]" SIZE
  ```

- Create a new volume on a different back end than both Volume_A and Volume_B:

  ```
  $ cinder create --hint different_host=Volume_A-UUID --hint different_host=Volume_B-UUID SIZE
  ```

  Or:

  ```
  $ cinder create --hint different_host="[Volume_A-UUID, Volume_B-UUID]" SIZE
  ```

Cross-project features

Many features are common to all the OpenStack services and are consistent in their configuration and deployment patterns. Unless explicitly noted, you can safely assume that the features in this chapter are supported and configured in a consistent manner.

Cross-origin resource sharing

> **Note**
>
> This is a new feature in OpenStack Liberty.

OpenStack supports *Cross-Origin Resource Sharing (CORS)*, a W3C specification defining a contract by which the single-origin policy of a user agent (usually a browser) may be relaxed. It permits the javascript engine to access an API that does not reside on the same domain, protocol, or port.

This feature is most useful to organizations which maintain one or more custom user interfaces for OpenStack, as it permits those interfaces to access the services directly, rather than requiring an intermediate proxy server. It can, however, also be misused by malicious actors; please review the security advisory below for more information.

> **Note**
>
> Both the Object Storage and dashboard projects provide CORS support that is not covered by this document. For those, please refer to their respective implementations:
>
> - CORS in Object Storage
> - CORS in dashboard

Enabling CORS with configuration

In most cases, CORS support is built directly into the service itself. To enable it, simply follow the configuration options exposed in the default configuration file, or add it yourself according to the pattern below.

```
[cors]
allowed_origin = https://first_ui.example.com
max_age = 3600
allow_methods = GET,POST,PUT,DELETE
allow_headers = Content-Type,Cache-Control,Content-Language,Expires,Last-Modified,Pragma,X-Custom-Header
expose_headers = Content-Type,Cache-Control,Content-Language,Expires,Last-Modified,Pragma,X-Custom-Header
```

Additional origins can be explicitly added. To express this in your configuration file, first begin with a `[cors]` group as above, into which you place your default configuration values. Then, add as many additional configuration groups as necessary, naming them `[cors.{something}]` (each name must be unique). The purpose of the suffix to `cors.` is legibility, we recommend using a reasonable human-readable string:

```
[cors.ironic_webclient]
# CORS Configuration for a hypothetical ironic webclient, which overrides
# authentication
allowed_origin = https://ironic.example.com:443
allow_credentials = True

[cors.horizon]
# CORS Configuration for horizon, which uses global options.
allowed_origin = https://horizon.example.com:443

[cors.wildcard]
# CORS Configuration for the CORS specified domain wildcard, which only
# permits HTTP GET requests.
allowed_origin = *
allow_methods = GET
```

Enabling CORS with PasteDeploy

CORS can also be configured using PasteDeploy. First of all, ensure that OpenStack's `oslo_middleware` package (version 2.4.0 or later) is available in the Python environment that is running the service. Then, add the following configuration block to your `paste.ini` file.

```
[filter:cors]
paste.filter_factory = oslo_middleware.cors:filter_factory
allowed_origin = https://website.example.com:443
max_age = 3600
allow_methods = GET,POST,PUT,DELETE
allow_headers = Content-Type,Cache-Control,Content-Language,Expires,Last-Modified,Pragma,X-Custom-Header
expose_headers = Content-Type,Cache-Control,Content-Language,Expires,Last-Modified,Pragma,X-Custom-Header
```

> **Note**
>
> To add an additional domain in oslo_middleware v2.4.0, add another filter. In v3.0.0 and after, you may add multiple domains in the above `allowed_origin` field, separated by commas.

Security concerns

CORS specifies a wildcard character *, which permits access to all user agents, regardless of domain, protocol, or host. While there are valid use cases for this approach, it also permits a malicious actor to create a convincing facsimile of a user interface, and trick users into revealing authentication credentials. Please carefully evaluate your use case and the relevant documentation for any risk to your organization.

> **Note**
>
> The CORS specification does not support using this wildcard as a part of a URI. Setting `allowed_origin` to `*` would work, while `*.openstack.org` would not.

Troubleshooting

CORS is very easy to get wrong, as even one incorrect property will violate the prescribed contract. Here are some steps you can take to troubleshoot your configuration.

Check the service log

The CORS middleware used by OpenStack provides verbose debug logging that should reveal most configuration problems. Here are some example log messages, and how to resolve them.

Problem

```
CORS request from origin 'http://example.com' not permitted.
```

Solution

A request was received from the origin `http://example.com`, however this origin was not found in the permitted list. The cause may be a superfluous port notation (ports 80 and 443 do not need to be specified). To correct, ensure that the configuration property for this host is identical to the host indicated in the log message.

Problem

```
Request method 'DELETE' not in permitted list: GET,PUT,POST
```

Solution

A user agent has requested permission to perform a DELETE request, however the CORS configuration for the domain does not permit this. To correct, add this method to the `allow_methods` configuration property.

Problem

```
Request header 'X-Custom-Header' not in permitted list: X-Other-Header
```

Solution

A request was received with the header `X-Custom-Header`, which is not permitted. Add this header to the `allow_headers` configuration property.

Open your browser's console log

Most browsers provide helpful debug output when a CORS request is rejected. Usually this happens when a request was successful, but the return headers on the response do not permit access to a property which the browser is trying to access.

Manually construct a CORS request

By using `curl` or a similar tool, you can trigger a CORS response with a properly constructed HTTP request. An example request and response might look like this.

Request example:

```
$ curl -I -X OPTIONS https://api.example.com/api -H "Origin:
https://ui.example.com"
```

Response example:

```
HTTP/1.1 204 No Content
Content-Length: 0
Access-Control-Allow-Origin: https://ui.example.com
Access-Control-Allow-Methods: GET,POST,PUT,DELETE
Access-Control-Expose-Headers: origin,authorization,accept,x-total,x-limit,x-
marker,x-client,content-type
Access-Control-Allow-Headers: origin,authorization,accept,x-total,x-limit,x-
marker,x-client,content-type
Access-Control-Max-Age: 3600
```

If the service does not return any access control headers, check the service log, such as `/var/log/upstart/ironic-api.log` for an indication on what went wrong.

Community support

The following resources are available to help you run and use OpenStack. The OpenStack community constantly improves and adds to the main features of OpenStack, but if you have any questions, do not hesitate to ask. Use the following resources to get OpenStack support and troubleshoot your installations.

Documentation

For the available OpenStack documentation, see docs.openstack.org.

To provide feedback on documentation, join and use the openstack-docs@lists.openstack.org mailing list at OpenStack Documentation Mailing List, or report a bug.

The following books explain how to install an OpenStack cloud and its associated components:

- Installation Guide for openSUSE Leap 42.1 and SUSE Linux Enterprise Server 12 SP1
- Installation Guide for Red Hat Enterprise Linux 7 and CentOS 7
- Installation Guide for Ubuntu 14.04 (LTS)

The following books explain how to configure and run an OpenStack cloud:

- Architecture Design Guide
- Administrator Guide
- Configuration Reference
- Operations Guide

- Networking Guide
- High Availability Guide
- Security Guide
- Virtual Machine Image Guide

The following books explain how to use the OpenStack dashboard and command-line clients:

- API Guide
- End User Guide
- Command-Line Interface Reference

The following documentation provides reference and guidance information for the OpenStack APIs:

- API Complete Reference (HTML)
- API Complete Reference (PDF)

The following guide provides how to contribute to OpenStack documentation:

- Documentation Contributor Guide

ask.openstack.org

During the set up or testing of OpenStack, you might have questions about how a specific task is completed or be in a situation where a feature does not work correctly. Use the ask.openstack.org site to ask questions and get answers. When you visit the https://ask.openstack.org site, scan the recently asked questions to see whether your question has already been answered. If not, ask a new question. Be sure to give a clear, concise summary in the title and provide as much detail as possible in the description. Paste in your command output or stack traces, links to screen shots, and any other information which might be useful.

OpenStack mailing lists

A great way to get answers and insights is to post your question or problematic scenario to the OpenStack mailing list. You can learn from and help others who might have similar issues. To subscribe or view the archives, go to http://lists.openstack.org/cgi-bin/mailman/listinfo/openstack. If you are interested in the other mailing lists for specific projects or development, refer to Mailing Lists.

The OpenStack wiki

The OpenStack wiki contains a broad range of topics but some of the information can be difficult to find or is a few pages deep. Fortunately, the wiki search feature enables you to search by title or content. If you search for specific information, such as about networking or OpenStack Compute, you can find a large amount of relevant material. More is being added all the time, so be sure to check back often. You can find the search box in the upper-right corner of any OpenStack wiki page.

The Launchpad Bugs area

The OpenStack community values your set up and testing efforts and wants your feedback. To log a bug, you must sign up for a Launchpad account at https://launchpad.net/+login. You can view existing bugs and report bugs in the Launchpad Bugs area. Use the search feature to determine whether the bug has already been reported or already been fixed. If it still seems like your bug is unreported, fill out a bug report.

Some tips:

- Give a clear, concise summary.
- Provide as much detail as possible in the description. Paste in your command output or stack traces, links to screen shots, and any other information which might be useful.
- Be sure to include the software and package versions that you are using, especially if you are using a development branch, such as, `"Kilo release" vs git commit bc79c3ecc55929bac585d04a03475b72e06a3208`.
- Any deployment-specific information is helpful, such as whether you are using Ubuntu 14.04 or are performing a multi-node installation.

The following Launchpad Bugs areas are available:

- Bugs: OpenStack Block Storage (cinder)
- Bugs: OpenStack Compute (nova)
- Bugs: OpenStack Dashboard (horizon)
- Bugs: OpenStack Identity (keystone)
- Bugs: OpenStack Image service (glance)
- Bugs: OpenStack Networking (neutron)
- Bugs: OpenStack Object Storage (swift)
- Bugs: Application catalog (murano)
- Bugs: Bare metal service (ironic)

- Bugs: Clustering service (senlin)
- Bugs: Container Infrastructure Management service (magnum)
- Bugs: Data processing service (sahara)
- Bugs: Database service (trove)
- Bugs: Deployment service (fuel)
- Bugs: DNS service (designate)
- Bugs: Key Manager Service (barbican)
- Bugs: Monitoring (monasca)
- Bugs: Orchestration (heat)
- Bugs: Rating (cloudkitty)
- Bugs: Shared file systems (manila)
- Bugs: Telemetry (ceilometer)
- Bugs: Telemetry v3 (gnocchi)
- Bugs: Workflow service (mistral)
- Bugs: Messaging service (zaqar)
- Bugs: OpenStack API Documentation (developer.openstack.org)
- Bugs: OpenStack Documentation (docs.openstack.org)

The OpenStack IRC channel

The OpenStack community lives in the #openstack IRC channel on the Freenode network. You can hang out, ask questions, or get immediate feedback for urgent and pressing issues. To install an IRC client or use a browser-based client, go to https://webchat.freenode.net/. You can also use Colloquy (Mac OS X, http://colloquy.info/), mIRC (Windows, http://www.mirc.com/), or XChat (Linux). When you are in the IRC channel and want to share code or command output, the generally accepted method is to use a Paste Bin. The OpenStack project has one at http://paste.openstack.org. Just paste your longer amounts of text or logs in the web form and you get a URL that you can paste into the channel. The OpenStack IRC channel is `#openstack` on `irc.freenode.net`. You can find a list of all OpenStack IRC channels at https://wiki.openstack.org/wiki/IRC.

