openstack.

High Availability Guide

Release Version: 15.0.0

OpenStack contributors

Jun 24, 2017

CONTENTS

ABSTRACT

This guide describes how to install and configure OpenStack for high availability. It supplements the Installation Tutorials and Guides and assumes that you are familiar with the material in those guides.

Note: This guide documents OpenStack Ocata, Newton, and Mitaka releases and may not apply to EOL releases Kilo and Liberty.

Warning: This guide is a work-in-progress and changing rapidly while we continue to test and enhance the guidance. There are open *TODO* items throughout and available on the OpenStack manuals bug list. Please help where you are able.

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.

Introduction to OpenStack high availability

High availability systems seek to minimize the following issues:

1. System downtime: Occurs when a user-facing service is unavailable beyond a specified maximum amount of time.
2. Data loss: Accidental deletion or destruction of data.

Most high availability systems guarantee protection against system downtime and data loss only in the event of a single failure. However, they are also expected to protect against cascading failures, where a single failure deteriorates into a series of consequential failures. Many service providers guarantee a *Service Level Agreement (SLA)* including uptime percentage of computing service, which is calculated based on the available time and system downtime excluding planned outage time.

Redundancy and failover

High availability is implemented with redundant hardware running redundant instances of each service. If one piece of hardware running one instance of a service fails, the system can then failover to use another instance of a service that is running on hardware that did not fail.

A crucial aspect of high availability is the elimination of single points of failure (SPOFs). A SPOF is an individual piece of equipment or software that causes system downtime or data loss if it fails. In order to eliminate SPOFs, check that mechanisms exist for redundancy of:

- Network components, such as switches and routers
- Applications and automatic service migration
- Storage components
- Facility services such as power, air conditioning, and fire protection

In the event that a component fails and a back-up system must take on its load, most high availability systems will replace the failed component as quickly as possible to maintain necessary redundancy. This way time spent in a degraded protection state is minimized.

Most high availability systems fail in the event of multiple independent (non-consequential) failures. In this case, most implementations favor protecting data over maintaining availability.

High availability systems typically achieve an uptime percentage of 99.99% or more, which roughly equates to less than an hour of cumulative downtime per year. In order to achieve this, high availability systems should keep recovery times after a failure to about one to two minutes, sometimes significantly less.

OpenStack currently meets such availability requirements for its own infrastructure services, meaning that an uptime of 99.99% is feasible for the OpenStack infrastructure proper. However, OpenStack does not guarantee 99.99% availability for individual guest instances.

This document discusses some common methods of implementing highly available systems, with an emphasis on the core OpenStack services and other open source services that are closely aligned with OpenStack.

You will need to address high availability concerns for any applications software that you run on your OpenStack environment. The important thing is to make sure that your services are redundant and available. How you achieve that is up to you.

Stateless versus stateful services

The following are the definitions of stateless and stateful services:

Stateless service A service that provides a response after your request and then requires no further attention. To make a stateless service highly available, you need to provide redundant instances and load balance them. OpenStack services that are stateless include `nova-api`, `nova-conductor`, `glance-api`, `keystone-api`, `neutron-api`, and `nova-scheduler`.

Stateful service A service where subsequent requests to the service depend on the results of the first request. Stateful services are more difficult to manage because a single action typically involves more than one request. Providing additional instances and load balancing does not solve the problem. For example, if the horizon user interface reset itself every time you went to a new page, it would not be very useful. OpenStack services that are stateful include the OpenStack database and message queue. Making stateful services highly available can depend on whether you choose an active/passive or active/active configuration.

Active/passive versus active/active

Stateful services can be configured as active/passive or active/active, which are defined as follows:

active/passive configuration Maintains a redundant instance that can be brought online when the active service fails. For example, OpenStack writes to the main database while maintaining a disaster recovery database that can be brought online if the main database fails.

A typical active/passive installation for a stateful service maintains a replacement resource that can be brought online when required. Requests are handled using a *virtual IP address (VIP)* that facilitates returning to service with minimal reconfiguration. A separate application (such as Pacemaker or Corosync) monitors these services, bringing the backup online as necessary.

active/active configuration Each service also has a backup but manages both the main and redundant systems concurrently. This way, if there is a failure, the user is unlikely to notice. The backup system is already online and takes on increased load while the main system is fixed and brought back online.

Typically, an active/active installation for a stateless service maintains a redundant instance, and requests are load balanced using a virtual IP address and a load balancer such as HAProxy.

A typical active/active installation for a stateful service includes redundant services, with all instances having an identical state. In other words, updates to one instance of a database update all other instances. This way a request to one instance is the same as a request to any other. A load balancer manages the traffic to these systems, ensuring that operational systems always handle the request.

Clusters and quorums

The quorum specifies the minimal number of nodes that must be functional in a cluster of redundant nodes in order for the cluster to remain functional. When one node fails and failover transfers control to other nodes, the system must ensure that data and processes remain sane. To determine this, the contents of the remaining nodes are compared and, if there are discrepancies, a majority rules algorithm is implemented.

For this reason, each cluster in a high availability environment should have an odd number of nodes and the quorum is defined as more than a half of the nodes. If multiple nodes fail so that the cluster size falls below the quorum value, the cluster itself fails.

For example, in a seven-node cluster, the quorum should be set to `floor(7/2) + 1 == 4`. If quorum is four and four nodes fail simultaneously, the cluster itself would fail, whereas it would continue to function, if no more than three nodes fail. If split to partitions of three and four nodes respectively, the quorum of four nodes would continue to operate the majority partition and stop or fence the minority one (depending on the no-quorum-policy cluster configuration).

And the quorum could also have been set to three, just as a configuration example.

Note: We do not recommend setting the quorum to a value less than `floor(n/2) + 1` as it would likely cause a split-brain in a face of network partitions.

When four nodes fail simultaneously, the cluster would continue to function as well. But if split to partitions of three and four nodes respectively, the quorum of three would have made both sides to attempt to fence the other and host resources. Without fencing enabled, it would go straight to running two copies of each resource.

This is why setting the quorum to a value less than `floor(n/2) + 1` is dangerous. However it may be required for some specific cases, such as a temporary measure at a point it is known with 100% certainty that the other nodes are down.

When configuring an OpenStack environment for study or demonstration purposes, it is possible to turn off the quorum checking. Production systems should always run with quorum enabled.

Single-controller high availability mode

OpenStack supports a single-controller high availability mode that is managed by the services that manage highly available environments but is not actually highly available because no redundant controllers are configured to use for failover. This environment can be used for study and demonstration but is not appropriate for a production environment.

It is possible to add controllers to such an environment to convert it into a truly highly available environment.

High availability is not for every user. It presents some challenges. High availability may be too complex for databases or systems with large amounts of data. Replication can slow large systems down. Different setups have different prerequisites. Read the guidelines for each setup.

Important: High availability is turned off as the default in OpenStack setups.

Configuring the basic environment

This chapter describes the basic environment for high availability, such as hardware, operating system, common services.

Hardware considerations for high availability

When you use high availability, consider the hardware requirements needed for your application.

Hardware setup

The following are the standard hardware requirements:

- Provider networks: See the *Overview -> Networking Option 1: Provider networks* section of the Install Tutorials and Guides depending on your distribution.
- Self-service networks: See the *Overview -> Networking Option 2: Self-service networks* section of the Install Tutorials and Guides depending on your distribution.

OpenStack does not require a significant amount of resources and the following minimum requirements should support a proof-of-concept high availability environment with core services and several instances:

Node type	Processor Cores	Memory	Storage	NIC
controller node	4	12 GB	120 GB	2
compute node	8+	12+ GB	120+ GB	2

We recommended that the maximum latency between any two controller nodes is 2 milliseconds. Although the cluster software can be tuned to operate at higher latencies, some vendors insist on this value before agreeing to support the installation.

You can use the *ping* command to find the latency between two servers.

Virtualized hardware

For demonstrations and studying, you can set up a test environment on virtual machines (VMs). This has the following benefits:

- One physical server can support multiple nodes, each of which supports almost any number of network interfaces.
- You can take periodic snap shots throughout the installation process and roll back to a working configuration in the event of a problem.

However, running an OpenStack environment on VMs degrades the performance of your instances, particularly if your hypervisor or processor lacks support for hardware acceleration of nested VMs.

Note: When installing highly available OpenStack on VMs, be sure that your hypervisor permits promiscuous mode and disables MAC address filtering on the external network.

Installing the operating system

The first step in setting up your highly available OpenStack cluster is to install the operating system on each node. Follow the instructions in the *Environment* section of the Installation Tutorials and Guides depending on your distribution.

The OpenStack Installation Tutorials and Guides also include a list of the services that use passwords with important notes about using them.

Note: Before following this guide to configure the highly available OpenStack cluster, ensure the IP `10.0.0.11` and hostname `controller` are not in use.

This guide uses the following example IP addresses:

```
# controller
10.0.0.11       controller # virtual IP
10.0.0.12       controller1
10.0.0.13       controller2
10.0.0.14       controller3
```

Configure NTP

You must configure NTP to properly synchronize services among nodes. We recommend that you configure the controller node to reference more accurate (lower stratum) servers and other nodes to reference the controller node. For more information, see the Installation Tutorials and Guides.

Installing Memcached

Most OpenStack services can use Memcached to store ephemeral data such as tokens. Although Memcached does not support typical forms of redundancy such as clustering, OpenStack services can use almost any number of instances by configuring multiple hostnames or IP addresses.

The Memcached client implements hashing to balance objects among the instances. Failure of an instance only impacts a percentage of the objects, and the client automatically removes it from the list of instances.

To install and configure Memcached, read the official documentation.

Memory caching is managed by oslo.cache. This ensures consistency across all projects when using multiple Memcached servers. The following is an example configuration with three hosts:

```
Memcached_servers = controller1:11211,controller2:11211,controller3:11211
```

By default, `controller1` handles the caching service. If the host goes down, `controller2` or `controller3` will complete the service.

For more information about Memcached installation, see the *Environment -> Memcached* section in the Installation Tutorials and Guides depending on your distribution.

Configuring the shared services

This chapter describes the shared services for high availability, such as database, messaging service.

Database (Galera Cluster) for high availability

Configuration

Before you launch Galera Cluster, you need to configure the server and the database to operate as part of the cluster.

Configuring the server

Certain services running on the underlying operating system of your OpenStack database may block Galera Cluster from normal operation or prevent `mysqld` from achieving network connectivity with the cluster.

Firewall

Galera Cluster requires that you open the following ports to network traffic:

- On `3306`, Galera Cluster uses TCP for database client connections and State Snapshot Transfers methods that require the client, (that is, `mysqldump`).

- On `4567`, Galera Cluster uses TCP for replication traffic. Multicast replication uses both TCP and UDP on this port.
- On `4568`, Galera Cluster uses TCP for Incremental State Transfers.
- On `4444`, Galera Cluster uses TCP for all other State Snapshot Transfer methods.

See also:

For more information on firewalls, see firewalls and default ports in OpenStack Administrator Guide.

This can be achieved using the **`iptables`** command:

```
# iptables --append INPUT --in-interface eth0 \
  --protocol tcp --match tcp --dport ${PORT} \
  --source ${NODE-IP-ADDRESS} --jump ACCEPT
```

Make sure to save the changes once you are done. This will vary depending on your distribution:

- For Ubuntu
- For Fedora

Alternatively, make modifications using the `firewall-cmd` utility for FirewallD that is available on many Linux distributions:

```
# firewall-cmd --add-service=mysql --permanent
# firewall-cmd --add-port=3306/tcp --permanent
```

SELinux

Security-Enhanced Linux is a kernel module for improving security on Linux operating systems. It is commonly enabled and configured by default on Red Hat-based distributions. In the context of Galera Cluster, systems with SELinux may block the database service, keep it from starting, or prevent it from establishing network connections with the cluster.

To configure SELinux to permit Galera Cluster to operate, you may need to use the `semanage` utility to open the ports it uses. For example:

```
# semanage port -a -t mysqld_port_t -p tcp 3306
```

Older versions of some distributions, which do not have an up-to-date policy for securing Galera, may also require SELinux to be more relaxed about database access and actions:

```
# semanage permissive -a mysqld_t
```

Note: Bear in mind, leaving SELinux in permissive mode is not a good security practice. Over the longer term, you need to develop a security policy for Galera Cluster and then switch SELinux back into enforcing mode.

For more information on configuring SELinux to work with Galera Cluster, see the SELinux Documentation

---

AppArmor

Application Armor is a kernel module for improving security on Linux operating systems. It is developed by Canonical and commonly used on Ubuntu-based distributions. In the context of Galera Cluster, systems with AppArmor may block the database service from operating normally.

To configure AppArmor to work with Galera Cluster, complete the following steps on each cluster node:

1. Create a symbolic link for the database server in the `disable` directory:

   ```
   # ln -s /etc/apparmor.d/usr /etc/apparmor.d/disable/.sbin.mysqld
   ```

2. Restart AppArmor. For servers that use `init`, run the following command:

   ```
   # service apparmor restart
   ```

   For servers that use `systemd`, run the following command:

   ```
   # systemctl restart apparmor
   ```

AppArmor now permits Galera Cluster to operate.

