SUSE OpenStack Cloud Crowbar

OpenStack User Guide

OpenStack User Guide

SUSE OpenStack Cloud Crowbar

Publication Date: 09/08/2022

SUSE LLC
1800 South Novell Place
Provo, UT 84606
USA
https://documentation.suse.com

Contents

1 Glance User Guide

1.1 Image Identifiers

Images are uniquely identified by way of a URI that matches the following signature:

```
<Glance Server Location>/v1/images/<ID>
```

where `<Glance Server Location>` is the resource location of the Glance service that knows about an image, and `<ID>` is the image's identifier. Image identifiers in Glance are *uuids*, making them *globally unique*.

1.2 Image Statuses

Images in Glance can be in one the following statuses:

- `queued`
  The image identifier has been reserved for an image in the Glance registry. No image data has been uploaded to Glance and the image size was not explicitly set to zero on creation.
- `saving`
  Denotes that an image's raw data is currently being uploaded to Glance. When an image is registered with a call to `POST /images` and there is an `x-image-meta-location` header present, that image will never be in the `saving` status (as the image data is already available in some other location).
- `active`
  Denotes an image that is fully available in Glance. This occurs when the image data is uploaded, or the image size is explicitly set to zero on creation.
- `deactivated`
  Denotes that access to image data is not allowed to any non-admin user. Prohibiting downloads of an image also prohibits operations like image export and image cloning that may require image data.
- `killed`
  Denotes that an error occurred during the uploading of an image's data, and that the image is not readable.

- `deleted`
  Glance has retained the information about the image, but it is no longer available to use. An image in this state will be removed automatically at a later date.

- `pending_delete`
  This is similar to `deleted`, however, Glance has not yet removed the image data. An image in this state is not recoverable.