Documentation feedback

To provide feedback on documentation, join and use the openstack-docs@lists.openstack.org mailing list at OpenStack Documentation Mailing List, or report a bug.

OpenStack distribution packages

The following Linux distributions provide community-supported packages for OpenStack:

- **Debian:** https://wiki.debian.org/OpenStack
- **CentOS, Fedora, and Red Hat Enterprise Linux:** https://www.rdoproject.org/
- **openSUSE and SUSE Linux Enterprise Server:** https://en.opensuse.org/Portal:OpenStack
- **Ubuntu:** https://wiki.ubuntu.com/ServerTeam/CloudArchive

Glossary

This glossary offers a list of terms and definitions to define a vocabulary for OpenStack-related concepts.

To add to OpenStack glossary, clone the openstack/openstack-manuals repository and update the source file `doc/common/glossary.rst` through the OpenStack contribution process.

0-9

6to4
A mechanism that allows IPv6 packets to be transmitted over an IPv4 network, providing a strategy for migrating to IPv6.

A

absolute limit
Impassable limits for guest VMs. Settings include total RAM size, maximum number of vCPUs, and maximum disk size.

access control list
A list of permissions attached to an object. An ACL specifies which users or system processes have access to objects. It also defines which operations can be performed on specified objects. Each entry in a typical ACL specifies a subject and an operation. For instance, the ACL entry `(Alice, delete)` for a file gives Alice permission to delete the file.

access key
Alternative term for an Amazon EC2 access key. See EC2 access key.

account
The Object Storage context of an account. Do not confuse with a user account from an authentication service, such as Active Directory, /etc/passwd, OpenLDAP, OpenStack Identity, and so on.

account auditor
Checks for missing replicas and incorrect or corrupted objects in a specified Object Storage account by running queries against the back-end SQLite database.

account database
A SQLite database that contains Object Storage accounts and related metadata and that the accounts server accesses.

account reaper
An Object Storage worker that scans for and deletes account databases and that the account server has marked for deletion.

account server
Lists containers in Object Storage and stores container information in the account database.

account service
An Object Storage component that provides account services such as list, create, modify, and audit. Do not confuse with OpenStack Identity service, OpenLDAP, or similar user-account services.

accounting
The Compute service provides accounting information through the event notification and system usage data facilities.

ACL
See access control list.

active/active configuration
In a high-availability setup with an active/active configuration, several systems share the load together and if one fails, the load is distributed to the remaining systems.

Active Directory
Authentication and identity service by Microsoft, based on LDAP. Supported in OpenStack.

active/passive configuration
In a high-availability setup with an active/passive configuration, systems are set up to bring additional resources online to replace those that have failed.

address pool
A group of fixed and/or floating IP addresses that are assigned to a project and can be used by or assigned to the VM instances in a project.

admin API
A subset of API calls that are accessible to authorized administrators and are generally not accessible to end users or the public Internet. They can exist as a separate service (keystone) or can be a subset of another API (nova).

administrator
The person responsible for installing, configuring, and managing an OpenStack cloud.

admin server
In the context of the Identity service, the worker process that provides access to the admin API.

Advanced Message Queuing Protocol (AMQP)
The open standard messaging protocol used by OpenStack components for intra-service communications, provided by RabbitMQ, Qpid, or ZeroMQ.

Advanced RISC Machine (ARM)
Lower power consumption CPU often found in mobile and embedded devices. Supported by OpenStack.

alert
The Compute service can send alerts through its notification system, which includes a facility to create custom notification drivers. Alerts can be sent to and displayed on the horizon dashboard.

allocate
The process of taking a floating IP address from the address pool so it can be associated with a fixed IP on a guest VM instance.

Amazon Kernel Image (AKI)
Both a VM container format and disk format. Supported by Image service.

Amazon Machine Image (AMI)
Both a VM container format and disk format. Supported by Image service.

Amazon Ramdisk Image (ARI)
Both a VM container format and disk format. Supported by Image service.

Anvil
A project that ports the shell script-based project named DevStack to Python.

Apache
The Apache Software Foundation supports the Apache community of open-source software projects. These projects provide software products for the public good.

Apache License 2.0
All OpenStack core projects are provided under the terms of the Apache License 2.0 license.

Apache Web Server
The most common web server software currently used on the Internet.

API endpoint
The daemon, worker, or service that a client communicates with to access an API. API endpoints can provide any number of services, such as authentication, sales data, performance meters, Compute VM commands, census data, and so on.

API extension
Custom modules that extend some OpenStack core APIs.

API extension plug-in
Alternative term for a Networking plug-in or Networking API extension.

API key

Alternative term for an API token.

API server

Any node running a daemon or worker that provides an API endpoint.

API token

Passed to API requests and used by OpenStack to verify that the client is authorized to run the requested operation.

API version

In OpenStack, the API version for a project is part of the URL. For example, `example.com/nova/v1/foobar`.

applet

A Java program that can be embedded into a web page.

Application Programming Interface (API)

A collection of specifications used to access a service, application, or program. Includes service calls, required parameters for each call, and the expected return values.

Application Catalog service

OpenStack project that provides an application catalog service so that users can compose and deploy composite environments on an application abstraction level while managing the application lifecycle. The code name of the project is murano.

application server

A piece of software that makes available another piece of software over a network.

Application Service Provider (ASP)

Companies that rent specialized applications that help businesses and organizations provide additional services with lower cost.

Address Resolution Protocol (ARP)

The protocol by which layer-3 IP addresses are resolved into layer-2 link local addresses.

arptables

Tool used for maintaining Address Resolution Protocol packet filter rules in the Linux kernel firewall modules. Used along with iptables, ebtables, and ip6tables in Compute to provide firewall services for VMs.

associate

The process associating a Compute floating IP address with a fixed IP address.

Asynchronous JavaScript and XML (AJAX)

A group of interrelated web development techniques used on the client-side to create asynchronous web applications. Used extensively in horizon.

ATA over Ethernet (AoE)

A disk storage protocol tunneled within Ethernet.

attach

The process of connecting a VIF or vNIC to a L2 network in Networking. In the context of Compute, this process connects a storage volume to an instance.

attachment (network)
Association of an interface ID to a logical port. Plugs an interface into a port.

auditing
Provided in Compute through the system usage data facility.

auditor
A worker process that verifies the integrity of Object Storage objects, containers, and accounts. Auditors is the collective term for the Object Storage account auditor, container auditor, and object auditor.

Austin
The code name for the initial release of OpenStack. The first design summit took place in Austin, Texas, US.

auth node
Alternative term for an Object Storage authorization node.

authentication
The process that confirms that the user, process, or client is really who they say they are through private key, secret token, password, fingerprint, or similar method.

authentication token
A string of text provided to the client after authentication. Must be provided by the user or process in subsequent requests to the API endpoint.

AuthN
The Identity service component that provides authentication services.

authorization
The act of verifying that a user, process, or client is authorized to perform an action.

authorization node
An Object Storage node that provides authorization services.

AuthZ
The Identity component that provides high-level authorization services.

Auto ACK
Configuration setting within RabbitMQ that enables or disables message acknowledgment. Enabled by default.

auto declare
A Compute RabbitMQ setting that determines whether a message exchange is automatically created when the program starts.

availability zone
An Amazon EC2 concept of an isolated area that is used for fault tolerance. Do not confuse with an OpenStack Compute zone or cell.

AWS
Amazon Web Services.

AWS CloudFormation template
AWS CloudFormation allows AWS users to create and manage a collection of related resources. The Orchestration service supports a CloudFormation-compatible format (CFN).

B

back end
Interactions and processes that are obfuscated from the user, such as Compute volume mount, data transmission to an iSCSI target by a daemon, or Object Storage object integrity checks.

back-end catalog
The storage method used by the Identity service catalog service to store and retrieve information about API endpoints that are available to the client. Examples include an SQL database, LDAP database, or KVS back end.

back-end store
The persistent data store used to save and retrieve information for a service, such as lists of Object Storage objects, current state of guest VMs, lists of user names, and so on. Also, the method that the Image service uses to get and store VM images. Options include Object Storage, local file system, S3, and HTTP.

backup restore and disaster recovery as a service
The OpenStack project that provides integrated tooling for backing up, restoring, and recovering file systems, instances, or database backups. The project name is freezer.

bandwidth
The amount of available data used by communication resources, such as the Internet. Represents the amount of data that is used to download things or the amount of data available to download.

barbican
Code name of the key management service for OpenStack.

bare
An Image service container format that indicates that no container exists for the VM image.

Bare Metal service
OpenStack project that provisions bare metal, as opposed to virtual machines. The code name for the project is ironic.

base image
An OpenStack-provided image.

Bell-LaPadula model
A security model that focuses on data confidentiality and controlled access to classified information. This model divides the entities into subjects and objects. The clearance of a subject is compared to the classification of the object to determine if the subject is authorized for the specific access mode. The clearance or classification scheme is expressed in terms of a lattice.

Benchmark service
OpenStack project that provides a framework for performance analysis and benchmarking of individual OpenStack components as well as full production

OpenStack cloud deployments. The code name of the project is rally.

Bexar

A grouped release of projects related to OpenStack that came out in February of 2011. It included only Compute (nova) and Object Storage (swift). Bexar is the code name for the second release of OpenStack. The design summit took place in San Antonio, Texas, US, which is the county seat for Bexar county.

binary

Information that consists solely of ones and zeroes, which is the language of computers.

bit

A bit is a single digit number that is in base of 2 (either a zero or one). Bandwidth usage is measured in bits per second.

bits per second (BPS)

The universal measurement of how quickly data is transferred from place to place.

block device

A device that moves data in the form of blocks. These device nodes interface the devices, such as hard disks, CD-ROM drives, flash drives, and other addressable regions of memory.

block migration

A method of VM live migration used by KVM to evacuate instances from one host to another with very little downtime during a user-initiated switchover. Does not require shared storage. Supported by Compute.

Block Storage service

The OpenStack core project that enables management of volumes, volume snapshots, and volume types. The project name of Block Storage is cinder.

Block Storage API

An API on a separate endpoint for attaching, detaching, and creating block storage for compute VMs.

BMC

Baseboard Management Controller. The intelligence in the IPMI architecture, which is a specialized micro-controller that is embedded on the motherboard of a computer and acts as a server. Manages the interface between system management software and platform hardware.

bootable disk image

A type of VM image that exists as a single, bootable file.

Bootstrap Protocol (BOOTP)

A network protocol used by a network client to obtain an IP address from a configuration server. Provided in Compute through the dnsmasq daemon when using either the FlatDHCP manager or VLAN manager network manager.

Border Gateway Protocol (BGP)

The Border Gateway Protocol is a dynamic routing protocol that connects autonomous systems. Considered the backbone of the Internet, this protocol connects disparate

networks to form a larger network.

browser

Any client software that enables a computer or device to access the Internet.

builder file

Contains configuration information that Object Storage uses to reconfigure a ring or to re-create it from scratch after a serious failure.

bursting

The practice of utilizing a secondary environment to elastically build instances on-demand when the primary environment is resource constrained.

button class

A group of related button types within horizon. Buttons to start, stop, and suspend VMs are in one class. Buttons to associate and disassociate floating IP addresses are in another class, and so on.

byte

Set of bits that make up a single character; there are usually 8 bits to a byte.