Database configuration

MySQL databases, including MariaDB and Percona XtraDB, manage their configurations using a `my.cnf` file, which is typically located in the `/etc` directory. Configuration options available in these databases are also available in Galera Cluster, with some restrictions and several additions.

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
binlog_format=ROW
bind-address=10.0.0.12

# InnoDB Configuration
default_storage_engine=innodb
innodb_autoinc_lock_mode=2
innodb_flush_log_at_trx_commit=0
innodb_buffer_pool_size=122M

# Galera Cluster Configuration
wsrep_provider=/usr/lib/libgalera_smm.so
wsrep_provider_options="pc.recovery=TRUE;gcache.size=300M"
wsrep_cluster_name="my_example_cluster"
wsrep_cluster_address="gcomm://GALERA1-IP,GALERA2-IP,GALERA3-IP"
wsrep_sst_method=rsync
```

Configuring mysqld

While all of the configuration parameters available to the standard MySQL, MariaDB, or Percona XtraDB database servers are available in Galera Cluster, there are some that you must define an outset to avoid conflict or unexpected behavior.

- Ensure that the database server is not bound only to the localhost: `127.0.0.1`. Also, do not bind it to `0.0.0.0`. Binding to the localhost or `0.0.0.0` makes `mySQL` bind to all IP addresses on the machine, including the virtual IP address causing `HAProxy` not to start. Instead, bind to the management IP address of the controller node to enable access by other nodes through the management network:

```
bind-address=10.0.0.12
```

- Ensure that the binary log format is set to use row-level replication, as opposed to statement-level replication:

```
binlog_format=ROW
```

Configuring InnoDB

Galera Cluster does not support non-transactional storage engines and requires that you use InnoDB by default. There are some additional parameters that you must define to avoid conflicts.

- Ensure that the default storage engine is set to InnoDB:

```
default_storage_engine=InnoDB
```

- Ensure that the InnoDB locking mode for generating auto-increment values is set to `2`, which is the interleaved locking mode:

```
innodb_autoinc_lock_mode=2
```

  Do not change this value. Other modes may cause `INSERT` statements on tables with auto-increment columns to fail as well as unresolved deadlocks that leave the system unresponsive.

- Ensure that the InnoDB log buffer is written to file once per second, rather than on each commit, to improve performance:

```
innodb_flush_log_at_trx_commit=0
```

  Setting this parameter to `1` or `2` can improve performance, but it introduces certain dangers. Operating system failures can erase the last second of transactions. While you can recover this data from another node, if the cluster goes down at the same time (in the event of a data center power outage), you lose this data permanently.

- Define the InnoDB memory buffer pool size. The default value is 128 MB, but to compensate for Galera Cluster's additional memory usage, scale your usual value back by 5%:

```
innodb_buffer_pool_size=122M
```

Configuring wsrep replication

Galera Cluster configuration parameters all have the `wsrep_` prefix. You must define the following parameters for each cluster node in your OpenStack database.

- **wsrep Provider**: The Galera Replication Plugin serves as the `wsrep` provider for Galera Cluster. It is installed on your system as the `libgalera_smm.so` file. Define the path to this file in your `my.cnf`:

```
wsrep_provider="/usr/lib/libgalera_smm.so"
```

- **Cluster Name**: Define an arbitrary name for your cluster.

  ```
  wsrep_cluster_name="my_example_cluster"
  ```

  You must use the same name on every cluster node. The connection fails when this value does not match.

- **Cluster Address**: List the IP addresses for each cluster node.

  ```
  wsrep_cluster_address="gcomm://192.168.1.1,192.168.1.2,192.168.1.3"
  ```

  Replace the IP addresses given here with comma-separated list of each OpenStack database in your cluster.

- **Node Name**: Define the logical name of the cluster node.

  ```
  wsrep_node_name="Galera1"
  ```

- **Node Address**: Define the IP address of the cluster node.

  ```
  wsrep_node_address="192.168.1.1"
  ```

Additional parameters

For a complete list of the available parameters, run the `SHOW VARIABLES` command from within the database client:

```
SHOW VARIABLES LIKE 'wsrep_%';

+------------------------------+-------+
| Variable_name                | Value |
+------------------------------+-------+
| wsrep_auto_increment_control | ON    |
+------------------------------+-------+
| wsrep_causal_reads           | OFF   |
+------------------------------+-------+
| wsrep_certify_nonPK          | ON    |
+------------------------------+-------+
| ...                          | ...   |
+------------------------------+-------+
| wsrep_sync_wait              | 0     |
+------------------------------+-------+
```

For documentation about these parameters, `wsrep` provider option, and status variables available in Galera Cluster, see the Galera cluster Reference.

Management

When you finish installing and configuring the OpenStack database, you can initialize the Galera Cluster.

Prerequisites

- Database hosts with Galera Cluster installed
- A minimum of three hosts
- No firewalls between the hosts
- SELinux and AppArmor set to permit access to `mysqld`
- The correct path to `libgalera_smm.so` given to the `wsrep_provider` parameter

Initializing the cluster

In the Galera Cluster, the Primary Component is the cluster of database servers that replicate into each other. In the event that a cluster node loses connectivity with the Primary Component, it defaults into a non-operational state, to avoid creating or serving inconsistent data.

By default, cluster nodes do not start as part of a Primary Component. In the Primary Component, replication and state transfers bring all databases to the same state.

To start the cluster, complete the following steps:

1. Initialize the Primary Component on one cluster node. For servers that use `init`, run the following command:

   ```
   # service mysql start --wsrep-new-cluster
   ```

   For servers that use `systemd`, run the following command:

   ```
   # systemctl start mariadb --wsrep-new-cluster
   ```

2. Once the database server starts, check the cluster status using the `wsrep_cluster_size` status variable. From the database client, run the following command:

   ```
   SHOW STATUS LIKE 'wsrep_cluster_size';

   +--------------------+-------+
   | Variable_name      | Value |
   +--------------------+-------+
   | wsrep_cluster_size | 1     |
   +--------------------+-------+
   ```

3. Start the database server on all other cluster nodes. For servers that use `init`, run the following command:

   ```
   # service mysql start
   ```

   For servers that use `systemd`, run the following command:

   ```
   # systemctl start mariadb
   ```

4. When you have all cluster nodes started, log into the database client of any cluster node and check the `wsrep_cluster_size` status variable again:

   ```
   SHOW STATUS LIKE 'wsrep_cluster_size';
   ```

```
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 3     |
+--------------------+-------+
```

When each cluster node starts, it checks the IP addresses given to the `wsrep_cluster_address` parameter. It then attempts to establish network connectivity with a database server running there. Once it establishes a connection, it attempts to join the Primary Component, requesting a state transfer as needed to bring itself into sync with the cluster.

Note: In the event that you need to restart any cluster node, you can do so. When the database server comes back it, it establishes connectivity with the Primary Component and updates itself to any changes it may have missed while down.

Restarting the cluster

Individual cluster nodes can stop and be restarted without issue. When a database loses its connection or restarts, the Galera Cluster brings it back into sync once it reestablishes connection with the Primary Component. In the event that you need to restart the entire cluster, identify the most advanced cluster node and initialize the Primary Component on that node.

To find the most advanced cluster node, you need to check the sequence numbers, or the `seqnos`, on the last committed transaction for each. You can find this by viewing `grastate.dat` file in database directory:

```
$ cat /path/to/datadir/grastate.dat

# Galera saved state
version: 3.8
uuid:    5ee99582-bb8d-11e2-b8e3-23de375c1d30
seqno:   8204503945773
```

Alternatively, if the database server is running, use the `wsrep_last_committed` status variable:

```
SHOW STATUS LIKE 'wsrep_last_committed';

+----------------------+--------+
| Variable_name        | Value  |
+----------------------+--------+
| wsrep_last_committed | 409745 |
+----------------------+--------+
```

This value increments with each transaction, so the most advanced node has the highest sequence number and therefore is the most up to date.

Configuration tips

Deployment strategies

Galera can be configured using one of the following strategies:

- Each instance has its own IP address:

  OpenStack services are configured with the list of these IP addresses so they can select one of the addresses from those available.

- Galera runs behind HAProxy:

  HAProxy load balances incoming requests and exposes just one IP address for all the clients.

  Galera synchronous replication guarantees a zero slave lag. The failover procedure completes once HAProxy detects that the active back end has gone down and switches to the backup one, which is then marked as `UP`. If no back ends are `UP`, the failover procedure finishes only when the Galera Cluster has been successfully reassembled. The SLA is normally no more than 5 minutes.

- Use MySQL/Galera in active/passive mode to avoid deadlocks on `SELECT ... FOR UPDATE` type queries (used, for example, by nova and neutron). This issue is discussed in the following:

  - IMPORTANT: MySQL Galera does *not* support SELECT ... FOR UPDATE
  - Understanding reservations, concurrency, and locking in Nova

Configuring HAProxy

If you use HAProxy as a load-balancing client to provide access to the Galera Cluster, as described in the *HAProxy*, you can use the `clustercheck` utility to improve health checks.

1. Create a configuration file for `clustercheck` at `/etc/sysconfig/clustercheck`:

```
MYSQL_USERNAME="clustercheck_user"
MYSQL_PASSWORD="my_clustercheck_password"
MYSQL_HOST="localhost"
MYSQL_PORT="3306"
```

Note: For Ubuntu 16.04.1: Create a configuration file for `clustercheck` at `/etc/default/clustercheck`.

1. Log in to the database client and grant the `clustercheck` user `PROCESS` privileges:

```
GRANT PROCESS ON *.* TO 'clustercheck_user'@'localhost'
IDENTIFIED BY 'my_clustercheck_password';

FLUSH PRIVILEGES;
```

   You only need to do this on one cluster node. Galera Cluster replicates the user to all the others.

2. Create a configuration file for the HAProxy monitor service, at `/etc/xinetd.d/galera-monitor`:

```
service galera-monitor
{
   port = 9200
   disable = no
   socket_type = stream
   protocol = tcp
   wait = no
   user = root
   group = root
   groups = yes
```

```
   server = /usr/bin/clustercheck
   type = UNLISTED
   per_source = UNLIMITED
   log_on_success =
   log_on_failure = HOST
   flags = REUSE
}
```

3. Start the `xinetd` daemon for `clustercheck`. For servers that use `init`, run the following commands:

```
# service xinetd enable
# service xinetd start
```

   For servers that use `systemd`, run the following commands:

```
# systemctl daemon-reload
# systemctl enable xinetd
# systemctl start xinetd
```

The first step is to install the database that sits at the heart of the cluster. To implement high availability, run an instance of the database on each controller node and use Galera Cluster to provide replication between them. Galera Cluster is a synchronous multi-master database cluster, based on MySQL and the InnoDB storage engine. It is a high-availability service that provides high system uptime, no data loss, and scalability for growth.

You can achieve high availability for the OpenStack database in many different ways, depending on the type of database that you want to use. There are three implementations of Galera Cluster available to you:

- Galera Cluster for MySQL: The MySQL reference implementation from Codership, Oy.
- MariaDB Galera Cluster: The MariaDB implementation of Galera Cluster, which is commonly supported in environments based on Red Hat distributions.
- Percona XtraDB Cluster: The XtraDB implementation of Galera Cluster from Percona.

In addition to Galera Cluster, you can also achieve high availability through other database options, such as PostgreSQL, which has its own replication system.

Messaging service for high availability

An AMQP (Advanced Message Queuing Protocol) compliant message bus is required for most OpenStack components in order to coordinate the execution of jobs entered into the system.

The most popular AMQP implementation used in OpenStack installations is RabbitMQ.

RabbitMQ nodes fail over on the application and the infrastructure layers.

The application layer is controlled by the `oslo.messaging` configuration options for multiple AMQP hosts. If the AMQP node fails, the application reconnects to the next one configured within the specified reconnect interval. The specified reconnect interval constitutes its SLA.

On the infrastructure layer, the SLA is the time for which RabbitMQ cluster reassembles. Several cases are possible. The Mnesia keeper node is the master of the corresponding Pacemaker resource for RabbitMQ. When it fails, the result is a full AMQP cluster downtime interval. Normally, its SLA is no more than several minutes. Failure of another node that is a slave of the corresponding Pacemaker resource for RabbitMQ results in no AMQP cluster downtime at all.

Making the RabbitMQ service highly available involves the following steps:

- *Install RabbitMQ*
- *Configure RabbitMQ for HA queues*
- *Configure OpenStack services to use RabbitMQ HA queues*

Note: Access to RabbitMQ is not normally handled by HAProxy. Instead, consumers must be supplied with the full list of hosts running RabbitMQ with `rabbit_hosts` and turn on the `rabbit_ha_queues` option. For more information, read the core issue. For more detail, read the history and solution.

Install RabbitMQ

The commands for installing RabbitMQ are specific to the Linux distribution you are using.

For Ubuntu or Debian:

For RHEL, Fedora, or CentOS:

For openSUSE:

For SLES 12:

Note: For SLES 12, the packages are signed by GPG key 893A90DAD85F9316. You should verify the fingerprint of the imported GPG key before using it.