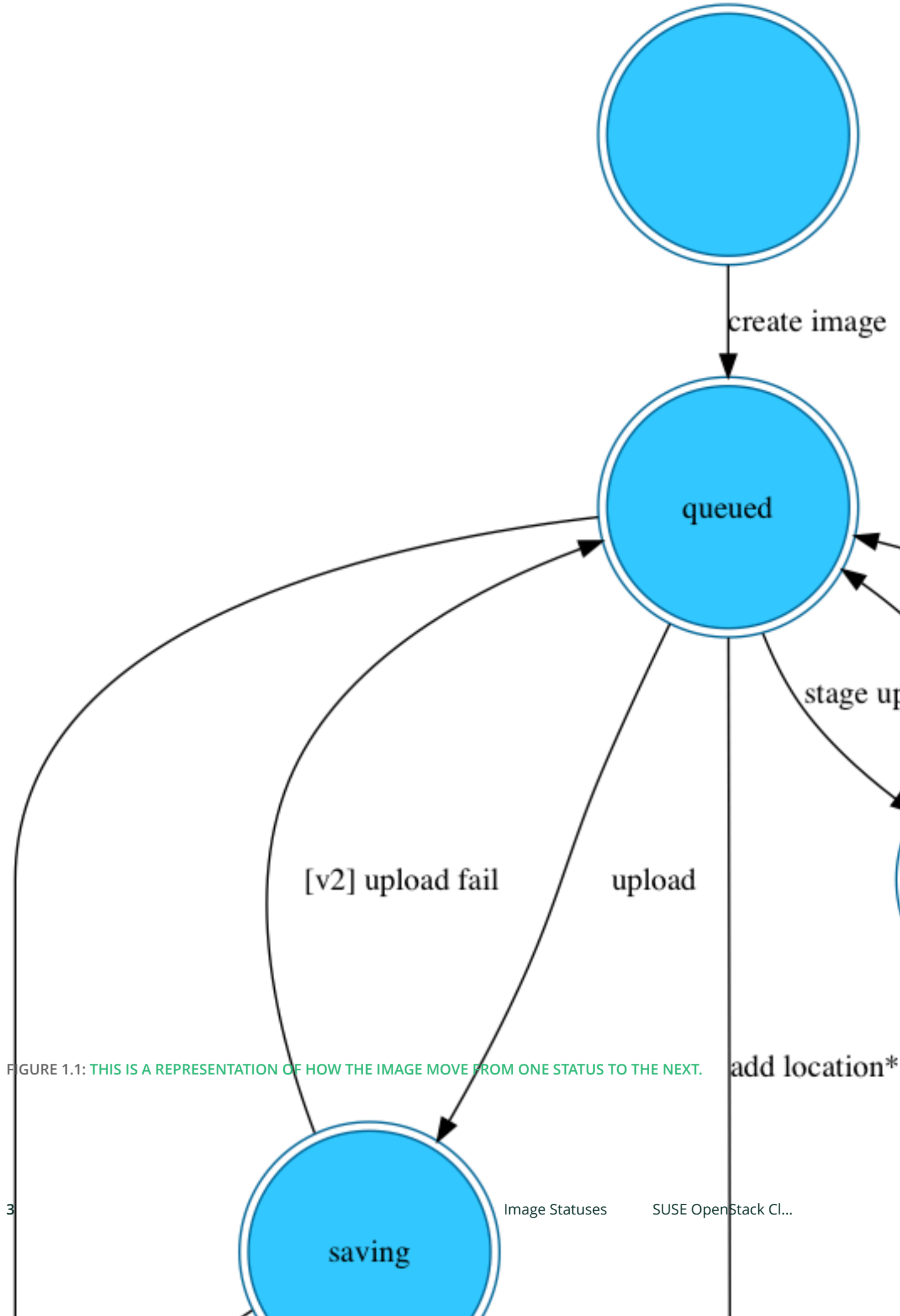


FIGURE 1.1: THIS IS A REPRESENTATION OF HOW THE IMAGE MOVE FROM ONE STATUS TO THE NEXT.

1.3 Task Statuses

Tasks in Glance can be in one the following statuses:

- `pending`
  The task identifier has been reserved for a task in the Glance. No processing has begun on it yet.
- `processing`
  The task has been picked up by the underlying executor and is being run using the backend Glance execution logic for that task type.
- `success`
  Denotes that the task has had a successful run within Glance. The `result` field of the task shows more details about the outcome.
- `failure`
  Denotes that an error occurred during the execution of the task and it cannot continue processing. The `message` field of the task shows what the error was.

1.4 Image Statuses

Images in Glance can be in one the following statuses:

- `queued`
  The image identifier has been reserved for an image in the Glance registry. No image data has been uploaded to Glance and the image size was not explicitly set to zero on creation.
- `saving`
  Denotes that an image's raw data is currently being uploaded to Glance. When an image is registered with a call to `POST /images` and there is an `x-image-meta-location` header present, that image will never be in the `saving` status (as the image data is already available in some other location).
- `active`
  Denotes an image that is fully available in Glance. This occurs when the image data is uploaded, or the image size is explicitly set to zero on creation.
- `deactivated`

Denotes that access to image data is not allowed to any non-admin user. Prohibiting downloads of an image also prohibits operations like image export and image cloning that may require image data.

- `killed`
  Denotes that an error occurred during the uploading of an image's data, and that the image is not readable.

- `deleted`
  Glance has retained the information about the image, but it is no longer available to use. An image in this state will be removed automatically at a later date.

- `pending_delete`
  This is similar to `deleted`, however, Glance has not yet removed the image data. An image in this state is not recoverable.