C

CA

Certificate Authority or Certification Authority. In cryptography, an entity that issues digital certificates. The digital certificate certifies the ownership of a public key by the named subject of the certificate. This enables others (relying parties) to rely upon signatures or assertions made by the private key that corresponds to the certified public key. In this model of trust relationships, a CA is a trusted third party for both the subject (owner) of the certificate and the party relying upon the certificate. CAs are characteristic of many public key infrastructure (PKI) schemes.

cache pruner

A program that keeps the Image service VM image cache at or below its configured maximum size.

Cactus

An OpenStack grouped release of projects that came out in the spring of 2011. It included Compute (nova), Object Storage (swift), and the Image service (glance). Cactus is a city in Texas, US and is the code name for the third release of OpenStack. When OpenStack releases went from three to six months long, the code name of the release changed to match a geography nearest the previous summit.

CADF

Cloud Auditing Data Federation (CADF) is a specification for audit event data. CADF is supported by OpenStack Identity.

CALL

One of the RPC primitives used by the OpenStack message queue software. Sends a message and waits for a response.

capability
Defines resources for a cell, including CPU, storage, and networking. Can apply to the specific services within a cell or a whole cell.

capacity cache
A Compute back-end database table that contains the current workload, amount of free RAM, and number of VMs running on each host. Used to determine on which host a VM starts.

capacity updater
A notification driver that monitors VM instances and updates the capacity cache as needed.

CAST
One of the RPC primitives used by the OpenStack message queue software. Sends a message and does not wait for a response.

catalog
A list of API endpoints that are available to a user after authentication with the Identity service.

catalog service
An Identity service that lists API endpoints that are available to a user after authentication with the Identity service.

ceilometer
The project name for the Telemetry service, which is an integrated project that provides metering and measuring facilities for OpenStack.

cell
Provides logical partitioning of Compute resources in a child and parent relationship. Requests are passed from parent cells to child cells if the parent cannot provide the requested resource.

cell forwarding
A Compute option that enables parent cells to pass resource requests to child cells if the parent cannot provide the requested resource.

cell manager
The Compute component that contains a list of the current capabilities of each host within the cell and routes requests as appropriate.

CentOS
A Linux distribution that is compatible with OpenStack.

Ceph
Massively scalable distributed storage system that consists of an object store, block store, and POSIX-compatible distributed file system. Compatible with OpenStack.

CephFS
The POSIX-compliant file system provided by Ceph.

certificate authority
A simple certificate authority provided by Compute for cloudpipe VPNs and VM image decryption.

Challenge-Handshake Authentication Protocol (CHAP)
An iSCSI authentication method supported by Compute.

chance scheduler
A scheduling method used by Compute that randomly chooses an available host from the pool.

changes since
A Compute API parameter that downloads changes to the requested item since your last request, instead of downloading a new, fresh set of data and comparing it against the old data.

Chef
An operating system configuration management tool supporting OpenStack deployments.

child cell
If a requested resource such as CPU time, disk storage, or memory is not available in the parent cell, the request is forwarded to its associated child cells. If the child cell can fulfill the request, it does. Otherwise, it attempts to pass the request to any of its children.

cinder
A core OpenStack project that provides block storage services for VMs.

CirrOS
A minimal Linux distribution designed for use as a test image on clouds such as OpenStack.

Cisco neutron plug-in
A Networking plug-in for Cisco devices and technologies, including UCS and Nexus.

cloud architect
A person who plans, designs, and oversees the creation of clouds.

cloud computing
A model that enables access to a shared pool of configurable computing resources, such as networks, servers, storage, applications, and services, that can be rapidly provisioned and released with minimal management effort or service provider interaction.

cloud controller
Collection of Compute components that represent the global state of the cloud; talks to services, such as Identity authentication, Object Storage, and node/storage workers through a queue.

cloud controller node
A node that runs network, volume, API, scheduler, and image services. Each service may be broken out into separate nodes for scalability or availability.

Cloud Data Management Interface (CDMI)
SINA standard that defines a RESTful API for managing objects in the cloud, currently unsupported in OpenStack.

Cloud Infrastructure Management Interface (CIMI)
An in-progress specification for cloud management. Currently unsupported in OpenStack.

cloud-init
A package commonly installed in VM images that performs initialization of an instance after boot using information that it retrieves from the metadata service, such as the SSH public key and user data.

cloudadmin
One of the default roles in the Compute RBAC system. Grants complete system access.

Cloudbase-Init
A Windows project providing guest initialization features, similar to cloud-init.

cloudpipe
A compute service that creates VPNs on a per-project basis.

cloudpipe image
A pre-made VM image that serves as a cloudpipe server. Essentially, OpenVPN running on Linux.

Clustering service
The OpenStack project that implements clustering services and libraries for the management of groups of homogeneous objects exposed by other OpenStack services. The project name of Clustering service is senlin.

CMDB
Configuration Management Database.

congress
OpenStack project that provides the Governance service.

command filter
Lists allowed commands within the Compute rootwrap facility.

Common Internet File System (CIFS)
A file sharing protocol. It is a public or open variation of the original Server Message Block (SMB) protocol developed and used by Microsoft. Like the SMB protocol, CIFS runs at a higher level and uses the TCP/IP protocol.

community project
A project that is not officially endorsed by the OpenStack Foundation. If the project is successful enough, it might be elevated to an incubated project and then to a core project, or it might be merged with the main code trunk.

compression
Reducing the size of files by special encoding, the file can be decompressed again to its original content. OpenStack supports compression at the Linux file system level but does not support compression for things such as Object Storage objects or Image service VM images.

Compute service
The OpenStack core project that provides compute services. The project name of

Compute service is nova.

Compute API

The nova-api daemon provides access to nova services. Can communicate with other APIs, such as the Amazon EC2 API.

compute controller

The Compute component that chooses suitable hosts on which to start VM instances.

compute host

Physical host dedicated to running compute nodes.

compute node

A node that runs the nova-compute daemon that manages VM instances that provide a wide range of services, such as web applications and analytics.

Compute service

Name for the Compute component that manages VMs.

compute worker

The Compute component that runs on each compute node and manages the VM instance lifecycle, including run, reboot, terminate, attach/detach volumes, and so on. Provided by the nova-compute daemon.

concatenated object

A set of segment objects that Object Storage combines and sends to the client.

conductor

In Compute, conductor is the process that proxies database requests from the compute process. Using conductor improves security because compute nodes do not need direct access to the database.

consistency window

The amount of time it takes for a new Object Storage object to become accessible to all clients.

console log

Contains the output from a Linux VM console in Compute.

container

Organizes and stores objects in Object Storage. Similar to the concept of a Linux directory but cannot be nested. Alternative term for an Image service container format.

container auditor

Checks for missing replicas or incorrect objects in specified Object Storage containers through queries to the SQLite back-end database.

container database

A SQLite database that stores Object Storage containers and container metadata. The container server accesses this database.

container format

A wrapper used by the Image service that contains a VM image and its associated metadata, such as machine state, OS disk size, and so on.

Container Infrastructure Management service
To provide a set of services for provisioning, scaling, and managing container orchestration engines.

container server
An Object Storage server that manages containers.

container service
The Object Storage component that provides container services, such as create, delete, list, and so on.

content delivery network (CDN)
A content delivery network is a specialized network that is used to distribute content to clients, typically located close to the client for increased performance.

controller node
Alternative term for a cloud controller node.

core API
Depending on context, the core API is either the OpenStack API or the main API of a specific core project, such as Compute, Networking, Image service, and so on.

core service
An official OpenStack service defined as core by DefCore Committee. Currently, consists of Block Storage service (cinder), Compute service (nova), Identity service (keystone), Image service (glance), Networking service (neutron), and Object Storage service (swift).

cost
Under the Compute distributed scheduler, this is calculated by looking at the capabilities of each host relative to the flavor of the VM instance being requested.

credentials
Data that is only known to or accessible by a user and used to verify that the user is who he says he is. Credentials are presented to the server during authentication. Examples include a password, secret key, digital certificate, and fingerprint.

Cross-Origin Resource Sharing (CORS)
A mechanism that allows many resources (for example, fonts, JavaScript) on a web page to be requested from another domain outside the domain from which the resource originated. In particular, JavaScript's AJAX calls can use the XMLHttpRequest mechanism.

Crowbar
An open source community project by Dell that aims to provide all necessary services to quickly deploy clouds.

current workload
An element of the Compute capacity cache that is calculated based on the number of build, snapshot, migrate, and resize operations currently in progress on a given host.

customer
Alternative term for tenant.

customization module
A user-created Python module that is loaded by horizon to change the look and feel of the dashboard.

D

daemon
A process that runs in the background and waits for requests. May or may not listen on a TCP or UDP port. Do not confuse with a worker.

DAC
Discretionary access control. Governs the ability of subjects to access objects, while enabling users to make policy decisions and assign security attributes. The traditional UNIX system of users, groups, and read-write-execute permissions is an example of DAC.

Dashboard
The web-based management interface for OpenStack. An alternative name for horizon.

data encryption
Both Image service and Compute support encrypted virtual machine (VM) images (but not instances). In-transit data encryption is supported in OpenStack using technologies such as HTTPS, SSL, TLS, and SSH. Object Storage does not support object encryption at the application level but may support storage that uses disk encryption.

database ID
A unique ID given to each replica of an Object Storage database.

database replicator
An Object Storage component that copies changes in the account, container, and object databases to other nodes.

Database service
An integrated project that provide scalable and reliable Cloud Database-as-a-Service functionality for both relational and non-relational database engines. The project name of Database service is trove.

Data loss prevention (DLP) software
Software programs used to protect sensitive information and prevent it from leaking outside a network boundary through the detection and denying of the data transportation.

Data Processing service
OpenStack project that provides a scalable data-processing stack and associated management interfaces. The code name for the project is sahara.

data store
A database engine supported by the Database service.

deallocate
The process of removing the association between a floating IP address and a fixed IP

address. Once this association is removed, the floating IP returns to the address pool.

Debian

A Linux distribution that is compatible with OpenStack.

deduplication

The process of finding duplicate data at the disk block, file, and/or object level to minimize storage use—currently unsupported within OpenStack.

default panel

The default panel that is displayed when a user accesses the horizon dashboard.

default tenant

New users are assigned to this tenant if no tenant is specified when a user is created.

default token

An Identity service token that is not associated with a specific tenant and is exchanged for a scoped token.

delayed delete

An option within Image service so that an image is deleted after a predefined number of seconds instead of immediately.

delivery mode

Setting for the Compute RabbitMQ message delivery mode; can be set to either transient or persistent.

denial of service (DoS)

Denial of service (DoS) is a short form for denial-of-service attack. This is a malicious attempt to prevent legitimate users from using a service.

deprecated auth

An option within Compute that enables administrators to create and manage users through the `nova-manage` command as opposed to using the Identity service.

designate

Code name for the DNS service project for OpenStack.

Desktop-as-a-Service

A platform that provides a suite of desktop environments that users access to receive a desktop experience from any location. This may provide general use, development, or even homogeneous testing environments.

developer

One of the default roles in the Compute RBAC system and the default role assigned to a new user.

device ID

Maps Object Storage partitions to physical storage devices.

device weight

Distributes partitions proportionately across Object Storage devices based on the storage capacity of each device.

DevStack

Community project that uses shell scripts to quickly build complete OpenStack development environments.