```
Key ID: 893A90DAD85F9316
Key Name: Cloud:OpenStack OBS Project <Cloud:OpenStack@build.opensuse.org>
Key Fingerprint: 35B34E18ABC1076D66D5A86B893A90DAD85F9316
Key Created: Tue Oct  8 13:34:21 2013
Key Expires: Thu Dec 17 13:34:21 2015
```

For more information, see the official installation manual for the distribution:

- Debian and Ubuntu
- RPM based (RHEL, Fedora, CentOS, openSUSE)

Configure RabbitMQ for HA queues

The following components/services can work with HA queues:

- OpenStack Compute
- OpenStack Block Storage
- OpenStack Networking
- Telemetry

Consider that, while exchanges and bindings survive the loss of individual nodes, queues and their messages do not because a queue and its contents are located on one node. If we lose this node, we also lose the queue.

Mirrored queues in RabbitMQ improve the availability of service since it is resilient to failures.

Production servers should run (at least) three RabbitMQ servers for testing and demonstration purposes, however it is possible to run only two servers. In this section, we configure two nodes, called `rabbit1` and `rabbit2`. To build a broker, ensure that all nodes have the same Erlang cookie file.

1. Stop RabbitMQ and copy the cookie from the first node to each of the other node(s):

```
# scp /var/lib/rabbitmq/.erlang.cookie root@NODE:/var/lib/rabbitmq/.erlang.cookie
```

2. On each target node, verify the correct owner, group, and permissions of the file `erlang.cookie`:

```
# chown rabbitmq:rabbitmq /var/lib/rabbitmq/.erlang.cookie
# chmod 400 /var/lib/rabbitmq/.erlang.cookie
```

3. Start the message queue service on all nodes and configure it to start when the system boots. On Ubuntu, it is configured by default.

   On CentOS, RHEL, openSUSE, and SLES:

```
# systemctl enable rabbitmq-server.service
# systemctl start rabbitmq-server.service
```

4. Verify that the nodes are running:

```
# rabbitmqctl cluster_status
Cluster status of node rabbit@NODE...
[{nodes,[{disc,[rabbit@NODE]}]},
 {running_nodes,[rabbit@NODE]},
 {partitions,[]}]
...done.
```

5. Run the following commands on each node except the first one:

```
# rabbitmqctl stop_app
Stopping node rabbit@NODE...
...done.
# rabbitmqctl join_cluster --ram rabbit@rabbit1
# rabbitmqctl start_app
Starting node rabbit@NODE ...
...done.
```

Note: The default node type is a disc node. In this guide, nodes join the cluster as RAM nodes.

---

1. Verify the cluster status:

```
# rabbitmqctl cluster_status
Cluster status of node rabbit@NODE...
[{nodes,[{disc,[rabbit@rabbit1]},{ram,[rabbit@NODE]}]}, \
    {running_nodes,[rabbit@NODE,rabbit@rabbit1]}]
```

   If the cluster is working, you can create usernames and passwords for the queues.

2. To ensure that all queues except those with auto-generated names are mirrored across all running nodes, set the `ha-mode` policy key to all by running the following command on one of the nodes:

```
# rabbitmqctl set_policy ha-all '^(?!amq\.).*' '{"ha-mode": "all"}'
```

More information is available in the RabbitMQ documentation:

- Highly Available Queues
- Clustering Guide

Note: As another option to make RabbitMQ highly available, RabbitMQ contains the OCF scripts for the Pacemaker cluster resource agents since version 3.5.7. It provides the active/active RabbitMQ cluster with mirrored queues. For more information, see Auto-configuration of a cluster with a Pacemaker.

Configure OpenStack services to use Rabbit HA queues

Configure the OpenStack components to use at least two RabbitMQ nodes.

Use these steps to configurate all services using RabbitMQ:

1. RabbitMQ HA cluster Transport URL using `[user:pass@]host:port` format:

   ```
   transport_url = rabbit://RABBIT_USER:RABBIT_PASS@rabbit1:5672,
   RABBIT_USER:RABBIT_PASS@rabbit2:5672,RABBIT_USER:RABBIT_PASS@rabbit3:5672
   ```

   Replace `RABBIT_USER` with RabbitMQ username and `RABBIT_PASS` with password for respective RabbitMQ host. For more information, see oslo messaging transport.

2. Retry connecting with RabbitMQ:

   ```
   rabbit_retry_interval=1
   ```

3. How long to back-off for between retries when connecting to RabbitMQ:

   ```
   rabbit_retry_backoff=2
   ```

4. Maximum retries with trying to connect to RabbitMQ (infinite by default):

   ```
   rabbit_max_retries=0
   ```

5. Use durable queues in RabbitMQ:

   ```
   rabbit_durable_queues=true
   ```

6. Use HA queues in RabbitMQ (`x-ha-policy: all`):

   ```
   rabbit_ha_queues=true
   ```

Note: If you change the configuration from an old set-up that did not use HA queues, restart the service:

```
# rabbitmqctl stop_app
# rabbitmqctl reset
# rabbitmqctl start_app
```

Configuring the controller

The cloud controller runs on the management network and must talk to all other services.

The Pacemaker architecture

What is a cluster manager?

At its core, a cluster is a distributed finite state machine capable of co-ordinating the startup and recovery of inter-related services across a set of machines.

Even a distributed or replicated application that is able to survive failures on one or more machines can benefit from a cluster manager because a cluster manager has the following capabilities:

1. Awareness of other applications in the stack

   While SYS-V init replacements like systemd can provide deterministic recovery of a complex stack of services, the recovery is limited to one machine and lacks the context of what is happening on other machines. This context is crucial to determine the difference between a local failure, and clean startup and recovery after a total site failure.

2. Awareness of instances on other machines

   Services like RabbitMQ and Galera have complicated boot-up sequences that require co-ordination, and often serialization, of startup operations across all machines in the cluster. This is especially true after a site-wide failure or shutdown where you must first determine the last machine to be active.

3. A shared implementation and calculation of quorum

   It is very important that all members of the system share the same view of who their peers are and whether or not they are in the majority. Failure to do this leads very quickly to an internal split-brain state. This is where different parts of the system are pulling in different and incompatible directions.

4. Data integrity through fencing (a non-responsive process does not imply it is not doing anything)

   A single application does not have sufficient context to know the difference between failure of a machine and failure of the application on a machine. The usual practice is to assume the machine is dead and continue working, however this is highly risky. A rogue process or machine could still be responding to requests and generally causing havoc. The safer approach is to make use of remotely accessible power switches and/or network switches and SAN controllers to fence (isolate) the machine before continuing.

5. Automated recovery of failed instances

   While the application can still run after the failure of several instances, it may not have sufficient capacity to serve the required volume of requests. A cluster can automatically recover failed instances to prevent additional load induced failures.

For these reasons, we highly recommend the use of a cluster manager like Pacemaker.

Deployment flavors

It is possible to deploy three different flavors of the Pacemaker architecture. The two extremes are `Collapsed` (where every component runs on every node) and `Segregated` (where every component runs in its own 3+ node cluster).

Regardless of which flavor you choose, we recommend that clusters contain at least three nodes so that you can take advantage of quorum.

Quorum becomes important when a failure causes the cluster to split in two or more partitions. In this situation, you want the majority members of the system to ensure the minority are truly dead (through fencing) and continue to host resources. For a two-node cluster, no side has the majority and you can end up in a situation where both sides fence each other, or both sides are running the same services. This can lead to data corruption.

Clusters with an even number of hosts suffer from similar issues. A single network failure could easily cause a N:N split where neither side retains a majority. For this reason, we recommend an odd number of cluster members when scaling up.

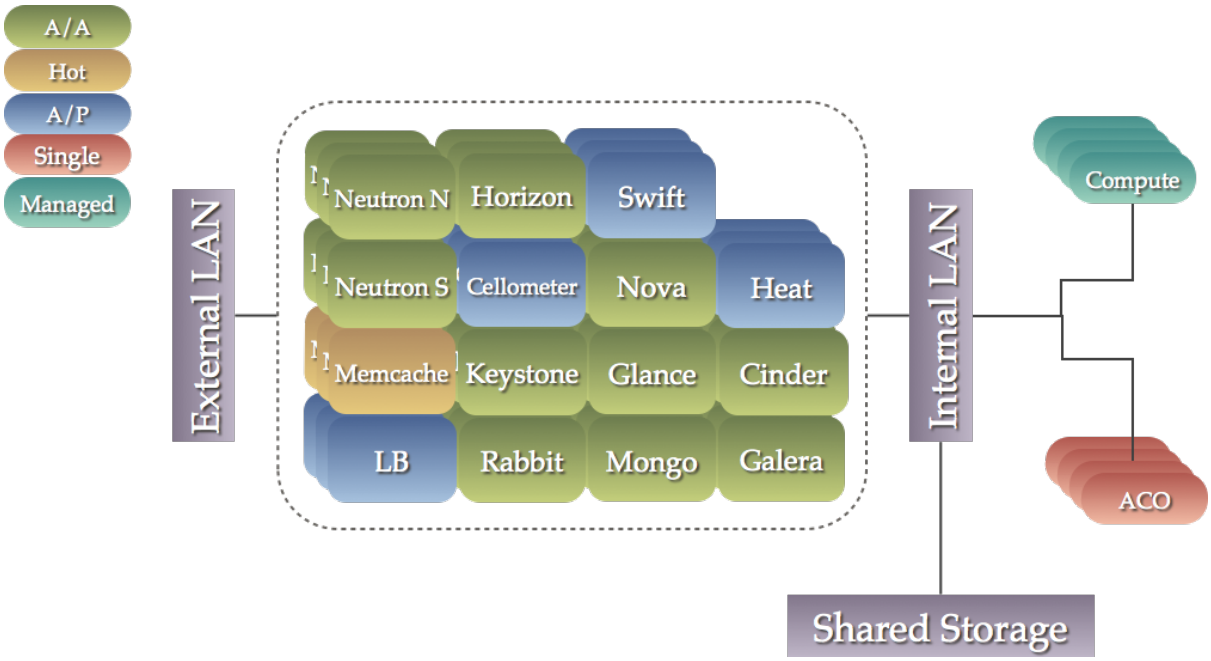
You can have up to 16 cluster members (this is currently limited by the ability of corosync to scale higher). In extreme cases, 32 and even up to 64 nodes could be possible. However, this is not well tested.

Collapsed

In a collapsed configuration, there is a single cluster of 3 or more nodes on which every component is running.

This scenario has the advantage of requiring far fewer, if more powerful, machines. Additionally, being part of a single cluster allows you to accurately model the ordering dependencies between components.

This scenario can be visualized as below.



You would choose this option if you prefer to have fewer but more powerful boxes.

This is the most common option and the one we document here.

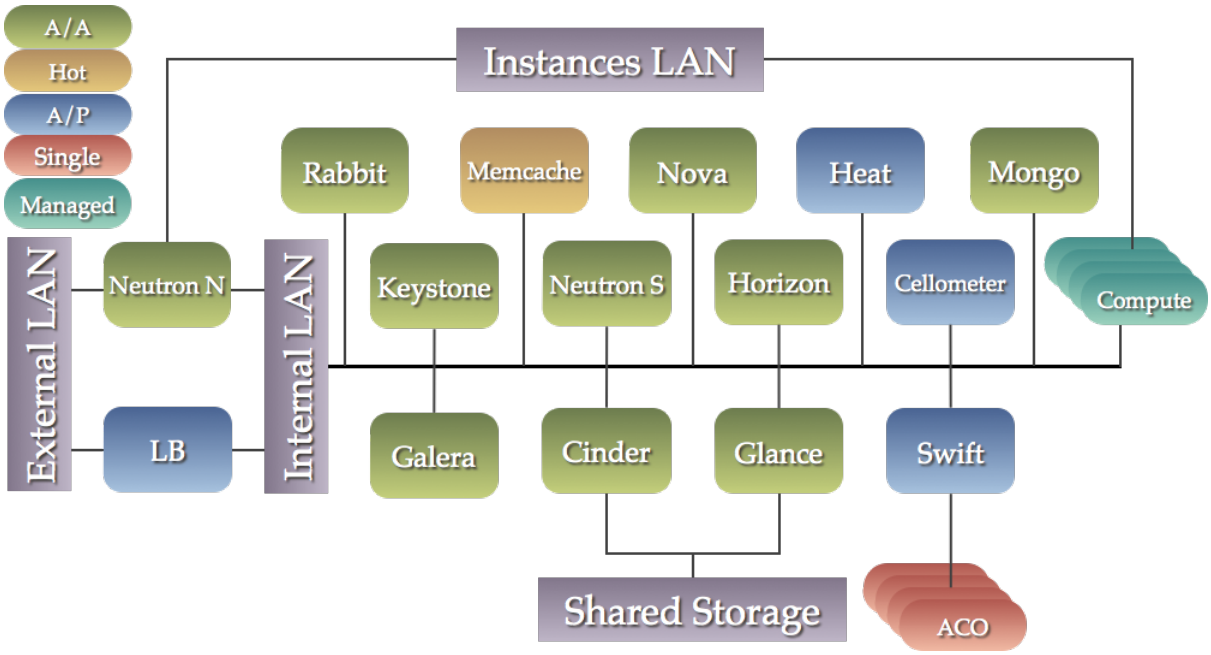
Segregated

In this configuration, each service runs in a dedicated cluster of 3 or more nodes.

The benefits of this approach are the physical isolation between components and the ability to add capacity to specific components.

You would choose this option if you prefer to have more but less powerful boxes.

This scenario can be visualized as below, where each box below represents a cluster of three or more guests.



Mixed

It is also possible to follow a segregated approach for one or more components that are expected to be a bottleneck and use a collapsed approach for the remainder.

Proxy server

Almost all services in this stack benefit from being proxied. Using a proxy server provides the following capabilities:

1. Load distribution

   Many services can act in an active/active capacity, however, they usually require an external mechanism for distributing requests to one of the available instances. The proxy server can serve this role.

2. API isolation

   By sending all API access through the proxy, you can clearly identify service interdependencies. You can also move them to locations other than `localhost` to increase capacity if the need arises.

3. Simplified process for adding/removing of nodes

   Since all API access is directed to the proxy, adding or removing nodes has no impact on the configuration of other services. This can be very useful in upgrade scenarios where an entirely new set of machines can be configured and tested in isolation before telling the proxy to direct traffic there instead.

4. Enhanced failure detection

   The proxy can be configured as a secondary mechanism for detecting service failures. It can even be configured to look for nodes in a degraded state (such as being too far behind in the replication) and take them out of circulation.

The following components are currently unable to benefit from the use of a proxy server:

- RabbitMQ
- Memcached
- MongoDB

We recommend HAProxy as the load balancer, however, there are many alternative load balancing solutions in the marketplace.

Generally, we use round-robin to distribute load amongst instances of active/active services. Alternatively, Galera uses `stack-table` options to ensure that incoming connection to virtual IP (VIP) are directed to only one of the available back ends. This helps avoid lock contention and prevent deadlocks, although Galera can run active/active. Used in combination with the `httpchk` option, this ensure only nodes that are in sync with their peers are allowed to handle requests.

Pacemaker cluster stack

Pacemaker cluster stack is a state-of-the-art high availability and load balancing stack for the Linux platform. Pacemaker is used to make OpenStack infrastructure highly available.

Note: It is storage and application-agnostic, and in no way specific to OpenStack.

Pacemaker relies on the Corosync messaging layer for reliable cluster communications. Corosync implements the Totem single-ring ordering and membership protocol. It also provides UDP and InfiniBand based messaging, quorum, and cluster membership to Pacemaker.

Pacemaker does not inherently understand the applications it manages. Instead, it relies on resource agents (RAs) that are scripts that encapsulate the knowledge of how to start, stop, and check the health of each application managed by the cluster.

These agents must conform to one of the OCF, SysV Init, Upstart, or Systemd standards.

Pacemaker ships with a large set of OCF agents (such as those managing MySQL databases, virtual IP addresses, and RabbitMQ), but can also use any agents already installed on your system and can be extended with your own (see the developer guide).

The steps to implement the Pacemaker cluster stack are:

- *Install packages*
- *Set up the cluster with pcs*
- *Start Corosync*
- *Start Pacemaker*
- *Set basic cluster properties*

Install packages

On any host that is meant to be part of a Pacemaker cluster, establish cluster communications through the Corosync messaging layer. This involves installing the following packages (and their dependencies, which your package manager usually installs automatically):

- *pacemaker*
- *pcs* (CentOS or RHEL) or crmsh
- *corosync*
- *fence-agents* (CentOS or RHEL) or cluster-glue
- *resource-agents*
- *libqb0*

Set up the cluster with pcs

1. Make sure *pcs* is running and configured to start at boot time:

```
$ systemctl enable pcsd
$ systemctl start pcsd
```

2. Set a password for hacluster user on each host:

```
$ echo my-secret-password-no-dont-use-this-one \
  | passwd --stdin hacluster
```

Note: Since the cluster is a single administrative domain, it is acceptable to use the same password on all nodes.

3. Use that password to authenticate to the nodes that will make up the cluster:

```
$ pcs cluster auth controller1 controller2 controller3 \
  -u hacluster -p my-secret-password-no-dont-use-this-one --force
```

Note: The `-p` option is used to give the password on command line and makes it easier to script.

4. Create and name the cluster. Then, start it and enable all components to auto-start at boot time:

```
$ pcs cluster setup --force --name my-first-openstack-cluster \
  controller1 controller2 controller3
$ pcs cluster start --all
$ pcs cluster enable --all
```

Note: In Red Hat Enterprise Linux or CentOS environments, this is a recommended path to perform configuration. For more information, see the RHEL docs.

Set up the cluster with *crmsh*

After installing the Corosync package, you must create the `/etc/corosync/corosync.conf` configuration file.

Note: For Ubuntu, you should also enable the Corosync service in the `/etc/default/corosync` configuration file.

Corosync can be configured to work with either multicast or unicast IP addresses or to use the votequorum library.

- *Set up Corosync with multicast*
- *Set up Corosync with unicast*
- *Set up Corosync with votequorum library*

Set up Corosync with multicast

Most distributions ship an example configuration file (`corosync.conf.example`) as part of the documentation bundled with the Corosync package. An example Corosync configuration file is shown below:

Example Corosync configuration file for multicast ("corosync.conf")

```
totem {
      version: 2

      # Time (in ms) to wait for a token (1)
      token: 10000

     # How many token retransmits before forming a new
     # configuration
     token_retransmits_before_loss_const: 10

     # Turn off the virtual synchrony filter
     vsftype: none

     # Enable encryption (2)
     secauth: on

     # How many threads to use for encryption/decryption
     threads: 0

     # This specifies the redundant ring protocol, which may be
     # none, active, or passive. (3)
     rrp_mode: active

     # The following is a two-ring multicast configuration. (4)
     interface {
             ringnumber: 0
             bindnetaddr: 10.0.0.0
             mcastaddr: 239.255.42.1
             mcastport: 5405
     }
     interface {
             ringnumber: 1
             bindnetaddr: 10.0.42.0
             mcastaddr: 239.255.42.2
             mcastport: 5405
     }
```

```
}

amf {
     mode: disabled
}

service {
        # Load the Pacemaker Cluster Resource Manager (5)
        ver:       1
        name:      pacemaker
}

aisexec {
        user:   root
        group:  root
}

logging {
        fileline: off
        to_stderr: yes
        to_logfile: no
        to_syslog: yes
        syslog_facility: daemon
        debug: off
        timestamp: on
        logger_subsys {
                subsys: AMF
                debug: off
                tags: enter|leave|trace1|trace2|trace3|trace4|trace6
        }}
```

Note the following:

- The `token` value specifies the time, in milliseconds, during which the Corosync token is expected to be transmitted around the ring. When this timeout expires, the token is declared lost, and after `token_retransmits_before_loss_const lost` tokens, the non-responding processor (cluster node) is declared dead. `token × token_retransmits_before_loss_const` is the maximum time a node is allowed to not respond to cluster messages before being considered dead. The default for token is 1000 milliseconds (1 second), with 4 allowed retransmits. These defaults are intended to minimize failover times, but can cause frequent false alarms and unintended failovers in case of short network interruptions. The values used here are safer, albeit with slightly extended failover times.
- With `secauth` enabled, Corosync nodes mutually authenticates using a 128-byte shared secret stored in the `/etc/corosync/authkey` file. This can be generated with the **`corosync-keygen`** utility. Cluster communications are encrypted when using `secauth`.
- In Corosync, configurations use redundant networking (with more than one interface). This means you must select a Redundant Ring Protocol (RRP) mode other than none. We recommend `active` as the RRP mode.

  Note the following about the recommended interface configuration:

  - Each configured interface must have a unique `ringnumber`, starting with 0.
  - The `bindnetaddr` is the network address of the interfaces to bind to. The example uses two network addresses of /24 IPv4 subnets.
  - Multicast groups (`mcastaddr`) must not be reused across cluster boundaries. No two distinct clus-

ters should ever use the same multicast group. Be sure to select multicast addresses compliant with RFC 2365, "Administratively Scoped IP Multicast".

  - For firewall configurations, Corosync communicates over UDP only, and uses `mcastport` (for receives) and `mcastport - 1` (for sends).

- The service declaration for the Pacemaker service may be placed in the `corosync.conf` file directly or in its own separate file, `/etc/corosync/service.d/pacemaker`.

Note: If you are using Corosync version 2 on Ubuntu 14.04, remove or comment out lines under the service stanza. These stanzas enable Pacemaker to start up. Another potential problem is the boot and shutdown order of Corosync and Pacemaker. To force Pacemaker to start after Corosync and stop before Corosync, fix the start and kill symlinks manually:

```
# update-rc.d pacemaker start 20 2 3 4 5 . stop 00 0 1 6 .
```

The Pacemaker service also requires an additional configuration file `/etc/corosync/uidgid.d/pacemaker` to be created with the following content:

```
uidgid {
  uid: hacluster
  gid: haclient
}
```

- Once created, synchronize the `corosync.conf` file (and the `authkey` file if the secauth option is enabled) across all cluster nodes.

Set up Corosync with unicast

For environments that do not support multicast, Corosync should be configured for unicast. An example fragment of the `corosync.conf` file for unicastis is shown below:

Corosync configuration file fragment for unicast ("corosync.conf")

```
totem {
        #...
        interface {
                ringnumber: 0
                bindnetaddr: 10.0.0.0
                broadcast: yes (1)
                mcastport: 5405
        }
        interface {
                ringnumber: 1
                bindnetaddr: 10.0.42.0
                broadcast: yes
                mcastport: 5405
        }
        transport: udpu (2)
}

nodelist { (3)
        node {
                ring0_addr: 10.0.0.12
```

```
                ring1_addr: 10.0.42.12
                nodeid: 1
        }
        node {
                ring0_addr: 10.0.0.13
                ring1_addr: 10.0.42.13
                nodeid: 2
        }
        node {
                ring0_addr: 10.0.0.14
                ring1_addr: 10.0.42.14
                nodeid: 3
        }
}
#...
```

Note the following:

- If the `broadcast` parameter is set to `yes`, the broadcast address is used for communication. If this option is set, the `mcastaddr` parameter should not be set.
- The `transport` directive controls the transport mechanism. To avoid the use of multicast entirely, specify the `udpu` unicast transport parameter. This requires specifying the list of members in the `nodelist` directive. This potentially makes up the membership before deployment. The default is `udp`. The transport type can also be set to `udpu` or `iba`.
- Within the `nodelist` directive, it is possible to specify specific information about the nodes in the cluster. The directive can contain only the node sub-directive, which specifies every node that should be a member of the membership, and where non-default options are needed. Every node must have at least the `ring0_addr` field filled.

  Note: For UDPU, every node that should be a member of the membership must be specified.

  Possible options are:

  - `ring{X}_addr` specifies the IP address of one of the nodes. `{X}` is the ring number.
  - `nodeid` is optional when using IPv4 and required when using IPv6. This is a 32-bit value specifying the node identifier delivered to the cluster membership service. If this is not specified with IPv4, the node ID is determined from the 32-bit IP address of the system to which the system is bound with ring identifier of 0. The node identifier value of zero is reserved and should not be used.

Set up Corosync with votequorum library

The votequorum library is part of the Corosync project. It provides an interface to the vote-based quorum service and it must be explicitly enabled in the Corosync configuration file. The main role of votequorum library is to avoid split-brain situations, but it also provides a mechanism to:

- Query the quorum status
- List the nodes known to the quorum service
- Receive notifications of quorum state changes
- Change the number of votes assigned to a node

- Change the number of expected votes for a cluster to be quorate
- Connect an additional quorum device to allow small clusters remain quorate during node outages

The votequorum library has been created to replace and eliminate `qdisk`, the disk-based quorum daemon for CMAN, from advanced cluster configurations.

A sample votequorum service configuration in the `corosync.conf` file is:

```
quorum {
        provider: corosync_votequorum (1)
        expected_votes: 7 (2)
        wait_for_all: 1 (3)
        last_man_standing: 1 (4)
        last_man_standing_window: 10000 (5)
       }
```

Note the following:

- Specifying `corosync_votequorum` enables the votequorum library. This is the only required option.
- The cluster is fully operational with `expected_votes` set to 7 nodes (each node has 1 vote), quorum: 4. If a list of nodes is specified as `nodelist`, the `expected_votes` value is ignored.
- When you start up a cluster (all nodes down) and set `wait_for_all` to 1, the cluster quorum is held until all nodes are online and have joined the cluster for the first time. This parameter is new in Corosync 2.0.
- Setting `last_man_standing` to 1 enables the Last Man Standing (LMS) feature. By default, it is disabled (set to 0). If a cluster is on the quorum edge (`expected_votes:` set to 7; `online nodes:` set to 4) for longer than the time specified for the `last_man_standing_window` parameter, the cluster can recalculate quorum and continue operating even if the next node will be lost. This logic is repeated until the number of online nodes in the cluster reaches 2. In order to allow the cluster to step down from 2 members to only 1, the `auto_tie_breaker` parameter needs to be set. We do not recommended this for production environments.
- `last_man_standing_window` specifies the time, in milliseconds, required to recalculate quorum after one or more hosts have been lost from the cluster. To perform a new quorum recalculation, the cluster must have quorum for at least the interval specified for `last_man_standing_window`. The default is 10000ms.

Start Corosync

Corosync is started as a regular system service. Depending on your distribution, it may ship with an LSB init script, an upstart job, or a Systemd unit file.

- Start `corosync` with the LSB init script:

  ```
  # /etc/init.d/corosync start
  ```

  Alternatively:

  ```
  # service corosync start
  ```

- Start `corosync` with upstart:

  ```
  # start corosync
  ```

- Start `corosync` with systemd unit file:

```
# systemctl start corosync
```

You can now check the `corosync` connectivity with one of these tools.

Use the **`corosync-cfgtool`** utility with the `-s` option to get a summary of the health of the communication rings:

```
# corosync-cfgtool -s
Printing ring status.
Local node ID 435324542
RING ID 0
        id      = 10.0.0.82
        status  = ring 0 active with no faults
RING ID 1
        id      = 10.0.42.100
        status  = ring 1 active with no faults
```

Use the **`corosync-objctl`** utility to dump the Corosync cluster member list:

```
# corosync-objctl runtime.totem.pg.mrp.srp.members
runtime.totem.pg.mrp.srp.435324542.ip=r(0) ip(10.0.0.82) r(1) ip(10.0.42.100)
runtime.totem.pg.mrp.srp.435324542.join_count=1
runtime.totem.pg.mrp.srp.435324542.status=joined
runtime.totem.pg.mrp.srp.983895584.ip=r(0) ip(10.0.0.87) r(1) ip(10.0.42.254)
runtime.totem.pg.mrp.srp.983895584.join_count=1
runtime.totem.pg.mrp.srp.983895584.status=joined
```

You should see a `status=joined` entry for each of your constituent cluster nodes.

Note: If you are using Corosync version 2, use the **`corosync-cmapctl`** utility instead of **`corosync-objctl`**; it is a direct replacement.

Start Pacemaker

After the `corosync` service have been started and you have verified that the cluster is communicating properly, you can start **`pacemakerd`**, the Pacemaker master control process. Choose one from the following four ways to start it:

1. Start `pacemaker` with the LSB init script:

```
# /etc/init.d/pacemaker start
```

   Alternatively:

```
# service pacemaker start
```

2. Start `pacemaker` with upstart:

```
# start pacemaker
```

3. Start `pacemaker` with the systemd unit file:

```
# systemctl start pacemaker
```

After the `pacemaker` service has started, Pacemaker creates a default empty cluster configuration with no resources. Use the **`crm_mon`** utility to observe the status of `pacemaker`:

```
# crm_mon -1
Last updated: Sun Oct  7 21:07:52 2012
Last change: Sun Oct  7 20:46:00 2012 via cibadmin on controller2
Stack: openais
Current DC: controller2 - partition with quorum
Version: 1.1.6-9971ebba4494012a93c03b40a2c58ec0eb60f50c
3 Nodes configured, 3 expected votes
0 Resources configured.


Online: [ controller3 controller2 controller1 ]
...
```

Set basic cluster properties

After you set up your Pacemaker cluster, set a few basic cluster properties:

- `crmsh`

```
$ crm configure property pe-warn-series-max="1000" \
  pe-input-series-max="1000" \
  pe-error-series-max="1000" \
  cluster-recheck-interval="5min"
```

- `pcs`

```
$ pcs property set pe-warn-series-max=1000 \
  pe-input-series-max=1000 \
  pe-error-series-max=1000 \
  cluster-recheck-interval=5min
```

Note the following:

- Setting the `pe-warn-series-max`, `pe-input-series-max`, and `pe-error-series-max` parameters to 1000 instructs Pacemaker to keep a longer history of the inputs processed and errors and warnings generated by its Policy Engine. This history is useful if you need to troubleshoot the cluster.
- Pacemaker uses an event-driven approach to cluster state processing. The `cluster-recheck-interval` parameter (which defaults to 15 minutes) defines the interval at which certain Pacemaker actions occur. It is usually prudent to reduce this to a shorter interval, such as 5 or 3 minutes.

By default, STONITH is enabled in Pacemaker, but STONITH mechanisms (to shutdown a node via IPMI or ssh) are not configured. In this case Pacemaker will refuse to start any resources. For production cluster it is recommended to configure appropriate STONITH mechanisms. But for demo or testing purposes STONITH can be disabled completely as follows:

- `crmsh`

```
$ crm configure property stonith-enabled=false
```

- pcs

```
$ pcs property set stonith-enabled=false
```

After you make these changes, commit the updated configuration.

Configure the VIP

You must select and assign a virtual IP address (VIP) that can freely float between cluster nodes.

This configuration creates vip, a virtual IP address for use by the API node (10.0.0.11).

For crmsh:

```
# crm configure primitive vip ocf:heartbeat:IPaddr2 \
  params ip="10.0.0.11" cidr_netmask="24" op monitor interval="30s"
```

For pcs:

```
# pcs resource create vip ocf:heartbeat:IPaddr2 \
  params ip="10.0.0.11" cidr_netmask="24" op monitor interval="30s"
```

HAProxy

HAProxy provides a fast and reliable HTTP reverse proxy and load balancer for TCP or HTTP applications. It is particularly suited for web crawling under very high loads while needing persistence or Layer 7 processing. It realistically supports tens of thousands of connections with recent hardware.

Each instance of HAProxy configures its front end to accept connections only to the virtual IP (VIP) address. The HAProxy back end (termination point) is a list of all the IP addresses of instances for load balancing.

Note: Ensure your HAProxy installation is not a single point of failure, it is advisable to have multiple HAProxy instances running.

You can also ensure the availability by other means, using Keepalived or Pacemaker.

Alternatively, you can use a commercial load balancer, which is hardware or software. We recommend a hardware load balancer as it generally has good performance.

For detailed instructions about installing HAProxy on your nodes, see the HAProxy official documentation.

Configuring HAProxy

1. Restart the HAProxy service.
2. Locate your HAProxy instance on each OpenStack controller node in your environment. The following is an example /etc/haproxy/haproxy.cfg configuration file. Configure your instance using the following configuration file, you will need a copy of it on each controller node.

```
global
 chroot  /var/lib/haproxy
 daemon
 group  haproxy
```

```
  maxconn  4000
  pidfile  /var/run/haproxy.pid
  user  haproxy

defaults
  log  global
  maxconn  4000
  option  redispatch
  retries  3
  timeout  http-request 10s
  timeout  queue 1m
  timeout  connect 10s
  timeout  client 1m
  timeout  server 1m
  timeout  check 10s

 listen dashboard_cluster
  bind <Virtual IP>:443
  balance  source
  option  tcpka
  option  httpchk
  option  tcplog
  server controller1 10.0.0.12:443 check inter 2000 rise 2 fall 5
  server controller2 10.0.0.13:443 check inter 2000 rise 2 fall 5
  server controller3 10.0.0.14:443 check inter 2000 rise 2 fall 5

 listen galera_cluster
  bind <Virtual IP>:3306
  balance  source
  option  mysql-check
  server controller1 10.0.0.12:3306 check port 9200 inter 2000 rise 2 fall 5
  server controller2 10.0.0.13:3306 backup check port 9200 inter 2000 rise 2 fall 5
  server controller3 10.0.0.14:3306 backup check port 9200 inter 2000 rise 2 fall 5

 listen glance_api_cluster
  bind <Virtual IP>:9292
  balance  source
  option  tcpka
  option  httpchk
  option  tcplog
  server controller1 10.0.0.12:9292 check inter 2000 rise 2 fall 5
  server controller2 10.0.0.13:9292 check inter 2000 rise 2 fall 5
  server controller3 10.0.0.14:9292 check inter 2000 rise 2 fall 5

 listen glance_registry_cluster
  bind <Virtual IP>:9191
  balance  source
  option  tcpka
  option  tcplog
  server controller1 10.0.0.12:9191 check inter 2000 rise 2 fall 5
  server controller2 10.0.0.13:9191 check inter 2000 rise 2 fall 5
  server controller3 10.0.0.14:9191 check inter 2000 rise 2 fall 5

 listen keystone_admin_cluster
  bind <Virtual IP>:35357
  balance  source
  option  tcpka
```

```
 option  httpchk
 option  tcplog
 server controller1 10.0.0.12:35357 check inter 2000 rise 2 fall 5
 server controller2 10.0.0.13:35357 check inter 2000 rise 2 fall 5
 server controller3 10.0.0.14:35357 check inter 2000 rise 2 fall 5

listen keystone_public_internal_cluster
 bind <Virtual IP>:5000
 balance  source
 option  tcpka
 option  httpchk
 option  tcplog
 server controller1 10.0.0.12:5000 check inter 2000 rise 2 fall 5
 server controller2 10.0.0.13:5000 check inter 2000 rise 2 fall 5
 server controller3 10.0.0.14:5000 check inter 2000 rise 2 fall 5

listen nova_ec2_api_cluster
 bind <Virtual IP>:8773
 balance  source
 option  tcpka
 option  tcplog
 server controller1 10.0.0.12:8773 check inter 2000 rise 2 fall 5
 server controller2 10.0.0.13:8773 check inter 2000 rise 2 fall 5
 server controller3 10.0.0.14:8773 check inter 2000 rise 2 fall 5

listen nova_compute_api_cluster
 bind <Virtual IP>:8774
 balance  source
 option  tcpka
 option  httpchk
 option  tcplog
 server controller1 10.0.0.12:8774 check inter 2000 rise 2 fall 5
 server controller2 10.0.0.13:8774 check inter 2000 rise 2 fall 5
 server controller3 10.0.0.14:8774 check inter 2000 rise 2 fall 5

listen nova_metadata_api_cluster
 bind <Virtual IP>:8775
 balance  source
 option  tcpka
 option  tcplog
 server controller1 10.0.0.12:8775 check inter 2000 rise 2 fall 5
 server controller2 10.0.0.13:8775 check inter 2000 rise 2 fall 5
 server controller3 10.0.0.14:8775 check inter 2000 rise 2 fall 5

listen cinder_api_cluster
 bind <Virtual IP>:8776
 balance  source
 option  tcpka
 option  httpchk
 option  tcplog
 server controller1 10.0.0.12:8776 check inter 2000 rise 2 fall 5
 server controller2 10.0.0.13:8776 check inter 2000 rise 2 fall 5
 server controller3 10.0.0.14:8776 check inter 2000 rise 2 fall 5

listen ceilometer_api_cluster
 bind <Virtual IP>:8777
 balance  source
```

```
 option  tcpka
 option  tcplog
 server controller1 10.0.0.12:8777 check inter 2000 rise 2 fall 5
 server controller2 10.0.0.13:8777 check inter 2000 rise 2 fall 5
 server controller3 10.0.0.14:8777 check inter 2000 rise 2 fall 5

listen nova_vncproxy_cluster
 bind <Virtual IP>:6080
 balance  source
 option  tcpka
 option  tcplog
 server controller1 10.0.0.12:6080 check inter 2000 rise 2 fall 5
 server controller2 10.0.0.13:6080 check inter 2000 rise 2 fall 5
 server controller3 10.0.0.14:6080 check inter 2000 rise 2 fall 5

listen neutron_api_cluster
 bind <Virtual IP>:9696
 balance  source
 option  tcpka
 option  httpchk
 option  tcplog
 server controller1 10.0.0.12:9696 check inter 2000 rise 2 fall 5
 server controller2 10.0.0.13:9696 check inter 2000 rise 2 fall 5
 server controller3 10.0.0.14:9696 check inter 2000 rise 2 fall 5

listen swift_proxy_cluster
 bind <Virtual IP>:8080
 balance  source
 option  tcplog
 option  tcpka
 server controller1 10.0.0.12:8080 check inter 2000 rise 2 fall 5
 server controller2 10.0.0.13:8080 check inter 2000 rise 2 fall 5
 server controller3 10.0.0.14:8080 check inter 2000 rise 2 fall 5
```

Note: The Galera cluster configuration directive `backup` indicates that two of the three controllers are standby nodes. This ensures that only one node services write requests because OpenStack support for multi-node writes is not yet production-ready.

Note: The Telemetry API service configuration does not have the `option httpchk` directive as it cannot process this check properly.

---

1. Configure the kernel parameter to allow non-local IP binding. This allows running HAProxy instances to bind to a VIP for failover. Add following line to `/etc/sysctl.conf`:

```
net.ipv4.ip_nonlocal_bind = 1
```

2. Restart the host or, to make changes work immediately, invoke:

```
$ sysctl -p
```

3. Add HAProxy to the cluster and ensure the VIPs can only run on machines where HAProxy is active:

   `pcs`