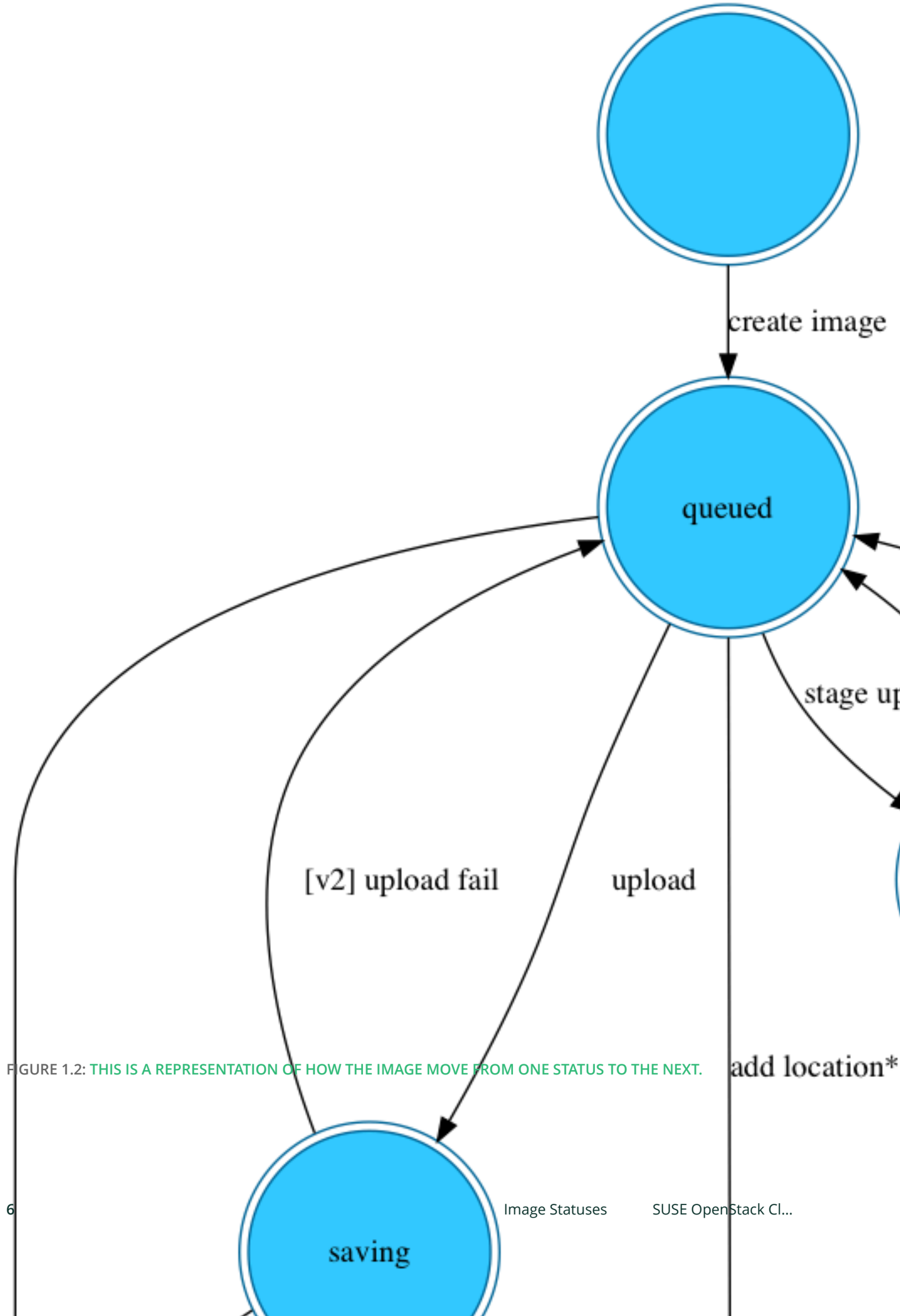


FIGURE 1.2: THIS IS A REPRESENTATION OF HOW THE IMAGE MOVE FROM ONE STATUS TO THE NEXT.

1.5 Task Statuses

Tasks in Glance can be in one the following statuses:

- `pending`
  The task identifier has been reserved for a task in the Glance. No processing has begun on it yet.
- `processing`
  The task has been picked up by the underlying executor and is being run using the backend Glance execution logic for that task type.
- `success`
  Denotes that the task has had a successful run within Glance. The `result` field of the task shows more details about the outcome.
- `failure`
  Denotes that an error occurred during the execution of the task and it cannot continue processing. The `message` field of the task shows what the error was.

1.6 Disk and Container Formats

When adding an image to Glance, you must specify what the virtual machine image's *disk format* and *container format* are. Disk and container formats are configurable on a per-deployment basis. This document intends to establish a global convention for what specific values of *disk_format* and *container_format* mean.

1.6.1 Disk Format

The disk format of a virtual machine image is the format of the underlying disk image. Virtual appliance vendors have different formats for laying out the information contained in a virtual machine disk image.