DHCP

Dynamic Host Configuration Protocol. A network protocol that configures devices that are connected to a network so that they can communicate on that network by using the Internet Protocol (IP). The protocol is implemented in a client-server model where DHCP clients request configuration data, such as an IP address, a default route, and one or more DNS server addresses from a DHCP server.

DHCP agent

OpenStack Networking agent that provides DHCP services for virtual networks.

Diablo

A grouped release of projects related to OpenStack that came out in the fall of 2011, the fourth release of OpenStack. It included Compute (nova 2011.3), Object Storage (swift 1.4.3), and the Image service (glance). Diablo is the code name for the fourth release of OpenStack. The design summit took place in the Bay Area near Santa Clara, California, US and Diablo is a nearby city.

direct consumer

An element of the Compute RabbitMQ that comes to life when a RPC call is executed. It connects to a direct exchange through a unique exclusive queue, sends the message, and terminates.

direct exchange

A routing table that is created within the Compute RabbitMQ during RPC calls; one is created for each RPC call that is invoked.

direct publisher

Element of RabbitMQ that provides a response to an incoming MQ message.

disassociate

The process of removing the association between a floating IP address and fixed IP and thus returning the floating IP address to the address pool.

disk encryption

The ability to encrypt data at the file system, disk partition, or whole-disk level. Supported within Compute VMs.

disk format

The underlying format that a disk image for a VM is stored as within the Image service back-end store. For example, AMI, ISO, QCOW2, VMDK, and so on.

dispersion

In Object Storage, tools to test and ensure dispersion of objects and containers to ensure fault tolerance.

distributed virtual router (DVR)

Mechanism for highly-available multi-host routing when using OpenStack Networking (neutron).

Django

A web framework used extensively in horizon.

DNS

Domain Name System. A hierarchical and distributed naming system for computers,

services, and resources connected to the Internet or a private network. Associates human-friendly, domain names to IP addresses.

DNS record

A record that specifies information about a particular domain and belongs to the domain.

DNS service

OpenStack project that provides scalable, on demand, self service access to authoritative DNS services, in a technology-agnostic manner. The code name for the project is designate.

dnsmasq

Daemon that provides DNS, DHCP, BOOTP, and TFTP services for virtual networks.

domain

An Identity API v3 entity. Represents a collection of projects, groups and users that defines administrative boundaries for managing OpenStack Identity entities. On the Internet, separates a website from other sites. Often, the domain name has two or more parts that are separated by dots. For example, yahoo.com, usa.gov, harvard.edu, or mail.yahoo.com. Also, a domain is an entity or container of all DNS-related information containing one or more records.

Domain Name System (DNS)

A system by which Internet domain name-to-address and address-to-name resolutions are determined. DNS helps navigate the Internet by translating the IP address into an address that is easier to remember. For example, translating 111.111.111.1 into www.yahoo.com. All domains and their components, such as mail servers, utilize DNS to resolve to the appropriate locations. DNS servers are usually set up in a master-slave relationship such that failure of the master invokes the slave. DNS servers might also be clustered or replicated such that changes made to one DNS server are automatically propagated to other active servers. In Compute, the support that enables associating DNS entries with floating IP addresses, nodes, or cells so that hostnames are consistent across reboots.

download

The transfer of data, usually in the form of files, from one computer to another.

DRTM

Dynamic root of trust measurement.

durable exchange

The Compute RabbitMQ message exchange that remains active when the server restarts.

durable queue

A Compute RabbitMQ message queue that remains active when the server restarts.

Dynamic Host Configuration Protocol (DHCP)

A method to automatically configure networking for a host at boot time. Provided by both Networking and Compute.

Dynamic HyperText Markup Language (DHTML)
Pages that use HTML, JavaScript, and Cascading Style Sheets to enable users to interact with a web page or show simple animation.

E

east-west traffic
Network traffic between servers in the same cloud or data center. See also north-south traffic.

EBS boot volume
An Amazon EBS storage volume that contains a bootable VM image, currently unsupported in OpenStack.

ebtables
Filtering tool for a Linux bridging firewall, enabling filtering of network traffic passing through a Linux bridge. Used in Compute along with arptables, iptables, and ip6tables to ensure isolation of network communications.

EC2
The Amazon commercial compute product, similar to Compute.

EC2 access key
Used along with an EC2 secret key to access the Compute EC2 API.

EC2 API
OpenStack supports accessing the Amazon EC2 API through Compute.

EC2 Compatibility API
A Compute component that enables OpenStack to communicate with Amazon EC2.

EC2 secret key
Used along with an EC2 access key when communicating with the Compute EC2 API; used to digitally sign each request.

Elastic Block Storage (EBS)
The Amazon commercial block storage product.

encryption
OpenStack supports encryption technologies such as HTTPS, SSH, SSL, TLS, digital certificates, and data encryption.

endpoint
See API endpoint.

endpoint registry
Alternative term for an Identity service catalog.

encapsulation
The practice of placing one packet type within another for the purposes of abstracting or securing data. Examples include GRE, MPLS, or IPsec.

endpoint template
A list of URL and port number endpoints that indicate where a service, such as Object Storage, Compute, Identity, and so on, can be accessed.

entity
Any piece of hardware or software that wants to connect to the network services provided by Networking, the network connectivity service. An entity can make use of Networking by implementing a VIF.

ephemeral image
A VM image that does not save changes made to its volumes and reverts them to their original state after the instance is terminated.

ephemeral volume
Volume that does not save the changes made to it and reverts to its original state when the current user relinquishes control.

Essex
A grouped release of projects related to OpenStack that came out in April 2012, the fifth release of OpenStack. It included Compute (nova 2012.1), Object Storage (swift 1.4.8), Image (glance), Identity (keystone), and Dashboard (horizon). Essex is the code name for the fifth release of OpenStack. The design summit took place in Boston, Massachusetts, US and Essex is a nearby city.

ESXi
An OpenStack-supported hypervisor.

ETag
MD5 hash of an object within Object Storage, used to ensure data integrity.

euca2ools
A collection of command-line tools for administering VMs; most are compatible with OpenStack.

Eucalyptus Kernel Image (EKI)
Used along with an ERI to create an EMI.

Eucalyptus Machine Image (EMI)
VM image container format supported by Image service.

Eucalyptus Ramdisk Image (ERI)
Used along with an EKI to create an EMI.

evacuate
The process of migrating one or all virtual machine (VM) instances from one host to another, compatible with both shared storage live migration and block migration.

exchange
Alternative term for a RabbitMQ message exchange.

exchange type
A routing algorithm in the Compute RabbitMQ.

exclusive queue
Connected to by a direct consumer in RabbitMQ—Compute, the message can be consumed only by the current connection.

extended attributes (xattr)
File system option that enables storage of additional information beyond owner, group, permissions, modification time, and so on. The underlying Object Storage file system must support extended attributes.

extension
Alternative term for an API extension or plug-in. In the context of Identity service, this is a call that is specific to the implementation, such as adding support for OpenID.

external network
A network segment typically used for instance Internet access.

extra specs
Specifies additional requirements when Compute determines where to start a new instance. Examples include a minimum amount of network bandwidth or a GPU.

F

FakeLDAP
An easy method to create a local LDAP directory for testing Identity and Compute. Requires Redis.

fan-out exchange
Within RabbitMQ and Compute, it is the messaging interface that is used by the scheduler service to receive capability messages from the compute, volume, and network nodes.

federated identity
A method to establish trusts between identity providers and the OpenStack cloud.

Fedora
A Linux distribution compatible with OpenStack.

Fibre Channel
Storage protocol similar in concept to TCP/IP; encapsulates SCSI commands and data.

Fibre Channel over Ethernet (FCoE)
The fibre channel protocol tunneled within Ethernet.

fill-first scheduler
The Compute scheduling method that attempts to fill a host with VMs rather than starting new VMs on a variety of hosts.

filter
The step in the Compute scheduling process when hosts that cannot run VMs are eliminated and not chosen.

firewall
Used to restrict communications between hosts and/or nodes, implemented in Compute using iptables, arptables, ip6tables, and ebtables.

FWaaS
A Networking extension that provides perimeter firewall functionality.

fixed IP address
An IP address that is associated with the same instance each time that instance boots, is generally not accessible to end users or the public Internet, and is used for management of the instance.

Flat Manager
The Compute component that gives IP addresses to authorized nodes and assumes DHCP, DNS, and routing configuration and services are provided by something else.

flat mode injection
A Compute networking method where the OS network configuration information is injected into the VM image before the instance starts.

flat network
Virtual network type that uses neither VLANs nor tunnels to segregate tenant traffic. Each flat network typically requires a separate underlying physical interface defined by bridge mappings. However, a flat network can contain multiple subnets.

FlatDHCP Manager
The Compute component that provides dnsmasq (DHCP, DNS, BOOTP, TFTP) and radvd (routing) services.

flavor
Alternative term for a VM instance type.

flavor ID
UUID for each Compute or Image service VM flavor or instance type.

floating IP address
An IP address that a project can associate with a VM so that the instance has the same public IP address each time that it boots. You create a pool of floating IP addresses and assign them to instances as they are launched to maintain a consistent IP address for maintaining DNS assignment.

Folsom
A grouped release of projects related to OpenStack that came out in the fall of 2012, the sixth release of OpenStack. It includes Compute (nova), Object Storage (swift), Identity (keystone), Networking (neutron), Image service (glance), and Volumes or Block Storage (cinder). Folsom is the code name for the sixth release of OpenStack. The design summit took place in San Francisco, California, US and Folsom is a nearby city.

FormPost
Object Storage middleware that uploads (posts) an image through a form on a web page.

freezer
OpenStack project that provides backup restore and disaster recovery as a service.

front end
The point where a user interacts with a service; can be an API endpoint, the horizon dashboard, or a command-line tool.

G

gateway
An IP address, typically assigned to a router, that passes network traffic between different networks.

generic receive offload (GRO)
Feature of certain network interface drivers that combines many smaller received packets into a large packet before delivery to the kernel IP stack.

generic routing encapsulation (GRE)
Protocol that encapsulates a wide variety of network layer protocols inside virtual point-to-point links.

glance
A core project that provides the OpenStack Image service.

glance API server
Processes client requests for VMs, updates Image service metadata on the registry server, and communicates with the store adapter to upload VM images from the back-end store.

glance registry
Alternative term for the Image service image registry.

global endpoint template
The Identity service endpoint template that contains services available to all tenants.

GlusterFS
A file system designed to aggregate NAS hosts, compatible with OpenStack.

golden image
A method of operating system installation where a finalized disk image is created and then used by all nodes without modification.

Governance service
OpenStack project to provide Governance-as-a-Service across any collection of cloud services in order to monitor, enforce, and audit policy over dynamic infrastructure. The code name for the project is congress.

Graphic Interchange Format (GIF)
A type of image file that is commonly used for animated images on web pages.

Graphics Processing Unit (GPU)
Choosing a host based on the existence of a GPU is currently unsupported in OpenStack.

Green Threads
The cooperative threading model used by Python; reduces race conditions and only context switches when specific library calls are made. Each OpenStack service is its own thread.

Grizzly
The code name for the seventh release of OpenStack. The design summit took place in

San Diego, California, US and Grizzly is an element of the state flag of California.

Group

An Identity v3 API entity. Represents a collection of users that is owned by a specific domain.

guest OS

An operating system instance running under the control of a hypervisor.

H

Hadoop

Apache Hadoop is an open source software framework that supports data-intensive distributed applications.