```
$ pcs resource create lb-haproxy systemd:haproxy --clone
$ pcs constraint order start vip then lb-haproxy-clone kind=Optional
$ pcs constraint colocation add lb-haproxy-clone with vip
```

crmsh

```
$ crm cib new conf-haproxy
$ crm configure primitive haproxy lsb:haproxy op monitor interval="1s"
$ crm configure clone haproxy-clone haproxy
$ crm configure colocation vip-with-haproxy inf: vip haproxy-clone
$ crm configure order haproxy-after-vip mandatory: vip haproxy-clone
```

Memcached

Memcached is a general-purpose distributed memory caching system. It is used to speed up dynamic database-driven websites by caching data and objects in RAM to reduce the number of times an external data source must be read.

Memcached is a memory cache demon that can be used by most OpenStack services to store ephemeral data, such as tokens.

Access to Memcached is not handled by HAProxy because replicated access is currently in an experimental state. Instead, OpenStack services must be supplied with the full list of hosts running Memcached.

The Memcached client implements hashing to balance objects among the instances. Failure of an instance impacts only a percentage of the objects and the client automatically removes it from the list of instances. The SLA is several minutes.

Highly available Identity API

Making the OpenStack Identity service highly available in active and passive mode involves:

- *Prerequisites*
- *Configure OpenStack Identity service*
- *Configure OpenStack services to use the highly available OpenStack Identity*

Prerequisites

Before beginning, ensure you have read the OpenStack Identity service getting started documentation.

Add OpenStack Identity resource to Pacemaker

The following section(s) detail how to add the OpenStack Identity resource to Pacemaker on SUSE and Red Hat.

SUSE

SUSE Enterprise Linux and SUSE-based distributions, such as openSUSE, use a set of OCF agents for controlling OpenStack services.

1. Run the following commands to download the OpenStack Identity resource to Pacemaker:

```
# cd /usr/lib/ocf/resource.d
# mkdir openstack
# cd openstack
# wget https://git.openstack.org/cgit/openstack/openstack-resource-agents/plain/ocf/
↪keystone
# chmod a+rx *
```

2. Add the Pacemaker configuration for the OpenStack Identity resource by running the following command to connect to the Pacemaker cluster:

```
# crm configure
```

3. Add the following cluster resources:

```
clone p_keystone ocf:openstack:keystone \
params config="/etc/keystone/keystone.conf" os_password="secretsecret" os_username=
↪"admin" os_tenant_name="admin" os_auth_url="http://10.0.0.11:5000/v2.0/" \
    op monitor interval="30s" timeout="30s"
```

Note: This configuration creates `p_keystone`, a resource for managing the OpenStack Identity service.

4. Commit your configuration changes from the **`crm configure`** menu with the following command:

```
# commit
```

The **`crm configure`** supports batch input. You may have to copy and paste the above lines into your live Pacemaker configuration, and then make changes as required.

For example, you may enter `edit p_ip_keystone` from the **`crm configure`** menu and edit the resource to match your preferred virtual IP address.

Pacemaker now starts the OpenStack Identity service and its dependent resources on all of your nodes.

Red Hat

For Red Hat Enterprise Linux and Red Hat-based Linux distributions, the following process uses Systemd unit files.

```
# pcs resource create openstack-keystone systemd:openstack-keystone --clone interleave=true
```

Configure OpenStack Identity service

1. Edit the `keystone.conf` file to change the values of the *`bind(2)`* parameters:

```
bind_host = 10.0.0.12
public_bind_host = 10.0.0.12
admin_bind_host = 10.0.0.12
```

The `admin_bind_host` parameter lets you use a private network for admin access.

2. To be sure that all data is highly available, ensure that everything is stored in the MySQL database (which is also highly available):

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

3. If the Identity service will be sending ceilometer notifications and your message bus is configured for high availability, you will need to ensure that the Identity service is correctly configured to use it. For details on how to configure the Identity service for this kind of deployment, see *Messaging service for high availability*.

Configure OpenStack services to use the highly available OpenStack Identity

Your OpenStack services now point their OpenStack Identity configuration to the highly available virtual cluster IP address.

1. For OpenStack Compute, (if your OpenStack Identity service IP address is 10.0.0.11) use the following configuration in the `api-paste.ini` file:

```
auth_host = 10.0.0.11
```

2. Create the OpenStack Identity Endpoint with this IP address.

Note: If you are using both private and public IP addresses, create two virtual IP addresses and define the endpoint. For example:

```
$ openstack endpoint create --region $KEYSTONE_REGION \
  $service-type public http://PUBLIC_VIP:5000/v2.0
$ openstack endpoint create --region $KEYSTONE_REGION \
  $service-type admin http://10.0.0.11:35357/v2.0
$ openstack endpoint create --region $KEYSTONE_REGION \
  $service-type internal http://10.0.0.11:5000/v2.0
```

---

3. If you are using the horizon Dashboard, edit the `local_settings.py` file to include the following:

```
OPENSTACK_HOST = 10.0.0.11
```

Highly available Telemetry

The Telemetry service provides a data collection service and an alarming service.

Telemetry polling agent

The Telemetry polling agent can be configured to partition its polling workload between multiple agents. This enables high availability (HA).

Both the central and the compute agent can run in an HA deployment. This 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 to set in the `ceilometer.conf`, see the coordination section in the OpenStack Configuration Reference.

Note: Only one instance for the central and compute agent service(s) is able to run and function correctly if the `backend_url` option is not set.

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 remaining 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 supports using different configuration files. This is for groups of service instances that are running in parallel. For enabling this configuration, set a value for the `partitioning_group_prefix` option in the polling 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 to avoid samples being 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 the Data processing and pipelines section.

To enable the compute agent to run multiple instances simultaneously with workload partitioning, the `workload_partitioning` option must be set to `True` under the compute section in the `ceilometer.conf` configuration file.

Overview of highly available controllers

OpenStack is a set of services exposed to the end users as HTTP(s) APIs. Additionally, for your own internal usage, OpenStack requires an SQL database server and AMQP broker. The physical servers, where all the components are running, are called controllers. This modular OpenStack architecture allows you to duplicate all the components and run them on different controllers. By making all the components redundant, it is possible to make OpenStack highly available.

In general, we can divide all the OpenStack components into three categories:

- OpenStack APIs: APIs that are HTTP(s) stateless services written in python, easy to duplicate and mostly easy to load balance.
- The SQL relational database server provides stateful type consumed by other components. Supported databases are MySQL, MariaDB, and PostgreSQL. Making the SQL database redundant is complex.
- *Advanced Message Queuing Protocol (AMQP)* provides OpenStack internal stateful communication service.

Common deployment architectures

We recommend two primary architectures for making OpenStack highly available.

The architectures differ in the sets of services managed by the cluster.

Both use a cluster manager, such as Pacemaker or Veritas, to orchestrate the actions of the various services across a set of machines. Because we are focused on FOSS, we refer to these as Pacemaker architectures.

Traditionally, Pacemaker has been positioned as an all-encompassing solution. However, as OpenStack services have matured, they are increasingly able to run in an active/active configuration and gracefully tolerate the disappearance of the APIs on which they depend.

With this in mind, some vendors are restricting Pacemaker's use to services that must operate in an active/passive mode (such as `cinder-volume`), those with multiple states (for example, Galera), and those with complex bootstrapping procedures (such as RabbitMQ).

The majority of services, needing no real orchestration, are handled by systemd on each node. This approach avoids the need to coordinate service upgrades or location changes with the cluster and has the added advantage of more easily scaling beyond Corosync's 16 node limit. However, it will generally require the addition of an enterprise monitoring solution such as Nagios or Sensu for those wanting centralized failure reporting.

Configuring the networking services

Run Networking DHCP agent

The OpenStack Networking (neutron) service has a scheduler that lets you run multiple agents across nodes. The DHCP agent can be natively highly available.

To configure the number of DHCP agents per network, modify the `dhcp_agents_per_network` parameter in the `/etc/neutron/neutron.conf` file. By default this is set to 1. To achieve high availability, assign more than one DHCP agent per network. For more information, see High-availability for DHCP.

Run Networking L3 agent

The Networking (neutron) service L3 agent is scalable, due to the scheduler that supports Virtual Router Redundancy Protocol (VRRP) to distribute virtual routers across multiple nodes. For more information about the VRRP and keepalived, see Linux bridge: High availability using VRRP and Open vSwitch: High availability using VRRP.

To enable high availability for configured routers, edit the `/etc/neutron/neutron.conf` file to set the following values:

Table 1: /etc/neutron/neutron.conf parameters for high availability

Parameter	Value	Description
l3_ha	True	All routers are highly available by default.
allow_automatic_l3agent_failover	True	Set automatic L3 agent failover for routers
max_l3_agents_per_router	2 or more	Maximum number of network nodes to use for the HA router.
min_l3_agents_per_router	2 or more	Minimum number of network nodes to use for the HA router. A new router can be created only if this number of network nodes are available.

Configure networking on each node. See the basic information about configuring networking in the *Networking service* section of the Install Tutorials and Guides, depending on your distribution.

OpenStack network nodes contain:

- *Networking DHCP agent*
- *Neutron L3 agent*
- Networking L2 agent

Note: The L2 agent cannot be distributed and highly available. Instead, it must be installed on each data forwarding node to control the virtual network driver such as Open vSwitch or Linux Bridge. One L2 agent runs per node and controls its virtual interfaces.

Note: For Liberty, you can not have the standalone network nodes. The Networking services are run on the controller nodes. In this guide, the term *network nodes* is used for convenience.

Configuring storage

Highly available Image API

The OpenStack Image service offers a service for discovering, registering, and retrieving virtual machine images. To make the OpenStack Image API service highly available in active/passive mode, you must:

- *Add OpenStack Image API resource to Pacemaker*
- *Configure OpenStack Image service API*
- *Configure OpenStack services to use the highly available OpenStack Image API*

Prerequisites

Before beginning, ensure that you are familiar with the documentation for installing the OpenStack Image API service. See the *Image service* section in the Installation Tutorials and Guides, depending on your distribution.

Add OpenStack Image API resource to Pacemaker

1. Download the resource agent to your system:

```
# cd /usr/lib/ocf/resource.d/openstack
# wget https://git.openstack.org/cgit/openstack/openstack-resource-agents/plain/ocf/
↪glance-api
# chmod a+rx *
```

2. Add the Pacemaker configuration for the OpenStack Image API resource. Use the following command to connect to the Pacemaker cluster:

```
crm configure
```

Note: The **`crm configure`** command supports batch input. Copy and paste the lines in the next step into your live Pacemaker configuration and then make changes as required.

For example, you may enter `edit p_ip_glance-api` from the **`crm configure`** menu and edit the resource to match your preferred virtual IP address.

---

3. Add the following cluster resources:

```
primitive p_glance-api ocf:openstack:glance-api \
  params config="/etc/glance/glance-api.conf" \
  os_password="secretsecret" \
  os_username="admin" os_tenant_name="admin" \
  os_auth_url="http://10.0.0.11:5000/v2.0/" \
  op monitor interval="30s" timeout="30s"
```

This configuration creates `p_glance-api`, a resource for managing the OpenStack Image API service.

4. Commit your configuration changes by entering the following command from the **`crm configure`** menu:

```
commit
```

Pacemaker then starts the OpenStack Image API service and its dependent resources on one of your nodes.

Configure OpenStack Image service API

Edit the `/etc/glance/glance-api.conf` file to configure the OpenStack Image service:

```
# We have to use MySQL connection to store data:
sql_connection=mysql://glance:password@10.0.0.11/glance
# Alternatively, you can switch to pymysql,
# a new Python 3 compatible library and use
# sql_connection=mysql+pymysql://glance:password@10.0.0.11/glance
# and be ready when everything moves to Python 3.
# Ref: https://wiki.openstack.org/wiki/PyMySQL_evaluation

# We bind OpenStack Image API to the VIP:
bind_host = 10.0.0.11

# Connect to OpenStack Image registry service:
```

```
registry_host = 10.0.0.11

# We send notifications to High Available RabbitMQ:
notifier_strategy = rabbit
rabbit_host = 10.0.0.11
```

[TODO: need more discussion of these parameters]

Configure OpenStack services to use the highly available OpenStack Image API

Your OpenStack services must now point their OpenStack Image API configuration to the highly available, virtual cluster IP address instead of pointing to the physical IP address of an OpenStack Image API server as you would in a non-HA cluster.

For example, if your OpenStack Image API service IP address is 10.0.0.11 (as in the configuration explained here), you would use the following configuration in your `nova.conf` file:

```
[glance]
# ...
api_servers = 10.0.0.11
# ...
```

You must also create the OpenStack Image API endpoint with this IP address. If you are using both private and public IP addresses, create two virtual IP addresses and define your endpoint. For example:

```
$ openstack endpoint create --region $KEYSTONE_REGION \
  image public http://PUBLIC_VIP:9292

$ openstack endpoint create --region $KEYSTONE_REGION \
  image admin http://10.0.0.11:9292