You can set your image's disk format to one of the following:

- **raw**
  This is an unstructured disk image format
- **vhd**

This is the VHD disk format, a common disk format used by virtual machine monitors from VMware, Xen, Microsoft, VirtualBox, and others.

- **vhdx**
  This is the VHDX disk format, an enhanced version of the VHD format which supports larger disk sizes among other features.
- **vmdk**
  Another common disk format supported by many common virtual machine monitors.
- **vdi**
  A disk format supported by VirtualBox virtual machine monitor and the QEMU emulator.
- **iso**
  An archive format for the data contents of an optical disc (For example, CDROM).
- **ploop**
  A disk format supported and used by Virtuozzo to run OS Containers.
- **qcow2**
  A disk format supported by the QEMU emulator that can expand dynamically and supports Copy on Write.
- **aki**
  This indicates what is stored in Glance is an Amazon kernel image.
- **ari**
  This indicates what is stored in Glance is an Amazon ramdisk image.
- **ami**
  This indicates what is stored in Glance is an Amazon machine image.

1.6.2 Container Format

The container format refers to whether the virtual machine image is in a file format that also contains metadata about the actual virtual machine.

Note

The container format string is not currently used by Glance or other OpenStack components, so it is safe to simply specify **bare** as the container format if you are unsure.

You can set your image's container format to one of the following:

- **bare**
  This indicates there is no container or metadata envelope for the image.
- **ovf**
  This is the OVF container format.
- **aki**
  This indicates what is stored in Glance is an Amazon kernel image.
- **ari**
  This indicates what is stored in Glance is an Amazon ramdisk image.
- **ami**
  This indicates what is stored in Glance is an Amazon machine image
- **ova**
  This indicates what is stored in Glance is an OVA tar archive file.
- **docker**
  This indicates what is stored in Glance is a Docker tar archive of the container filesystem.

1.7 Common Image Properties

When adding an image to Glance, you may specify some common image properties that may prove useful to consumers of your image.

This document explains the names of these properties and the expected values.

The common image properties are also described in a JSON schema, found in `/etc/glance/schema-image.json` in the Glance source code.

1.7.1 architecture

Operating system architecture as specified in https://docs.openstack.org/python-glanceclient/latest/cli/property-keys.html.

1.7.2 instance_uuid

Metadata which can be used to record which instance this image is associated with. (Informational only, does not create an instance snapshot.)

1.7.3 kernel_id

The ID of image stored in Glance that should be used as the kernel when booting an AMI-style image.

1.7.4 ramdisk_id

The ID of image stored in Glance that should be used as the ramdisk when booting an AMI-style image.

1.7.5 os_distro

The common name of the operating system distribution as specified in https://docs.openstack.org/python-glanceclient/latest/cli/property-keys.html.

1.7.6 os_version

The operating system version as specified by the distributor.

1.8 Metadata Definition Concepts

The metadata definition service was added to Glance in the Juno release of OpenStack.

It provides a common API for vendors, admins, services, and users to meaningfully **define** available key and value pair metadata that can be used on different types of resources (images, artifacts, volumes, flavors, aggregates, and other resources). A definition includes a property's key, its description, its constraints, and the resource types to which it can be associated.

This catalog does not store the values for specific instance properties.

For example, a definition of a virtual CPU topology property for the number of cores will include the base key to use (for example, `cpu_cores`), a description, and value constraints like requiring it to be an integer. So, a user, potentially through Horizon, would be able to search this catalog to list the available properties they can add to a flavor or image. They will see the virtual CPU topology property in the list and know that it must be an integer.

When the user adds the property its key and value will be stored in the service that owns that resource (for example, Nova for flavors and in Glance for images). The catalog also includes any additional prefix required when the property is applied to different types of resources, such as `hw_` for images and `hw:` for flavors. So, on an image, the user would know to set the property as `hw_cpu_cores=1`.

1.8.1 Terminology

1.8.1.1 Background

The term *metadata* can become very overloaded and confusing. This catalog is about the additional metadata that is passed as arbitrary key and value pairs or tags across various artifacts and OpenStack services.