Hadoop Distributed File System (HDFS)

A distributed, highly fault-tolerant file system designed to run on low-cost commodity hardware.

handover

An object state in Object Storage where a new replica of the object is automatically created due to a drive failure.

hard reboot

A type of reboot where a physical or virtual power button is pressed as opposed to a graceful, proper shutdown of the operating system.

Havana

The code name for the eighth release of OpenStack. The design summit took place in Portland, Oregon, US and Havana is an unincorporated community in Oregon.

heat

An integrated project that aims to orchestrate multiple cloud applications for OpenStack.

Heat Orchestration Template (HOT)

Heat input in the format native to OpenStack.

health monitor

Determines whether back-end members of a VIP pool can process a request. A pool can have several health monitors associated with it. When a pool has several monitors associated with it, all monitors check each member of the pool. All monitors must declare a member to be healthy for it to stay active.

high availability (HA)

A high availability system design approach and associated service implementation ensures that a prearranged level of operational performance will be met during a contractual measurement period. High availability systems seek to minimize system downtime and data loss.

horizon

OpenStack project that provides a dashboard, which is a web interface.

horizon plug-in

A plug-in for the OpenStack dashboard (horizon).

host

A physical computer, not a VM instance (node).

host aggregate

A method to further subdivide availability zones into hypervisor pools, a collection of common hosts.

Host Bus Adapter (HBA)

Device plugged into a PCI slot, such as a fibre channel or network card.

hybrid cloud

A hybrid cloud is a composition of two or more clouds (private, community or public) that remain distinct entities but are bound together, offering the benefits of multiple deployment models. Hybrid cloud can also mean the ability to connect colocation, managed and/or dedicated services with cloud resources.

Hyper-V

One of the hypervisors supported by OpenStack.

hyperlink

Any kind of text that contains a link to some other site, commonly found in documents where clicking on a word or words opens up a different website.

Hypertext Transfer Protocol (HTTP)

An application protocol for distributed, collaborative, hypermedia information systems. It is the foundation of data communication for the World Wide Web. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

Hypertext Transfer Protocol Secure (HTTPS)

An encrypted communications protocol for secure communication over a computer network, with especially wide deployment on the Internet. Technically, it is not a protocol in and of itself; rather, it is the result of simply layering the Hypertext Transfer Protocol (HTTP) on top of the TLS or SSL protocol, thus adding the security capabilities of TLS or SSL to standard HTTP communications. Most OpenStack API endpoints and many inter-component communications support HTTPS communication.

hypervisor

Software that arbitrates and controls VM access to the actual underlying hardware.

hypervisor pool

A collection of hypervisors grouped together through host aggregates.

IaaS

Infrastructure-as-a-Service. IaaS is a provisioning model in which an organization outsources physical components of a data center, such as storage, hardware, servers,

and networking components. A service provider owns the equipment and is responsible for housing, operating and maintaining it. The client typically pays on a per-use basis. IaaS is a model for providing cloud services.

Icehouse

The code name for the ninth release of OpenStack. The design summit took place in Hong Kong and Ice House is a street in that city.

ICMP

Internet Control Message Protocol, used by network devices for control messages. For example, **ping** uses ICMP to test connectivity.

ID number

Unique numeric ID associated with each user in Identity, conceptually similar to a Linux or LDAP UID.

Identity API

Alternative term for the Identity service API.

Identity back end

The source used by Identity service to retrieve user information; an OpenLDAP server, for example.

identity provider

A directory service, which allows users to login with a user name and password. It is a typical source of authentication tokens.

Identity service

The OpenStack core project that provides a central directory of users mapped to the OpenStack services they can access. It also registers endpoints for OpenStack services. It acts as a common authentication system. The project name of Identity is keystone.

Identity service API

The API used to access the OpenStack Identity service provided through keystone.

IDS

Intrusion Detection System.

image

A collection of files for a specific operating system (OS) that you use to create or rebuild a server. OpenStack provides pre-built images. You can also create custom images, or snapshots, from servers that you have launched. Custom images can be used for data backups or as “gold” images for additional servers.

Image API

The Image service API endpoint for management of VM images.

image cache

Used by Image service to obtain images on the local host rather than re-downloading them from the image server each time one is requested.

image ID

Combination of a URI and UUID used to access Image service VM images through the image API.

image membership
A list of tenants that can access a given VM image within Image service.

image owner
The tenant who owns an Image service virtual machine image.

image registry
A list of VM images that are available through Image service.

Image service
An OpenStack core project that provides discovery, registration, and delivery services for disk and server images. The project name of the Image service is glance.

Image service API
Alternative name for the glance image API.

image status
The current status of a VM image in Image service, not to be confused with the status of a running instance.

image store
The back-end store used by Image service to store VM images, options include Object Storage, local file system, S3, or HTTP.

image UUID
UUID used by Image service to uniquely identify each VM image.

incubated project
A community project may be elevated to this status and is then promoted to a core project.

ingress filtering
The process of filtering incoming network traffic. Supported by Compute.

INI
The OpenStack configuration files use an INI format to describe options and their values. It consists of sections and key value pairs.

injection
The process of putting a file into a virtual machine image before the instance is started.

instance
A running VM, or a VM in a known state such as suspended, that can be used like a hardware server.

instance ID
Alternative term for instance UUID.

instance state
The current state of a guest VM image.

instance tunnels network
A network segment used for instance traffic tunnels between compute nodes and the network node.

instance type
Describes the parameters of the various virtual machine images that are available to

users; includes parameters such as CPU, storage, and memory. Alternative term for flavor.

instance type ID

Alternative term for a flavor ID.

instance UUID

Unique ID assigned to each guest VM instance.

interface

A physical or virtual device that provides connectivity to another device or medium.

interface ID

Unique ID for a Networking VIF or vNIC in the form of a UUID.

Internet protocol (IP)

Principal communications protocol in the internet protocol suite for relaying datagrams across network boundaries.

Internet Service Provider (ISP)

Any business that provides Internet access to individuals or businesses.

Internet Small Computer System Interface (iSCSI)

Storage protocol that encapsulates SCSI frames for transport over IP networks.

ironic

OpenStack project that provisions bare metal, as opposed to virtual machines.

IOPS

IOPS (Input/Output Operations Per Second) are a common performance measurement used to benchmark computer storage devices like hard disk drives, solid state drives, and storage area networks.

IP address

Number that is unique to every computer system on the Internet. Two versions of the Internet Protocol (IP) are in use for addresses: IPv4 and IPv6.

IP Address Management (IPAM)

The process of automating IP address allocation, deallocation, and management. Currently provided by Compute, melange, and Networking.

IPL

Initial Program Loader.

IPMI

Intelligent Platform Management Interface. IPMI is a standardized computer system interface used by system administrators for out-of-band management of computer systems and monitoring of their operation. In layman’s terms, it is a way to manage a computer using a direct network connection, whether it is turned on or not; connecting to the hardware rather than an operating system or login shell.

ip6tables

Tool used to set up, maintain, and inspect the tables of IPv6 packet filter rules in the Linux kernel. In OpenStack Compute, ip6tables is used along with arptables, ebtables, and iptables to create firewalls for both nodes and VMs.

ipset
Extension to iptables that allows creation of firewall rules that match entire “sets” of IP addresses simultaneously. These sets reside in indexed data structures to increase efficiency, particularly on systems with a large quantity of rules.

iptables
Used along with arptables and ebtables, iptables create firewalls in Compute. iptables are the tables provided by the Linux kernel firewall (implemented as different Netfilter modules) and the chains and rules it stores. Different kernel modules and programs are currently used for different protocols: iptables applies to IPv4, ip6tables to IPv6, arptables to ARP, and ebtables to Ethernet frames. Requires root privilege to manipulate.

IQN
iSCSI Qualified Name (IQN) is the format most commonly used for iSCSI names, which uniquely identify nodes in an iSCSI network. All IQNs follow the pattern iqn.yyyy-mm.domain:identifier, where ‘yyyy-mm’ is the year and month in which the domain was registered, ‘domain’ is the reversed domain name of the issuing organization, and ‘identifier’ is an optional string which makes each IQN under the same domain unique. For example, ‘iqn.2015-10.org.openstack.408ae959bce1’.

iSCSI
The SCSI disk protocol tunneled within Ethernet, supported by Compute, Object Storage, and Image service.

ISO9660
One of the VM image disk formats supported by Image service.

itsec
A default role in the Compute RBAC system that can quarantine an instance in any project.

J

Java
A programming language that is used to create systems that involve more than one computer by way of a network.

JavaScript
A scripting language that is used to build web pages.

JavaScript Object Notation (JSON)
One of the supported response formats in OpenStack.

Jenkins
Tool used to run jobs automatically for OpenStack development.

jumbo frame
Feature in modern Ethernet networks that supports frames up to approximately 9000 bytes.

Juno
The code name for the tenth release of OpenStack. The design summit took place in Atlanta, Georgia, US and Juno is an unincorporated community in Georgia.

K

Kerberos
A network authentication protocol which works on the basis of tickets. Kerberos allows nodes communication over a non-secure network, and allows nodes to prove their identity to one another in a secure manner.

kernel-based VM (KVM)
An OpenStack-supported hypervisor. KVM is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V), ARM, IBM Power, and IBM zSeries. It consists of a loadable kernel module, that provides the core virtualization infrastructure and a processor specific module.

Key Manager service
OpenStack project that produces a secret storage and generation system capable of providing key management for services wishing to enable encryption features. The code name of the project is barbican.

keystone
The project that provides OpenStack Identity services.

Kickstart
A tool to automate system configuration and installation on Red Hat, Fedora, and CentOS-based Linux distributions.

Kilo
The code name for the eleventh release of OpenStack. The design summit took place in Paris, France. Due to delays in the name selection, the release was known only as K. Because k is the unit symbol for kilo and the reference artifact is stored near Paris in the Pavillon de Breteuil in Sèvres, the community chose Kilo as the release name.

L

large object
An object within Object Storage that is larger than 5 GB.

Launchpad
The collaboration site for OpenStack.

Layer-2 network
Term used in the OSI network architecture for the data link layer. The data link layer is responsible for media access control, flow control and detecting and possibly correcting errors that may occur in the physical layer.

Layer-3 network
Term used in the OSI network architecture for the network layer. The network layer is responsible for packet forwarding including routing from one node to another.

Layer-2 (L2) agent
OpenStack Networking agent that provides layer-2 connectivity for virtual networks.

Layer-3 (L3) agent
OpenStack Networking agent that provides layer-3 (routing) services for virtual networks.

Liberty
The code name for the twelfth release of OpenStack. The design summit took place in Vancouver, Canada and Liberty is the name of a village in the Canadian province of Saskatchewan.

libvirt
Virtualization API library used by OpenStack to interact with many of its supported hypervisors.

Lightweight Directory Access Protocol (LDAP)
An application protocol for accessing and maintaining distributed directory information services over an IP network.

Linux bridge
Software that enables multiple VMs to share a single physical NIC within Compute.

Linux Bridge neutron plug-in
Enables a Linux bridge to understand a Networking port, interface attachment, and other abstractions.

Linux containers (LXC)
An OpenStack-supported hypervisor.

live migration
The ability within Compute to move running virtual machine instances from one host to another with only a small service interruption during switchover.

load balancer
A load balancer is a logical device that belongs to a cloud account. It is used to distribute workloads between multiple back-end systems or services, based on the criteria defined as part of its configuration.