$ openstack endpoint create --region $KEYSTONE_REGION \
  image internal http://10.0.0.11:9292
```

Highly available Block Storage API

Cinder provides Block-Storage-as-a-Service suitable for performance sensitive scenarios such as databases, expandable file systems, or providing a server with access to raw block level storage.

Persistent block storage can survive instance termination and can also be moved across instances like any external storage device. Cinder also has volume snapshots capability for backing up the volumes.

Making the Block Storage API service highly available in active/passive mode involves:

- *Add Block Storage API resource to Pacemaker*
- *Configure Block Storage API service*
- *Configure OpenStack services to use the highly available Block Storage API*

In theory, you can run the Block Storage service as active/active. However, because of sufficient concerns, we recommend running the volume component as active/passive only.

You can read more about these concerns on the Red Hat Bugzilla and there is a psuedo roadmap for addressing them upstream.

Add Block Storage API resource to Pacemaker

On RHEL-based systems, create resources for cinder's systemd agents and create constraints to enforce startup/shutdown ordering:

```
pcs resource create openstack-cinder-api systemd:openstack-cinder-api --clone
↪interleave=true
pcs resource create openstack-cinder-scheduler systemd:openstack-cinder-scheduler --clone
↪interleave=true
pcs resource create openstack-cinder-volume systemd:openstack-cinder-volume

pcs constraint order start openstack-cinder-api-clone then openstack-cinder-scheduler-clone
pcs constraint colocation add openstack-cinder-scheduler-clone with openstack-cinder-api-
↪clone
pcs constraint order start openstack-cinder-scheduler-clone then openstack-cinder-volume
pcs constraint colocation add openstack-cinder-volume with openstack-cinder-scheduler-clone
```

If the Block Storage service runs on the same nodes as the other services, then it is advisable to also include:

```
pcs constraint order start openstack-keystone-clone then openstack-cinder-api-clone
```

Alternatively, instead of using systemd agents, download and install the OCF resource agent:

```
# cd /usr/lib/ocf/resource.d/openstack
# wget https://git.openstack.org/cgit/openstack/openstack-resource-agents/plain/ocf/cinder-
↪api
# chmod a+rx *
```

You can now add the Pacemaker configuration for Block Storage API resource. Connect to the Pacemaker cluster with the **`crm configure`** command and add the following cluster resources:

```
primitive p_cinder-api ocf:openstack:cinder-api \
   params config="/etc/cinder/cinder.conf" \
   os_password="secretsecret" \
   os_username="admin" \
   os_tenant_name="admin" \
   keystone_get_token_url="http://10.0.0.11:5000/v2.0/tokens" \
   op monitor interval="30s" timeout="30s"
```

This configuration creates `p_cinder-api`, a resource for managing the Block Storage API service.

The command **`crm configure`** supports batch input, copy and paste the lines above into your live Pacemaker configuration and then make changes as required. For example, you may enter `edit p_ip_cinder-api` from the **`crm configure`** menu and edit the resource to match your preferred virtual IP address.

Once completed, commit your configuration changes by entering **`commit`** from the **`crm configure`** menu. Pacemaker then starts the Block Storage API service and its dependent resources on one of your nodes.

Configure Block Storage API service

Edit the `/etc/cinder/cinder.conf` file. For example, on a RHEL-based system:

```
[DEFAULT]
# This is the name which we should advertise ourselves as and for
# A/P installations it should be the same everywhere
host = cinder-cluster-1
```

```

# Listen on the Block Storage VIP
osapi_volume_listen = 10.0.0.11

auth_strategy = keystone
control_exchange = cinder

volume_driver = cinder.volume.drivers.nfs.NfsDriver
nfs_shares_config = /etc/cinder/nfs_exports
nfs_sparsed_volumes = true
nfs_mount_options = v3

[database]
connection = mysql://cinder:CINDER_DBPASS@10.0.0.11/cinder
max_retries = -1

[keystone_authtoken]
# 10.0.0.11 is the Keystone VIP
identity_uri = http://10.0.0.11:35357/
auth_uri = http://10.0.0.11:5000/
admin_tenant_name = service
admin_user = cinder
admin_password = CINDER_PASS

[oslo_messaging_rabbit]
# Explicitly list the rabbit hosts as it doesn't play well with HAProxy
rabbit_hosts = 10.0.0.12,10.0.0.13,10.0.0.14
# As a consequence, we also need HA queues
rabbit_ha_queues = True
heartbeat_timeout_threshold = 60
heartbeat_rate = 2
```

Replace `CINDER_DBPASS` with the password you chose for the Block Storage database. Replace `CINDER_PASS` with the password you chose for the `cinder` user in the Identity service.

This example assumes that you are using NFS for the physical storage, which will almost never be true in a production installation.

If you are using the Block Storage service OCF agent, some settings will be filled in for you, resulting in a shorter configuration file:

```
# We have to use MySQL connection to store data:
connection = mysql://cinder:CINDER_DBPASS@10.0.0.11/cinder
# Alternatively, you can switch to pymysql,
# a new Python 3 compatible library and use
# connection = mysql+pymysql://cinder:CINDER_DBPASS@10.0.0.11/cinder
# and be ready when everything moves to Python 3.
# Ref: https://wiki.openstack.org/wiki/PyMySQL_evaluation

# We bind Block Storage API to the VIP:
osapi_volume_listen = 10.0.0.11

# We send notifications to High Available RabbitMQ:
notifier_strategy = rabbit
rabbit_host = 10.0.0.11
```

Replace `CINDER_DBPASS` with the password you chose for the Block Storage database.

Configure OpenStack services to use the highly available Block Storage API

Your OpenStack services must now point their Block Storage API configuration to the highly available, virtual cluster IP address rather than a Block Storage API server's physical IP address as you would for a non-HA environment.

Create the Block Storage API endpoint with this IP.

If you are using both private and public IP addresses, create two virtual IPs and define your endpoint. For example:

```
$ openstack endpoint create volume --region $KEYSTONE_REGION \
  --publicurl 'http://PUBLIC_VIP:8776/v1/%(tenant_id)s' \
  --adminurl 'http://10.0.0.11:8776/v1/%(tenant_id)s' \
  --internalurl 'http://10.0.0.11:8776/v1/%(tenant_id)s'
```

Highly available Shared File Systems API

Making the Shared File Systems (manila) API service highly available in active/passive mode involves:

- *Add Shared File Systems API resource to Pacemaker*
- *Configure Shared File Systems API service*
- *Configure OpenStack services to use HA Shared File Systems API*

Add Shared File Systems API resource to Pacemaker

1. Download the resource agent to your system:

```
# cd /usr/lib/ocf/resource.d/openstack
# wget https://git.openstack.org/cgit/openstack/openstack-resource-agents/plain/ocf/
↪manila-api
# chmod a+rx *
```

2. Add the Pacemaker configuration for the Shared File Systems API resource. Connect to the Pacemaker cluster with the following command:

```
# crm configure
```

Note: The **`crm configure`** supports batch input. Copy and paste the lines in the next step into your live Pacemaker configuration and then make changes as required.

For example, you may enter `edit p_ip_manila-api` from the **`crm configure`** menu and edit the resource to match your preferred virtual IP address.

---

3. Add the following cluster resources:

```
primitive p_manila-api ocf:openstack:manila-api \
  params config="/etc/manila/manila.conf" \
  os_password="secretsecret" \
  os_username="admin" \
  os_tenant_name="admin" \
```

```
  keystone_get_token_url="http://10.0.0.11:5000/v2.0/tokens" \
  op monitor interval="30s" timeout="30s"
```

This configuration creates `p_manila-api`, a resource for managing the Shared File Systems API service.

4. Commit your configuration changes by entering the following command from the **`crm configure`** menu:

```
# commit
```

Pacemaker now starts the Shared File Systems API service and its dependent resources on one of your nodes.

Configure Shared File Systems API service

Edit the `/etc/manila/manila.conf` file:

```
# We have to use MySQL connection to store data:
sql_connection = mysql+pymysql://manila:password@10.0.0.11/manila?charset=utf8

# We bind Shared File Systems API to the VIP:
osapi_volume_listen = 10.0.0.11

# We send notifications to High Available RabbitMQ:
notifier_strategy = rabbit
rabbit_host = 10.0.0.11
```

Configure OpenStack services to use HA Shared File Systems API

Your OpenStack services must now point their Shared File Systems API configuration to the highly available, virtual cluster IP address rather than a Shared File Systems API server's physical IP address as you would for a non-HA environment.

You must create the Shared File Systems API endpoint with this IP.

If you are using both private and public IP addresses, you should create two virtual IPs and define your endpoints like this:

```
$ openstack endpoint create --region RegionOne \
  sharev2 public 'http://PUBLIC_VIP:8786/v2/%(tenant_id)s'

$ openstack endpoint create --region RegionOne \
  sharev2 internal 'http://10.0.0.11:8786/v2/%(tenant_id)s'