Below are a few examples of the various terms used for metadata across OpenStack services today:

Nova	Cinder	Glance
Flavor • *extra specs* Host Aggregate	Volume & Snapshot • *properties* • *tags*	Image & Snapshot

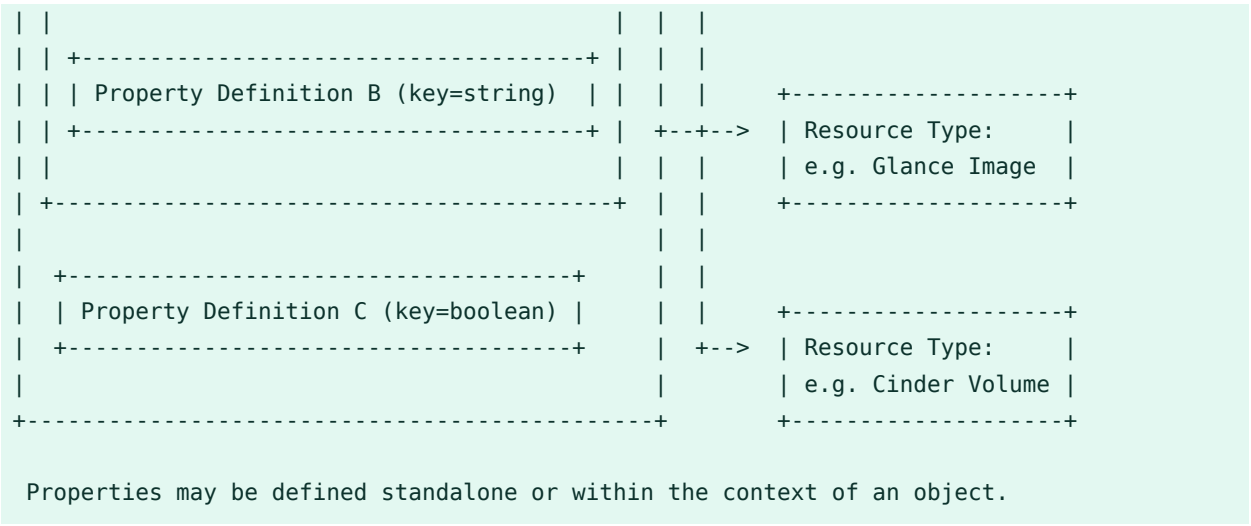
Nova	Cinder	Glance
Servers		• *metadata* • *image metadata*
		• *metadata* • *metadata*
	Volume-Type	• *scheduler_hints* • *tags* • *extra specs* • *qos specs*

1.8.1.2 Catalog Concepts

The below figure illustrates the concept terminology used in the metadata definitions catalog:

```
A namespace is associated with 0 to many resource types, making it visible to
the API and UI for applying to that type of resource. RBAC Permissions are
managed at a namespace level.

+----------------------------------------------+
|         Namespace                            |
|                                              |
| +------------------------------------------+ |
| |        Object Definition                 | |
| |                                          | |        +--------------------+
| | +--------------------------------------+ | |  +-->  | Resource Type:     |
| | | Property Definition A (key=integer) | | |  |     | e.g. Nova Flavor   |
| | +--------------------------------------+ | |  |     +--------------------+
```



1.8.1.3 Catalog Terminology

The following terminology is used within the metadata definition catalog.

Namespaces

Metadata definitions are contained in namespaces.

- Specify the access controls (CRUD) for everything defined in it. Allows for admin only, different projects, or the entire cloud to define and use the definitions in the namespace.
- Associates the contained definitions to different types of resources.

Properties

A property describes a single property and its primitive constraints. Each property can only be a primitive type:

- string, integer, number, boolean, array

Each primitive type is described using simple JSON schema notation. This means no nested objects and no definition referencing.

Objects

An object describes a group of one to many properties and their primitive constraints. Each property in the group can ONLY be a primitive type:

- string, integer, number, boolean, array

Each primitive type is described using simple JSON schema notation. This means no nested objects.

The object may optionally define required properties under the semantic understanding that a user who uses the object should provide all required properties.