load balancing
The process of spreading client requests between two or more nodes to improve performance and availability.

LBaaS
Enables Networking to distribute incoming requests evenly between designated instances.

Logical Volume Manager (LVM)
Provides a method of allocating space on mass-storage devices that is more flexible than conventional partitioning schemes.

M

magnum
Code name for the OpenStack project that provides the Containers Service.

management API
Alternative term for an admin API.

management network
A network segment used for administration, not accessible to the public Internet.

manager
Logical groupings of related code, such as the Block Storage volume manager or network manager.

manifest
Used to track segments of a large object within Object Storage.

manifest object
A special Object Storage object that contains the manifest for a large object.

manila
OpenStack project that provides shared file systems as service to applications.

manila-share
Responsible for managing Shared File System Service devices, specifically the back-end devices.

maximum transmission unit (MTU)
Maximum frame or packet size for a particular network medium. Typically 1500 bytes for Ethernet networks.

mechanism driver
A driver for the Modular Layer 2 (ML2) neutron plug-in that provides layer-2 connectivity for virtual instances. A single OpenStack installation can use multiple mechanism drivers.

melange
Project name for OpenStack Network Information Service. To be merged with Networking.

membership
The association between an Image service VM image and a tenant. Enables images to be shared with specified tenants.

membership list
A list of tenants that can access a given VM image within Image service.

memcached
A distributed memory object caching system that is used by Object Storage for caching.

memory overcommit
The ability to start new VM instances based on the actual memory usage of a host, as opposed to basing the decision on the amount of RAM each running instance thinks it

has available. Also known as RAM overcommit.

message broker
The software package used to provide AMQP messaging capabilities within Compute. Default package is RabbitMQ.

message bus
The main virtual communication line used by all AMQP messages for inter-cloud communications within Compute.

message queue
Passes requests from clients to the appropriate workers and returns the output to the client after the job completes.

Message service
OpenStack project that aims to produce an OpenStack messaging service that affords a variety of distributed application patterns in an efficient, scalable and highly-available manner, and to create and maintain associated Python libraries and documentation. The code name for the project is zaqar.

Metadata agent
OpenStack Networking agent that provides metadata services for instances.

Meta-Data Server (MDS)
Stores CephFS metadata.

migration
The process of moving a VM instance from one host to another.

mistral
OpenStack project that provides the Workflow service.

Mitaka
The code name for the thirteenth release of OpenStack. The design summit took place in Tokyo, Japan. Mitaka is a city in Tokyo.

monasca
OpenStack project that provides a Monitoring service.

multi-host
High-availability mode for legacy (nova) networking. Each compute node handles NAT and DHCP and acts as a gateway for all of the VMs on it. A networking failure on one compute node doesn’t affect VMs on other compute nodes.

multinic
Facility in Compute that allows each virtual machine instance to have more than one VIF connected to it.

murano
OpenStack project that provides an Application catalog.

Modular Layer 2 (ML2) neutron plug-in
Can concurrently use multiple layer-2 networking technologies, such as 802.1Q and VXLAN, in Networking.

Monitor (LBaaS)
LBaaS feature that provides availability monitoring using the `ping` command, TCP, and

HTTP/HTTPS GET.

Monitor (Mon)

A Ceph component that communicates with external clients, checks data state and consistency, and performs quorum functions.

Monitoring

The OpenStack project that provides a multi-tenant, highly scalable, performant, fault-tolerant Monitoring-as-a-Service solution for metrics, complex event processing, and logging. It builds an extensible platform for advanced monitoring services that can be used by both operators and tenants to gain operational insight and visibility, ensuring availability and stability. The project name is monasca.

multi-factor authentication

Authentication method that uses two or more credentials, such as a password and a private key. Currently not supported in Identity.

MultiNic

Facility in Compute that enables a virtual machine instance to have more than one VIF connected to it.

N

Nebula

Released as open source by NASA in 2010 and is the basis for Compute.

netadmin

One of the default roles in the Compute RBAC system. Enables the user to allocate publicly accessible IP addresses to instances and change firewall rules.

NetApp volume driver

Enables Compute to communicate with NetApp storage devices through the NetApp OnCommand Provisioning Manager.

network

A virtual network that provides connectivity between entities. For example, a collection of virtual ports that share network connectivity. In Networking terminology, a network is always a layer-2 network.

NAT

Network Address Translation; Process of modifying IP address information while in transit. Supported by Compute and Networking.

network controller

A Compute daemon that orchestrates the network configuration of nodes, including IP addresses, VLANs, and bridging. Also manages routing for both public and private networks.

Network File System (NFS)

A method for making file systems available over the network. Supported by OpenStack.

network ID
Unique ID assigned to each network segment within Networking. Same as network UUID.

network manager
The Compute component that manages various network components, such as firewall rules, IP address allocation, and so on.

network namespace
Linux kernel feature that provides independent virtual networking instances on a single host with separate routing tables and interfaces. Similar to virtual routing and forwarding (VRF) services on physical network equipment.

network node
Any compute node that runs the network worker daemon.

network segment
Represents a virtual, isolated OSI layer-2 subnet in Networking.

Newton
The code name for the fourteenth release of OpenStack. The design summit took place in Austin, Texas, US. The release is named after “Newton House” which is located at 1013 E. Ninth St., Austin, TX. which is listed on the National Register of Historic Places.

NTP
Network Time Protocol; Method of keeping a clock for a host or node correct via communication with a trusted, accurate time source.

network UUID
Unique ID for a Networking network segment.

network worker
The `nova-network` worker daemon; provides services such as giving an IP address to a booting nova instance.

Networking service
A core OpenStack project that provides a network connectivity abstraction layer to OpenStack Compute. The project name of Networking is neutron.

Networking API
API used to access OpenStack Networking. Provides an extensible architecture to enable custom plug-in creation.

neutron
A core OpenStack project that provides a network connectivity abstraction layer to OpenStack Compute.

neutron API
An alternative name for Networking API.

neutron manager
Enables Compute and Networking integration, which enables Networking to perform network management for guest VMs.

neutron plug-in
Interface within Networking that enables organizations to create custom plug-ins for

advanced features, such as QoS, ACLs, or IDS.

Nexenta volume driver
Provides support for NexentaStor devices in Compute.

No ACK
Disables server-side message acknowledgment in the Compute RabbitMQ. Increases performance but decreases reliability.

node
A VM instance that runs on a host.

non-durable exchange
Message exchange that is cleared when the service restarts. Its data is not written to persistent storage.

non-durable queue
Message queue that is cleared when the service restarts. Its data is not written to persistent storage.

non-persistent volume
Alternative term for an ephemeral volume.

north-south traffic
Network traffic between a user or client (north) and a server (south), or traffic into the cloud (south) and out of the cloud (north). See also east-west traffic.

nova
OpenStack project that provides compute services.

Nova API
Alternative term for the Compute API.

nova-network
A Compute component that manages IP address allocation, firewalls, and other network-related tasks. This is the legacy networking option and an alternative to Networking.

O

object
A BLOB of data held by Object Storage; can be in any format.

object auditor
Opens all objects for an object server and verifies the MD5 hash, size, and metadata for each object.

object expiration
A configurable option within Object Storage to automatically delete objects after a specified amount of time has passed or a certain date is reached.

object hash
Uniquely ID for an Object Storage object.

object path hash
Used by Object Storage to determine the location of an object in the ring. Maps objects to partitions.

object replicator
An Object Storage component that copies an object to remote partitions for fault tolerance.

object server
An Object Storage component that is responsible for managing objects.

Object Storage service
The OpenStack core project that provides eventually consistent and redundant storage and retrieval of fixed digital content. The project name of OpenStack Object Storage is swift.

Object Storage API
API used to access OpenStack Object Storage.

Object Storage Device (OSD)
The Ceph storage daemon.

object versioning
Allows a user to set a flag on an Object Storage container so that all objects within the container are versioned.

Ocata
The code name for the fifteenth release of OpenStack. The design summit will take place in Barcelona, Spain. Ocata is a beach north of Barcelona.

Oldie
Term for an Object Storage process that runs for a long time. Can indicate a hung process.

Open Cloud Computing Interface (OCCI)
A standardized interface for managing compute, data, and network resources, currently unsupported in OpenStack.

Open Virtualization Format (OVF)
Standard for packaging VM images. Supported in OpenStack.

Open vSwitch
Open vSwitch is a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license. It is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols (for example NetFlow, sFlow, SPAN, RSPAN, CLI, LACP, 802.1ag).

Open vSwitch (OVS) agent
Provides an interface to the underlying Open vSwitch service for the Networking plug-in.

Open vSwitch neutron plug-in
Provides support for Open vSwitch in Networking.

OpenLDAP
An open source LDAP server. Supported by both Compute and Identity.

OpenStack
OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a data center, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface. OpenStack is an open source project licensed under the Apache License 2.0.

OpenStack code name
Each OpenStack release has a code name. Code names ascend in alphabetical order: Austin, Bexar, Cactus, Diablo, Essex, Folsom, Grizzly, Havana, Icehouse, Juno, Kilo, Liberty, and Mitaka. Code names are cities or counties near where the corresponding OpenStack design summit took place. An exception, called the Waldon exception, is granted to elements of the state flag that sound especially cool. Code names are chosen by popular vote.

openSUSE
A Linux distribution that is compatible with OpenStack.

operator
The person responsible for planning and maintaining an OpenStack installation.

optional service
An official OpenStack service defined as optional by DefCore Committee. Currently, consists of Dashboard (horizon), Telemetry service (Telemetry), Orchestration service (heat), Database service (trove), Bare Metal service (ironic), and so on.

Orchestration service
An integrated project that orchestrates multiple cloud applications for OpenStack. The project name of Orchestration is heat.

orphan
In the context of Object Storage, this is a process that is not terminated after an upgrade, restart, or reload of the service.

Oslo
OpenStack project that produces a set of Python libraries containing code shared by OpenStack projects.

P

parent cell
If a requested resource, such as CPU time, disk storage, or memory, is not available in the parent cell, the request is forwarded to associated child cells.

partition
A unit of storage within Object Storage used to store objects. It exists on top of devices and is replicated for fault tolerance.

partition index
Contains the locations of all Object Storage partitions within the ring.

partition shift value
Used by Object Storage to determine which partition data should reside on.

path MTU discovery (PMTUD)
Mechanism in IP networks to detect end-to-end MTU and adjust packet size accordingly.

pause
A VM state where no changes occur (no changes in memory, network communications stop, etc); the VM is frozen but not shut down.

PCI passthrough
Gives guest VMs exclusive access to a PCI device. Currently supported in OpenStack Havana and later releases.

persistent message
A message that is stored both in memory and on disk. The message is not lost after a failure or restart.

persistent volume
Changes to these types of disk volumes are saved.

personality file
A file used to customize a Compute instance. It can be used to inject SSH keys or a specific network configuration.

Platform-as-a-Service (PaaS)
Provides to the consumer the ability to deploy applications through a programming language or tools supported by the cloud platform provider. An example of Platform-as-a-Service is an Eclipse/Java programming platform provided with no downloads required.

plug-in
Software component providing the actual implementation for Networking APIs, or for Compute APIs, depending on the context.

policy service
Component of Identity that provides a rule-management interface and a rule-based authorization engine.

pool
A logical set of devices, such as web servers, that you group together to receive and process traffic. The load balancing function chooses which member of the pool handles the new requests or connections received on the VIP address. Each VIP has one pool.