$ openstack endpoint create --region RegionOne \
  sharev2 admin 'http://10.0.0.11:8786/v2/%(tenant_id)s'
```

Storage back end

An OpenStack environment includes multiple data pools for the VMs:

- Ephemeral storage is allocated for an instance and is deleted when the instance is deleted. The Compute service manages ephemeral storage and by default, Compute stores ephemeral drives as files on

local disks on the compute node. As an alternative, you can use Ceph RBD as the storage back end for ephemeral storage.

- Persistent storage exists outside all instances. Two types of persistent storage are provided:
  - The Block Storage service (cinder) that can use LVM or Ceph RBD as the storage back end.
  - The Image service (glance) that can use the Object Storage service (swift) or Ceph RBD as the storage back end.

For more information about configuring storage back ends for the different storage options, see Manage volumes in the OpenStack Administrator Guide.

This section discusses ways to protect against data loss in your OpenStack environment.

RAID drives

Configuring RAID on the hard drives that implement storage protects your data against a hard drive failure. If the node itself fails, data may be lost. In particular, all volumes stored on an LVM node can be lost.

Ceph

Ceph RBD is an innately high availability storage back end. It creates a storage cluster with multiple nodes that communicate with each other to replicate and redistribute data dynamically. A Ceph RBD storage cluster provides a single shared set of storage nodes that can handle all classes of persistent and ephemeral data (glance, cinder, and nova) that are required for OpenStack instances.

Ceph RBD provides object replication capabilities by storing Block Storage volumes as Ceph RBD objects. Ceph RBD ensures that each replica of an object is stored on a different node. This means that your volumes are protected against hard drive and node failures, or even the failure of the data center itself.

When Ceph RBD is used for ephemeral volumes as well as block and image storage, it supports live migration of VMs with ephemeral drives. LVM only supports live migration of volume-backed VMs.

Making the Block Storage (cinder) API service highly available in active/active mode involves:

- Configuring Block Storage to listen on the VIP address
- Managing the Block Storage API daemon with the Pacemaker cluster manager
- Configuring OpenStack services to use this IP address

Configuring the compute node

The Installation Tutorials and Guides provide instructions for installing multiple compute nodes. To make the compute nodes highly available, you must configure the environment to include multiple instances of the API and other services.

Configuring high availability for instances

As of September 2016, the OpenStack High Availability community is designing and developing an official and unified way to provide high availability for instances. We are developing automatic recovery from failures of hardware or hypervisor-related software on the compute node, or other failures that could prevent instances from functioning correctly, such as, issues with a cinder volume I/O path.

More details are available in the user story co-authored by OpenStack's HA community and Product Working Group (PWG), where this feature is identified as missing functionality in OpenStack, which should be addressed with high priority.

Existing solutions

The architectural challenges of instance HA and several currently existing solutions were presented in a talk at the Austin summit, for which slides are also available.

The code for three of these solutions can be found online at the following links:

- a mistral-based auto-recovery workflow, by Intel
- masakari, by NTT
- OCF RAs, as used by Red Hat and SUSE

Current upstream work

Work is in progress on a unified approach, which combines the best aspects of existing upstream solutions. More details are available on the HA VMs user story wiki.

To get involved with this work, see the section on the *HA community*.

Appendix

HA community

The OpenStack HA community holds weekly IRC meetings to discuss a range of topics relating to HA in OpenStack. Everyone interested is encouraged to attend. The logs of all previous meetings are available to read.

You can contact the HA community directly in the #openstack-ha channel on Freenode IRC, or by sending mail to the openstack-dev or openstack-docs mailing list with the `[HA]` prefix in the `Subject` header.

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, join our IRC channel `#openstack-doc` on the freenode IRC network, or report a bug.

The following books explain how to install an OpenStack cloud and its associated components:

- Installation Tutorial for openSUSE Leap 42.2 and SUSE Linux Enterprise Server 12 SP2

- Installation Tutorial for Red Hat Enterprise Linux 7 and CentOS 7
- Installation Tutorial for Ubuntu 16.04 (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:

- End User Guide
- Command-Line Interface Reference

The following documentation provides reference and guidance information for the OpenStack APIs:

- API Guide

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 Ask OpenStack 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 the general OpenStack mailing list. 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. 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), mIRC (Windows), 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 Paste. 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 on the IRC page on the wiki.

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 (ACL) 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.

Active Directory Authentication and identity service by Microsoft, based on LDAP. Supported in OpenStack.

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/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.

Address Resolution Protocol (ARP) The protocol by which layer-3 IP addresses are resolved into layer-2 link local addresses.

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).

admin server In the context of the Identity service, the worker process that provides access to the admin API.

administrator The person responsible for installing, configuring, and managing an OpenStack cloud.

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 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.

aodh Part of the OpenStack *Telemetry service*; provides alarming functionality.

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 Catalog service (murano) The 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.

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 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.

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 CloudFormation template AWS CloudFormation allows Amazon Web Services (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, locally mounted file system, RADOS block devices, VMware datastore, and HTTP.

Backup, Restore, and Disaster Recovery service (freezer) The project that provides integrated tooling for backing up, restoring, and recovering file systems, instances, or database backups.

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 Manager service*.

bare An Image service container format that indicates that no container exists for the VM image.

Bare Metal service (ironic) The OpenStack service that provides a service and associated libraries capable of managing and provisioning physical machines in a security-aware and fault-tolerant manner.

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 (rally) OpenStack project that provides a framework for performance analysis and benchmarking of individual OpenStack components as well as full production OpenStack cloud deployments.

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 API An API on a separate endpoint for attaching, detaching, and creating block storage for compute VMs.

Block Storage service (cinder) The OpenStack service that implement services and libraries to provide on-demand, self-service access to Block Storage resources via abstraction and automation on top of other block storage devices.

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

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.

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 Part of the OpenStack *Telemetry service*; gathers and stores metrics from other OpenStack services.

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 (CA) 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. In OpenStack, a simple certificate authority is 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 Codename for *Block Storage service*.

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 Auditing Data Federation (CADF) Cloud Auditing Data Federation (CADF) is a specification for audit event data. CADF is supported by OpenStack Identity.

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 (senlin) The project that implements clustering services and libraries for the management of groups of homogeneous objects exposed by other OpenStack services.

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.

Common Libraries (oslo) The project that produces a set of python libraries containing code shared by OpenStack projects. The APIs provided by these libraries should be high quality, stable, consistent, documented and generally applicable.

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 API (Nova 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 (nova) The OpenStack core project that implements services and associated libraries to provide massively-scalable, on-demand, self-service access to compute resources, including bare metal, virtual machines, and containers.

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.

congress Code name for the *Governance service*.

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 (magnum) The project which provides 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.

CRL A Certificate Revocation List (CRL) in a PKI model is a list of certificates that have been revoked. End entities presenting these certificates should not be trusted.

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 SUSE that aims to provide all necessary services to quickly deploy and manage 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 project.

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.

Dashboard (horizon) OpenStack project which provides an extensible, unified, web-based user interface for all OpenStack services.

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.

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 (sahara) OpenStack project that provides a scalable data-processing stack and associated management interfaces.

data store A database engine supported by the Database service.

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 (trove) An integrated project that provides scalable and reliable Cloud Database-as-a-Service functionality for both relational and non-relational database engines.

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 dashboard.

default project New users are assigned to this project if no project is specified when a user is created.

default token An Identity service token that is not associated with a specific project 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*.

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 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.

Discretionary Access Control (DAC) 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.

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 record A record that specifies information about a particular domain and belongs to the domain.

DNS service (designate) OpenStack project that provides scalable, on demand, self service access to authoritative DNS services, in a technology-agnostic manner.

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.

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 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. 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.

encapsulation The practice of placing one packet type within another for the purposes of abstracting or securing data. Examples include GRE, MPLS, or IPsec.

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.

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.

FireWall-as-a-Service (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 project 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 Code name for the *Backup, Restore, and Disaster Recovery service*.

front end The point where a user interacts with a service; can be an API endpoint, the 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 Codename for the *Image service*.

glance API server Alternative name for the *Image API*.

glance registry Alternative term for the Image service *image registry*.

global endpoint template The Identity service endpoint template that contains services available to all projects.

GlusterFS A file system designed to aggregate NAS hosts, compatible with OpenStack.

gnocchi Part of the OpenStack *Telemetry service*; provides an indexer and time-series database.

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 (congress) The project that provides Governance-as-a-Service across any collection of cloud services in order to monitor, enforce, and audit policy over dynamic infrastructure.

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.

HAProxy Provides a load balancer for TCP and HTTP-based applications that spreads requests across multiple servers.

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.

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.

heat Codename for the *Orchestration service*.

Heat Orchestration Template (HOT) Heat input in the format native to OpenStack.

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 Codename for the *Dashboard*.

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.

I

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.

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 (keystone) The project that facilitates API client authentication, service discovery, distributed multi-project authorization, and auditing. It provides a central directory of users mapped to the OpenStack services they can access. It also registers endpoints for OpenStack services and acts as a common authentication system.

Identity service API The API used to access the OpenStack Identity service provided through keystone.

IETF Internet Engineering Task Force (IETF) is an open standards organization that develops Internet standards, particularly the standards pertaining to TCP/IP.

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. 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.

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 projects that can access a given VM image within Image service.

image owner The project who owns an Image service virtual machine image.

image registry A list of VM images that are available through Image service.

Image service (glance) The OpenStack service that provide services and associated libraries to store, browse, share, distribute and manage bootable disk images, other data closely associated with initializing compute resources, and metadata definitions.

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, locally mounted file system, RADOS block devices, VMware datastore, 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.

Infrastructure Optimization service (watcher) OpenStack project that aims to provide a flexible and scalable resource optimization service for multi-project OpenStack-based clouds.

Infrastructure-as-a-Service (IaaS) 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.

ingress filtering The process of filtering incoming network traffic. Supported by Compute.

INI format 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.

Input/Output Operations Per Second (IOPS) IOPS are a common performance measurement used to benchmark computer storage devices like hard disk drives, solid state drives, and storage area networks.

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.

Intelligent Platform Management Interface (IPMI) 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.

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 Control Message Protocol (ICMP) A network protocol used by network devices for control messages. For example, **`ping`** uses ICMP to test connectivity.

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. Supported by Compute, Object Storage, and Image service.

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.

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.

ironic Codename for the *Bare Metal service*.

iSCSI Qualified Name (IQN) 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'.

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.

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 (barbican) The project that produces a secret storage and generation system capable of providing key management for services wishing to enable encryption features.

keystone Codename of the *Identity service*.

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 kilogram 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 (L2) agent OpenStack Networking agent that provides layer-2 connectivity for virtual networks.

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 (L3) agent OpenStack Networking agent that provides layer-3 (routing) services for virtual networks.

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.

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 Unix-like computer operating system assembled under the model of free and open-source software development and distribution.

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.

Load-Balancer-as-a-Service (LBaaS) Enables Networking to distribute incoming requests evenly between designated instances.

Load-balancing service (octavia) The project that aims to provide scalable, on demand, self service access to load-balancer services, in technology-agnostic manner.

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 *Containers Infrastructure Management 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 Codename for OpenStack *Shared File Systems service*.

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 project. Enables images to be shared with specified projects.

membership list A list of projects 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 (zaqar) The project that provides a 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.

Meta-Data Server (MDS) Stores CephFS metadata.

Metadata agent OpenStack Networking agent that provides metadata services for instances.

migration The process of moving a VM instance from one host to another.

mistral Code name for *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.

Modular Layer 2 (ML2) neutron plug-in Can concurrently use multiple layer-2 networking technologies, such as 802.1Q and VXLAN, in Networking.

monasca Codename for OpenStack *Monitoring*.

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 (monasca) The OpenStack service that provides a multi-project, highly scalable, performant, fault-tolerant monitoring-as-a-service solution for metrics, complex event processing and logging. To build an extensible platform for advanced monitoring services that can be used by both operators and projects to gain operational insight and visibility, ensuring availability and stability.

multi-factor authentication Authentication method that uses two or more credentials, such as a password and a private key. Currently not supported in Identity.

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 Codename for the *Application Catalog service*.

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.

Network Address Translation (NAT) 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.

Network Service Header (NSH) Provides a mechanism for metadata exchange along the instantiated service path.

Network Time Protocol (NTP) 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 API (Neutron API) API used to access OpenStack Networking. Provides an extensible architecture to enable custom plug-in creation.

Networking service (neutron) The OpenStack project which implements services and associated libraries to provide on-demand, scalable, and technology-agnostic network abstraction.

neutron Codename for OpenStack *Networking service*.

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.

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.

Nexenta volume driver Provides support for NexentaStor devices in Compute.

NFV Orchestration Service (tacker) OpenStack service that aims to implement Network Function Virtualization (NFV) orchestration services and libraries for end-to-end life-cycle management of network services and Virtual Network Functions (VNFs).

Nginx An HTTP and reverse proxy server, a mail proxy server, and a generic TCP/UDP proxy server.

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 Codename for OpenStack *Compute service*.

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 Unique 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 API API used to access OpenStack *Object Storage*.

Object Storage Device (OSD) The Ceph storage daemon.

Object Storage service (swift) The OpenStack core project that provides eventually consistent and redundant storage and retrieval of fixed digital content.

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.

Octavia Code name for the *Load-balancing service*.

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, Mitaka, Newton, Ocata, Pike, Queens, and Rocky. 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 (heat) The OpenStack service which orchestrates composite cloud applications using a declarative template format through an OpenStack-native REST API.

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 Codename for the *Common Libraries project*.

P

panko Part of the OpenStack *Telemetry service*; provides event storage.

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.

Pike The code name for the sixteenth release of OpenStack. The design summit will take place in Boston, Massachusetts, US. The release is named after the Massachusetts Turnpike, abbreviated commonly as the Mass Pike, which is the easternmost stretch of Interstate 90.

Platform-as-a-Service (PaaS) Provides to the consumer an operating system and, often, a language runtime and libraries (collectively, the "platform") upon which they can run their own application code, without providing any control over the underlying infrastructure. Examples of Platform-as-a-Service providers include Cloud Foundry and OpenShift.

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.

policy-based routing (PBR) Provides a mechanism to implement packet forwarding and routing according to the policies defined by the network administrator.

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 projects.

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 Unique ID assigned to each project by the Identity service.

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 projects.

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.

Queens The code name for the seventeenth release of OpenStack. The design summit will take place in Sydney, Australia. The release is named after the Queens Pound river in the South Coast region of New South Wales.

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 Codename for 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.

Rocky The code name for the eighteenth release of OpenStack. The design summit will take place in Vancouver, Kanada. The release is named after the Rocky Mountains.

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 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 dashboard.

role ID Alphanumeric ID assigned to each Identity service role.

Root Cause Analysis (RCA) service (Vitrage) OpenStack project that aims to organize, analyze and visualize OpenStack alarms and events, yield insights regarding the root cause of problems and deduce their existence before they are directly detected.

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

sahara Codename for the *Data Processing service*.

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 project.

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 Code name for the *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 Function Chain (SFC) For a given service, SFC is the abstracted view of the required service functions and the order in which they are to be applied.

service ID Unique ID assigned to each service that is available in the Identity service catalog.

Service Level Agreement (SLA) Contractual obligations that ensure the availability of a service.

service project Special project that contains all services that are listed in the 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 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 service*. 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 service* 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 (manila) The service that provides a set of services for management of shared file systems in a multi-project cloud environment, 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.

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.

Simple Protocol for Independent Computing Environments (SPICE) SPICE provides remote desktop access to guest virtual machines. It is an alternative to VNC. SPICE is supported 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.

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 (solum) 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.

Software-defined networking (SDN) Provides an approach for network administrators to manage computer network services through abstraction of lower-level functionality.

SolidFire Volume Driver The Block Storage driver for the SolidFire iSCSI storage appliance.

solum Code name for the *Software Development Lifecycle Automation service*.

spread-first scheduler The Compute VM scheduling algorithm that attempts to start a new VM on the host with the least amount of load.

SQLAlchemy 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 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 node An Object Storage node that provides container services, account services, and object services; controls the account databases, container databases, and object storage.

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 The VM instance is paused and its state is saved to disk of the host.

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 Codename for OpenStack *Object Storage service*.

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

tacker Code name for the *NFV Orchestration service*

Telemetry service (telemetry) The OpenStack project which collects measurements of the utilization of the physical and virtual resources comprising deployed clouds, persists this data for subsequent retrieval and analysis, and triggers actions when defined criteria are met.

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 projects.

tenant endpoint An Identity service API endpoint that is associated with one or more projects.

tenant ID An alternative term for *project ID*.

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 project 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 Codename for OpenStack *Database service*.

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.

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.

Virtual Extensible LAN (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 address (VIP) 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 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 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 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.

Vitrage Code name for the *Root Cause Analysis service*.

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

Watcher Code name for the *Infrastructure Optimization service*.

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.

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.

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 (mistral) The OpenStack service 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.

X

X.509 X.509 is the most widely used standard for defining digital certificates. It is a data structure that contains the subject (entity) identifiable information such as its name along with its public key. The certificate can contain a few other attributes as well depending upon the version. The most recent and standard version of X.509 is v3.

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 Codename for the *Message service*.

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.

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z