Resource Type Association

Resource type association specifies the relationship between resource types and the namespaces that are applicable to them. This information can be used to drive UI and CLI views. For example, the same namespace of objects, properties, and tags may be used for images, snapshots, volumes, and flavors. Or a namespace may only apply to images.

Resource types should be aligned with Heat resource types whenever possible. http://docs.openstack.org/developer/heat/template_guide/openstack.html

It is important to note that the same base property key can require different prefixes depending on the target resource type. The API provides a way to retrieve the correct property based on the target resource type.

Below are a few examples:

The desired virtual CPU topology can be set on both images and flavors via metadata. The keys have different prefixes on images than on flavors. On flavors keys are prefixed with `hw:`, but on images the keys are prefixed with `hw_`.

For more: https://github.com/openstack/nova-specs/blob/master/specs/juno/implemented/virt-driver-vcpu-topology.rst.

Another example is the `AggregateInstanceExtraSpecsFilter` and scoped properties (For example, properties with `something:something=value`). For scoped or namespaced properties, the `AggregateInstanceExtraSpecsFilter` requires a prefix of `aggregate_instance_extra_specs:` to be used on flavors but not on the aggregate itself. Otherwise, the filter will not evaluate the property during scheduling.

So, on a host aggregate, you may see:

`companyx:fastio=true`

But then when used on the flavor, the `AggregateInstanceExtraSpecsFilter` needs:

`aggregate_instance_extra_specs:companyx:fastio=true`

In some cases, there may be multiple different filters that may use the same property with different prefixes. In this case, the correct prefix needs to be set based on which filter is enabled.

1.9 Using Glance's Image Public APIs

Glance is the reference implementation of the OpenStack Images API. As such, Glance fully implements versions 1 and 2 of the Images API.

Note

The Images API v1 has been DEPRECATED in the Newton release. The migration path is to use the Images API v2 (http://developer.openstack.org/api-ref/image/v2/) instead of version 1 of the API. The Images API v1 will ultimately be removed, following the OpenStack standard deprecation policy (https://governance.openstack.org/reference/tags/assert_follows-standard-deprecation.html).

There used to be a sentence here saying, "The Images API specification is developed alongside Glance, but is not considered part of the Glance project." That's only partially true (or completely false, depending upon how strict you are about these things). Conceptually, the OpenStack Images API is an independent definition of a REST API. In practice, however, the only way to participate in the evolution of the Images API is to work with the Glance community to define the new functionality and provide its reference implementation. Further, Glance falls under the designated sections provision of the OpenStack DefCore Guidelines, which basically means that in order to qualify as OpenStack, a cloud exposing an OpenStack Images API must include the Glance Images API implementation code. Thus, although conceptually independent, the OpenStack Images APIs are intimately associated with Glance.

References

- Designated sections (definition) (http://git.openstack.org/cgit/openstack/defcore/tree/doc/source/process/Lexicon.rst#n54)
- 2014-04-02 DefCore Designated Sections Guidelines (https://governance.openstack.org/resolutions/20140402-defcore-designated-sections-guidelines.html)
- OpenStack Core Definition (https://github.com/openstack/defcore/blob/master/doc/source/process/CoreDefinition.rst)
- DefCore Guidelines Repository (https://github.com/openstack/defcore)

1.9.1 Glance and the Images APIs: Past, Present, and Future

Here's a quick summary of the Images APIs that have been implemented by Glance. If you're interested in more details, you can consult the Release Notes for all the OpenStack releases (beginning with "Bexar") to follow the evolution of features in Glance and the Images APIs.

1.9.1.1 Images v1 API

The v1 API was originally designed as a service API for use by Nova and other OpenStack services. In the Kilo release, the v1.1 API was downgraded from CURRENT to SUPPORTED. In the Newton release, the version 1 API is officially declared DEPRECATED.

During the deprecation period, the Images v1 API is closed to further development. The Glance code implementing the v1 API accepts only serious bug fixes.