pool member
An application that runs on the back-end server in a load-balancing system.

port
A virtual network port within Networking; VIFs / vNICs are connected to a port.

port UUID
Unique ID for a Networking port.

preseed
A tool to automate system configuration and installation on Debian-based Linux distributions.

private image
An Image service VM image that is only available to specified tenants.

private IP address
An IP address used for management and administration, not available to the public Internet.

private network
The Network Controller provides virtual networks to enable compute servers to interact with each other and with the public network. All machines must have a public and private network interface. A private network interface can be a flat or VLAN network interface. A flat network interface is controlled by the flat_interface with flat managers. A VLAN network interface is controlled by the `vlan_interface` option with VLAN managers.

project
Projects represent the base unit of “ownership” in OpenStack, in that all resources in OpenStack should be owned by a specific project. In OpenStack Identity, a project must be owned by a specific domain.

project ID
User-defined alphanumeric string in Compute; the name of a project.

project VPN
Alternative term for a cloudpipe.

promiscuous mode
Causes the network interface to pass all traffic it receives to the host rather than passing only the frames addressed to it.

protected property
Generally, extra properties on an Image service image to which only cloud administrators have access. Limits which user roles can perform CRUD operations on that property. The cloud administrator can configure any image property as protected.

provider
An administrator who has access to all hosts and instances.

proxy node
A node that provides the Object Storage proxy service.

proxy server
Users of Object Storage interact with the service through the proxy server, which in turn looks up the location of the requested data within the ring and returns the results to the user.

public API
An API endpoint used for both service-to-service communication and end-user interactions.

public image
An Image service VM image that is available to all tenants.

public IP address
An IP address that is accessible to end-users.

public key authentication
Authentication method that uses keys rather than passwords.

public network
The Network Controller provides virtual networks to enable compute servers to interact with each other and with the public network. All machines must have a public and private network interface. The public network interface is controlled by the `public_interface` option.

Puppet
An operating system configuration-management tool supported by OpenStack.

Python
Programming language used extensively in OpenStack.

Q

QEMU Copy On Write 2 (QCOW2)
One of the VM image disk formats supported by Image service.

Qpid
Message queue software supported by OpenStack; an alternative to RabbitMQ.

Quality of Service (QoS)
The ability to guarantee certain network or storage requirements to satisfy a Service Level Agreement (SLA) between an application provider and end users. Typically includes performance requirements like networking bandwidth, latency, jitter correction, and reliability as well as storage performance in Input/Output Operations Per Second (IOPS), throttling agreements, and performance expectations at peak load.

quarantine
If Object Storage finds objects, containers, or accounts that are corrupt, they are placed in this state, are not replicated, cannot be read by clients, and a correct copy is re-replicated.

Quick EMUlator (QEMU)
QEMU is a generic and open source machine emulator and virtualizer. One of the hypervisors supported by OpenStack, generally used for development purposes.

quota
In Compute and Block Storage, the ability to set resource limits on a per-project basis.

R

RabbitMQ
The default message queue software used by OpenStack.

Rackspace Cloud Files
Released as open source by Rackspace in 2010; the basis for Object Storage.

RADOS Block Device (RBD)
Ceph component that enables a Linux block device to be striped over multiple distributed data stores.

radvd
The router advertisement daemon, used by the Compute VLAN manager and FlatDHCP manager to provide routing services for VM instances.

rally
OpenStack project that provides the Benchmark service.

RAM filter
The Compute setting that enables or disables RAM overcommitment.

RAM overcommit
The ability to start new VM instances based on the actual memory usage of a host, as opposed to basing the decision on the amount of RAM each running instance thinks it has available. Also known as memory overcommit.

rate limit
Configurable option within Object Storage to limit database writes on a per-account and/or per-container basis.

raw
One of the VM image disk formats supported by Image service; an unstructured disk image.

rebalance
The process of distributing Object Storage partitions across all drives in the ring; used during initial ring creation and after ring reconfiguration.

reboot
Either a soft or hard reboot of a server. With a soft reboot, the operating system is signaled to restart, which enables a graceful shutdown of all processes. A hard reboot is the equivalent of power cycling the server. The virtualization platform should ensure that the reboot action has completed successfully, even in cases in which the underlying domain/VM is paused or halted/stopped.

rebuild
Removes all data on the server and replaces it with the specified image. Server ID and IP addresses remain the same.

Recon
An Object Storage component that collects meters.

record
Belongs to a particular domain and is used to specify information about the domain. There are several types of DNS records. Each record type contains particular information used to describe the purpose of that record. Examples include mail exchange (MX) records, which specify the mail server for a particular domain; and name server (NS) records, which specify the authoritative name servers for a domain.

record ID
A number within a database that is incremented each time a change is made. Used by Object Storage when replicating.

Red Hat Enterprise Linux (RHEL)
A Linux distribution that is compatible with OpenStack.

reference architecture
A recommended architecture for an OpenStack cloud.

region
A discrete OpenStack environment with dedicated API endpoints that typically shares only the Identity (keystone) with other regions.

registry
Alternative term for the Image service registry.

registry server
An Image service that provides VM image metadata information to clients.

Reliable, Autonomic Distributed Object Store (RADOS)
A collection of components that provides object storage within Ceph. Similar to OpenStack Object Storage.

Remote Procedure Call (RPC)
The method used by the Compute RabbitMQ for intra-service communications.

replica
Provides data redundancy and fault tolerance by creating copies of Object Storage objects, accounts, and containers so that they are not lost when the underlying storage fails.

replica count
The number of replicas of the data in an Object Storage ring.

replication
The process of copying data to a separate physical device for fault tolerance and performance.

replicator
The Object Storage back-end process that creates and manages object replicas.

request ID
Unique ID assigned to each request sent to Compute.

rescue image
A special type of VM image that is booted when an instance is placed into rescue mode. Allows an administrator to mount the file systems for an instance to correct the problem.

resize
Converts an existing server to a different flavor, which scales the server up or down. The original server is saved to enable rollback if a problem occurs. All resizes must be tested and explicitly confirmed, at which time the original server is removed.

RESTful
A kind of web service API that uses REST, or Representational State Transfer. REST is the style of architecture for hypermedia systems that is used for the World Wide Web.

ring
An entity that maps Object Storage data to partitions. A separate ring exists for each service, such as account, object, and container.

ring builder
Builds and manages rings within Object Storage, assigns partitions to devices, and pushes the configuration to other storage nodes.

Role Based Access Control (RBAC)
Provides a predefined list of actions that the user can perform, such as start or stop VMs, reset passwords, and so on. Supported in both Identity and Compute and can be configured using the horizon dashboard.

role
A personality that a user assumes to perform a specific set of operations. A role includes a set of rights and privileges. A user assuming that role inherits those rights and privileges.

role ID
Alphanumeric ID assigned to each Identity service role.

rootwrap
A feature of Compute that allows the unprivileged “nova” user to run a specified list of commands as the Linux root user.

round-robin scheduler
Type of Compute scheduler that evenly distributes instances among available hosts.

router
A physical or virtual network device that passes network traffic between different networks.

routing key
The Compute direct exchanges, fanout exchanges, and topic exchanges use this key to determine how to process a message; processing varies depending on exchange type.

RPC driver
Modular system that allows the underlying message queue software of Compute to be changed. For example, from RabbitMQ to ZeroMQ or Qpid.

rsync
Used by Object Storage to push object replicas.

RXTX cap
Absolute limit on the amount of network traffic a Compute VM instance can send and receive.

RXTX quota
Soft limit on the amount of network traffic a Compute VM instance can send and receive.

S

S3
Object storage service by Amazon; similar in function to Object Storage, it can act as a back-end store for Image service VM images.

sahara
OpenStack project that provides a scalable data-processing stack and associated management interfaces.

SAML assertion
Contains information about a user as provided by the identity provider. It is an indication that a user has been authenticated.

scheduler manager
A Compute component that determines where VM instances should start. Uses modular design to support a variety of scheduler types.

scoped token
An Identity service API access token that is associated with a specific tenant.

scrubber
Checks for and deletes unused VMs; the component of Image service that implements delayed delete.

secret key
String of text known only by the user; used along with an access key to make requests to the Compute API.

secure boot
Process whereby the system firmware validates the authenticity of the code involved in the boot process.

secure shell (SSH)
Open source tool used to access remote hosts through an encrypted communications channel, SSH key injection is supported by Compute.

security group
A set of network traffic filtering rules that are applied to a Compute instance.

segmented object
An Object Storage large object that has been broken up into pieces. The re-assembled object is called a concatenated object.

self-service
For IaaS, ability for a regular (non-privileged) account to manage a virtual infrastructure component such as networks without involving an administrator.

SELinux
Linux kernel security module that provides the mechanism for supporting access control policies.

senlin
OpenStack project that provides a Clustering service.

server
Computer that provides explicit services to the client software running on that system, often managing a variety of computer operations. A server is a VM instance in the Compute system. Flavor and image are requisite elements when creating a server.

server image
Alternative term for a VM image.

server UUID
Unique ID assigned to each guest VM instance.

service
An OpenStack service, such as Compute, Object Storage, or Image service. Provides one or more endpoints through which users can access resources and perform operations.

service catalog
Alternative term for the Identity service catalog.

service ID
Unique ID assigned to each service that is available in the Identity service catalog.

service provider
A system that provides services to other system entities. In case of federated identity, OpenStack Identity is the service provider.

service registration
An Identity service feature that enables services, such as Compute, to automatically register with the catalog.

service tenant
Special tenant that contains all services that are listed in the catalog.

service token
An administrator-defined token used by Compute to communicate securely with the Identity service.

session back end
The method of storage used by horizon to track client sessions, such as local memory, cookies, a database, or memcached.

session persistence
A feature of the load-balancing service. It attempts to force subsequent connections to a service to be redirected to the same node as long as it is online.

session storage
A horizon component that stores and tracks client session information. Implemented through the Django sessions framework.

share

A remote, mountable file system in the context of the Shared File Systems. You can mount a share to, and access a share from, several hosts by several users at a time.

share network

An entity in the context of the Shared File Systems that encapsulates interaction with the Networking service. If the driver you selected runs in the mode requiring such kind of interaction, you need to specify the share network to create a share.

Shared File Systems API

A Shared File Systems service that provides a stable RESTful API. The service authenticates and routes requests throughout the Shared File Systems service. There is python-manilaclient to interact with the API.

Shared File Systems service

An OpenStack service that provides a set of services for management of shared file systems in a multi-tenant cloud environment. The service is similar to how OpenStack provides block-based storage management through the OpenStack Block Storage service project. With the Shared File Systems service, you can create a remote file system and mount the file system on your instances. You can also read and write data from your instances to and from your file system. The project name of the Shared File Systems service is manila.

shared IP address

An IP address that can be assigned to a VM instance within the shared IP group. Public IP addresses can be shared across multiple servers for use in various high-availability scenarios. When an IP address is shared to another server, the cloud network restrictions are modified to enable each server to listen to and respond on that IP address. You can optionally specify that the target server network configuration be modified. Shared IP addresses can be used with many standard heartbeat facilities, such as keepalive, that monitor for failure and manage IP failover.

shared IP group

A collection of servers that can share IPs with other members of the group. Any server in a group can share one or more public IPs with any other server in the group. With the exception of the first server in a shared IP group, servers must be launched into shared IP groups. A server may be a member of only one shared IP group.