Since Folsom, it has been possible to deploy OpenStack without exposing the Images v1 API to end users. The Compute v2 API contains image-related API calls allowing users to list images, list images details, show image details for a specific image, delete images, and manipulate image metadata. Nova acts as a proxy to Glance for these image-related calls. It's important to note that the image-related calls in the Compute v2 API are a proper subset of the calls available in the Images APIs.

In the Newton release, Nova (and other OpenStack services that consume images) have been modified to use the Images v2 API by default.

Reference

- OpenStack Standard Deprecation Requirements (https://governance.openstack.org/reference/tags/assert_follows-standard-deprecation.html#requirements)

1.9.1.2 Images v2 API

The v2 API is the CURRENT OpenStack Images API. It provides a more friendly interface to consumers than did the v1 API, as it was specifically designed to expose images-related functionality as a public-facing endpoint. It's the version that's currently open to development.

A common strategy is to deploy multiple Glance nodes: internal-facing nodes providing the Images APIs for internal consumers like Nova, and external-facing nodes providing the Images v2 API for public use.

1.9.1.3 The Future

During the long and tumultuous design phase of what has since become an independent service named “Glare” (the Glance Artifacts Repository), the Glance community loosely spoke about the Artifacts API being “Glance v3”. This, however, was only a shorthand way of speaking of the Artifacts effort. The Artifacts API can’t be the Images v3 API since Artifacts are not the same as Images. Conceptually, a virtual machine image could be an Artifact, and the Glare code has been designed to be compatible with the Images v2 API. But at this time, there are no plans to implement an Images v3 API.

During the Newton development cycle, Glare became an independent OpenStack project. While it’s evident that there’s a need for an Artifact Repository in OpenStack, whether it will be as ubiquitous as the need for an Images Repository isn’t clear. On the other hand, industry trends could go in the opposite direction where everyone needs Artifacts and deployers view images as simply another type of digital artifact.

1.9.2 Authentication

Glance depends on Keystone and the OpenStack Identity API to handle authentication of clients. You must obtain an authentication token from Keystone using and send it along with all API requests to Glance through the `X-Auth-Token` header. Glance will communicate back to Keystone to verify the token validity and obtain your identity credentials.

See Authentication With Keystone (https://docs.openstack.org/glance/pike/admin/authentication.html#authentication) for more information on integrating with Keystone.

1.9.3 Using v1.X

Note

The Images API v1 has been DEPRECATED in the Newton release. The migration path is to use the Images API v2 (http://developer.openstack.org/api-ref/image/v2/) instead of version 1 of the API. The Images API v1 will ultimately be removed, following the OpenStack standard deprecation policy (https://governance.openstack.org/reference/tags/assert_follows-standard-deprecation.html).

For the purpose of examples, assume there is a Glance API server running at the URL `http://glance.openstack.example.org` on the default port 80.

1.9.3.1 List Available Images

We want to see a list of available images that the authenticated user has access to. This includes images owned by the user, images shared with the user and public images.

We issue a `GET` request to `http://glance.openstack.example.org/v1/images` to retrieve this list of available images. The data is returned as a JSON-encoded mapping in the following format:

```
{'images': [
  {'uri': 'http://glance.openstack.example.org/v1/images/71c675ab-d94f-49cd-a114-
e12490b328d9',
   'name': 'SLES 15',
   'disk_format': 'vhd',
   'container_format': 'ovf',
   'size': '5368709120'}
  ...]}
```

1.9.3.2 List Available Images in More Detail

We want to see a more detailed list of available images that the authenticated user has access to. This includes images owned by the user, images shared with the user and public images.

We issue a `GET` request to `http://glance.openstack.example.org/v1/images/detail` to retrieve this list of available images. The data is returned as a JSON-encoded mapping in the following format:

```
{'images': [
  {'uri': 'http://glance.openstack.example.org/v1/images/71c675ab-d94f-49cd-a114-
e12490b328d9',
   'name': 'SLES 15',
   'disk_format': 'vhd',
   'container_format': 'ovf',
   'size': '5368709120',
   'checksum': 'c2e5db72bd7fd153f53ede5da5a06de3',
   'created_at': '2010-02-03 09:34:01',
   'updated_at': '2010-02-03 09:34:01',
   'deleted_at': '',
   'status': 'active',
   'is_public': true,
```

```
  'min_ram': 256,
  'min_disk': 5,
  'owner': null,
  'properties': {'distro': 'SLES 15'}},
 ...]}
```

Note

All timestamps returned are in UTC.