shared storage

Block storage that is simultaneously accessible by multiple clients, for example, NFS.

Sheepdog

Distributed block storage system for QEMU, supported by OpenStack.

Simple Cloud Identity Management (SCIM)

Specification for managing identity in the cloud, currently unsupported by OpenStack.

Single-root I/O Virtualization (SR-IOV)

A specification that, when implemented by a physical PCIe device, enables it to appear as multiple separate PCIe devices. This enables multiple virtualized guests to share direct access to the physical device, offering improved performance over an

equivalent virtual device. Currently supported in OpenStack Havana and later releases.

Service Level Agreement (SLA)

Contractual obligations that ensure the availability of a service.

SmokeStack

Runs automated tests against the core OpenStack API; written in Rails.

snapshot

A point-in-time copy of an OpenStack storage volume or image. Use storage volume snapshots to back up volumes. Use image snapshots to back up data, or as “gold” images for additional servers.

soft reboot

A controlled reboot where a VM instance is properly restarted through operating system commands.

Software Development Lifecycle Automation service

OpenStack project that aims to make cloud services easier to consume and integrate with application development process by automating the source-to-image process, and simplifying app-centric deployment. The project name is solum.

SolidFire Volume Driver

The Block Storage driver for the SolidFire iSCSI storage appliance.

solum

OpenStack project that provides a Software Development Lifecycle Automation service.

SPICE

The Simple Protocol for Independent Computing Environments (SPICE) provides remote desktop access to guest virtual machines. It is an alternative to VNC. SPICE is supported by OpenStack.

spread-first scheduler

The Compute VM scheduling algorithm that attempts to start a new VM on the host with the least amount of load.

SQL-Alchemy

An open source SQL toolkit for Python, used in OpenStack.

SQLite

A lightweight SQL database, used as the default persistent storage method in many OpenStack services.

stack

A set of OpenStack resources created and managed by the Orchestration service according to a given template (either an AWS CloudFormation template or a Heat Orchestration Template (HOT)).

StackTach

Community project that captures Compute AMQP communications; useful for debugging.

static IP address
Alternative term for a fixed IP address.

StaticWeb
WSGI middleware component of Object Storage that serves container data as a static web page.

storage back end
The method that a service uses for persistent storage, such as iSCSI, NFS, or local disk.

storage node
An Object Storage node that provides container services, account services, and object services; controls the account databases, container databases, and object storage.

storage manager
A XenAPI component that provides a pluggable interface to support a wide variety of persistent storage back ends.

storage manager back end
A persistent storage method supported by XenAPI, such as iSCSI or NFS.

storage services
Collective name for the Object Storage object services, container services, and account services.

strategy
Specifies the authentication source used by Image service or Identity. In the Database service, it refers to the extensions implemented for a data store.

subdomain
A domain within a parent domain. Subdomains cannot be registered. Subdomains enable you to delegate domains. Subdomains can themselves have subdomains, so third-level, fourth-level, fifth-level, and deeper levels of nesting are possible.

subnet
Logical subdivision of an IP network.

SUSE Linux Enterprise Server (SLES)
A Linux distribution that is compatible with OpenStack.

suspend
Alternative term for a paused VM instance.

swap
Disk-based virtual memory used by operating systems to provide more memory than is actually available on the system.

swauth
An authentication and authorization service for Object Storage, implemented through WSGI middleware; uses Object Storage itself as the persistent backing store.

swift
An OpenStack core project that provides object storage services.

swift All in One (SAIO)
Creates a full Object Storage development environment within a single VM.

swift middleware
Collective term for Object Storage components that provide additional functionality.

swift proxy server
Acts as the gatekeeper to Object Storage and is responsible for authenticating the user.

swift storage node
A node that runs Object Storage account, container, and object services.

sync point
Point in time since the last container and accounts database sync among nodes within Object Storage.

sysadmin
One of the default roles in the Compute RBAC system. Enables a user to add other users to a project, interact with VM images that are associated with the project, and start and stop VM instances.

system usage
A Compute component that, along with the notification system, collects meters and usage information. This information can be used for billing.

T

Telemetry service
An integrated project that provides metering and measuring facilities for OpenStack. The project name of Telemetry is ceilometer.

TempAuth
An authentication facility within Object Storage that enables Object Storage itself to perform authentication and authorization. Frequently used in testing and development.

Tempest
Automated software test suite designed to run against the trunk of the OpenStack core project.

TempURL
An Object Storage middleware component that enables creation of URLs for temporary object access.

tenant
A group of users; used to isolate access to Compute resources. An alternative term for a project.

Tenant API
An API that is accessible to tenants.

tenant endpoint
An Identity service API endpoint that is associated with one or more tenants.

tenant ID
Unique ID assigned to each tenant within the Identity service. The project IDs map to the tenant IDs.

token
An alpha-numeric string of text used to access OpenStack APIs and resources.

token services
An Identity service component that manages and validates tokens after a user or tenant has been authenticated.

tombstone
Used to mark Object Storage objects that have been deleted; ensures that the object is not updated on another node after it has been deleted.

topic publisher
A process that is created when a RPC call is executed; used to push the message to the topic exchange.

Torpedo
Community project used to run automated tests against the OpenStack API.

transaction ID
Unique ID assigned to each Object Storage request; used for debugging and tracing.

transient
Alternative term for non-durable.

transient exchange
Alternative term for a non-durable exchange.

transient message
A message that is stored in memory and is lost after the server is restarted.

transient queue
Alternative term for a non-durable queue.

TripleO
OpenStack-on-OpenStack program. The code name for the OpenStack Deployment program.

trove
OpenStack project that provides database services to applications.

trusted platform module (TPM)
Specialized microprocessor for incorporating cryptographic keys into devices for authenticating and securing a hardware platform.

U

Ubuntu
A Debian-based Linux distribution.

unscoped token
Alternative term for an Identity service default token.

updater
Collective term for a group of Object Storage components that processes queued and failed updates for containers and objects.

user
In OpenStack Identity, entities represent individual API consumers and are owned by a specific domain. In OpenStack Compute, a user can be associated with roles, projects, or both.

user data
A blob of data that the user can specify when they launch an instance. The instance can access this data through the metadata service or config drive. Commonly used to pass a shell script that the instance runs on boot.

User Mode Linux (UML)
An OpenStack-supported hypervisor.

V

VIF UUID
Unique ID assigned to each Networking VIF.

VIP
The primary load balancing configuration object. Specifies the virtual IP address and port where client traffic is received. Also defines other details such as the load balancing method to be used, protocol, and so on. This entity is sometimes known in load-balancing products as a virtual server, vserver, or listener.

Virtual Central Processing Unit (vCPU)
Subdivides physical CPUs. Instances can then use those divisions.

Virtual Disk Image (VDI)
One of the VM image disk formats supported by Image service.

VXLAN
A network virtualization technology that attempts to reduce the scalability problems associated with large cloud computing deployments. It uses a VLAN-like encapsulation technique to encapsulate Ethernet frames within UDP packets.

Virtual Hard Disk (VHD)
One of the VM image disk formats supported by Image service.

virtual IP
An Internet Protocol (IP) address configured on the load balancer for use by clients connecting to a service that is load balanced. Incoming connections are distributed to back-end nodes based on the configuration of the load balancer.

virtual machine (VM)
An operating system instance that runs on top of a hypervisor. Multiple VMs can run at the same time on the same physical host.

virtual network

An L2 network segment within Networking.

virtual networking

A generic term for virtualization of network functions such as switching, routing, load balancing, and security using a combination of VMs and overlays on physical network infrastructure.

Virtual Network Computing (VNC)

Open source GUI and CLI tools used for remote console access to VMs. Supported by Compute.

Virtual Network InterFace (VIF)

An interface that is plugged into a port in a Networking network. Typically a virtual network interface belonging to a VM.

virtual port

Attachment point where a virtual interface connects to a virtual network.

virtual private network (VPN)

Provided by Compute in the form of cloudpipes, specialized instances that are used to create VPNs on a per-project basis.

virtual server

Alternative term for a VM or guest.

virtual switch (vSwitch)

Software that runs on a host or node and provides the features and functions of a hardware-based network switch.

virtual VLAN

Alternative term for a virtual network.

VirtualBox

An OpenStack-supported hypervisor.

VLAN manager

A Compute component that provides dnsmasq and radvd and sets up forwarding to and from cloudpipe instances.

VLAN network

The Network Controller provides virtual networks to enable compute servers to interact with each other and with the public network. All machines must have a public and private network interface. A VLAN network is a private network interface, which is controlled by the `vlan_interface` option with VLAN managers.

VM disk (VMDK)

One of the VM image disk formats supported by Image service.

VM image

Alternative term for an image.

VM Remote Control (VMRC)

Method to access VM instance consoles using a web browser. Supported by Compute.

VMware API

Supports interaction with VMware products in Compute.

VMware NSX Neutron plug-in
Provides support for VMware NSX in Neutron.

VNC proxy
A Compute component that provides users access to the consoles of their VM instances through VNC or VMRC.

volume
Disk-based data storage generally represented as an iSCSI target with a file system that supports extended attributes; can be persistent or ephemeral.

Volume API
Alternative name for the Block Storage API.

volume controller
A Block Storage component that oversees and coordinates storage volume actions.

volume driver
Alternative term for a volume plug-in.

volume ID
Unique ID applied to each storage volume under the Block Storage control.

volume manager
A Block Storage component that creates, attaches, and detaches persistent storage volumes.

volume node
A Block Storage node that runs the cinder-volume daemon.

volume plug-in
Provides support for new and specialized types of back-end storage for the Block Storage volume manager.

volume worker
A cinder component that interacts with back-end storage to manage the creation and deletion of volumes and the creation of compute volumes, provided by the cinder-volume daemon.

vSphere
An OpenStack-supported hypervisor.

W

weighting
A Compute process that determines the suitability of the VM instances for a job for a particular host. For example, not enough RAM on the host, too many CPUs on the host, and so on.

weight
Used by Object Storage devices to determine which storage devices are suitable for the job. Devices are weighted by size.

weighted cost
The sum of each cost used when deciding where to start a new VM instance in Compute.

worker
A daemon that listens to a queue and carries out tasks in response to messages. For example, the cinder-volume worker manages volume creation and deletion on storage arrays.

Workflow service
OpenStack project that provides a simple YAML-based language to write workflows, tasks and transition rules, and a service that allows to upload them, modify, run them at scale and in a highly available manner, manage and monitor workflow execution state and state of individual tasks. The code name of the project is mistral.

X

Xen
Xen is a hypervisor using a microkernel design, providing services that allow multiple computer operating systems to execute on the same computer hardware concurrently.

Xen API
The Xen administrative API, which is supported by Compute.

Xen Cloud Platform (XCP)
An OpenStack-supported hypervisor.

Xen Storage Manager Volume Driver
A Block Storage volume plug-in that enables communication with the Xen Storage Manager API.

XenServer
An OpenStack-supported hypervisor.

XFS
High-performance 64-bit file system created by Silicon Graphics. Excels in parallel I/O operations and data consistency.

Z

zaqar
OpenStack project that provides a message service to applications.

ZeroMQ
Message queue software supported by OpenStack. An alternative to RabbitMQ. Also spelled 0MQ.

Zuul
Tool used in OpenStack development to ensure correctly ordered testing of changes in parallel.