The `updated_at` timestamp is the timestamp when an image's metadata was last updated, not its image data, as all image data is immutable once stored in Glance.

The `properties` field is a mapping of free-form key and value pairs that have been saved with the image metadata.

The `checksum` field is an MD5 checksum of the image file data.

The `is_public` field is a boolean indicating whether the image is publicly available.

The `min_ram` field is an integer specifying the minimum amount of RAM needed to run this image on an instance in megabytes.

The `min_disk` field is an integer specifying the minimum amount of disk space needed to run this image on an instance in gigabytes.

The `owner` field is a string which may either be null or which will indicate the owner of the image.

1.9.3.3 Filtering Images Lists

Both the `GET /v1/images` and `GET /v1/images/detail` requests take query parameters that serve to filter the returned list of images. The following list details these query parameters.

- `name=NAME`
  Filters images having a `name` attribute matching `NAME`.
- `container_format=FORMAT`
  Filters images having a `container_format` attribute matching `FORMAT`.
  For more information, see *Section 1.6, "Disk and Container Formats"*.
- `disk_format=FORMAT`
  Filters images having a `disk_format` attribute matching `FORMAT`.
  For more information, see *Section 1.6, "Disk and Container Formats"*.

- `status=STATUS`
  Filters images having a `status` attribute matching `STATUS`.
  For more information, see *Section 1.2, "Image Statuses"*.

- `size_min=BYTES`
  Filters images having a `size` attribute greater than or equal to `BYTES`.

- `size_max=BYTES`
  Filters images having a `size` attribute less than or equal to `BYTES`.

These two resources also accept additional query parameters:

- `sort_key=KEY`
  Results will be ordered by the specified image attribute `KEY`. Accepted values include `id`, `name`, `status`, `disk_format`, `container_format`, `size`, `created_at` (default) and `updated_at`.

- `sort_dir=DIR`
  Results will be sorted in the direction `DIR`. Accepted values are `asc` for ascending or `desc` (default) for descending.

- `marker=ID`
  An image identifier marker may be specified. When present, only images which occur after the identifier `ID` will be listed. (These are the images that have a `sort_key` later than that of the marker `ID` in the `sort_dir` direction.)

- `limit=LIMIT`
  When present, the maximum number of results returned will not exceed `LIMIT`.

Note

If the specified `LIMIT` exceeds the operator defined limit (`api_limit_max`) then the number of results returned may be less than `LIMIT`.

- `is_public=PUBLIC`
  An admin user may use the `is_public` parameter to control which results are returned. When the `is_public` parameter is absent or set to `True` the following images will be listed: Images whose `is_public` field is `True`, owned images and shared images.
  When the `is_public` parameter is set to `False` the following images will be listed: Images (owned, shared, or non-owned) whose `is_public` field is `False`.

When the `is_public` parameter is set to `None` all images will be listed irrespective of owner, shared status or the `is_public` field.

Note

Use of the `is_public` parameter is restricted to admin users. For all other users it will be ignored.

1.9.3.4 Retrieve Image Metadata

We want to see detailed information for a specific virtual machine image that the Glance server knows about.

We have queried the Glance server for a list of images and the data returned includes the `uri` field for each available image. This `uri` field value contains the exact location needed to get the metadata for a specific image.

Continuing the example from above, in order to get metadata about the first image returned, we can issue a `HEAD` request to the Glance server for the image's URI.

We issue a `HEAD` request to `http://glance.openstack.example.org/v1/images/71c675ab-d94f-49cd-a114-e12490b328d9` to retrieve complete metadata for that image. The metadata is returned as a set of HTTP headers that begin with the prefix `x-image-meta-`. The following shows an example of the HTTP headers returned from the above `HEAD` request:

```
x-image-meta-uri              http://glance.openstack.example.org/v1/images/71c675ab-
d94f-49cd-a114-e12490b328d9
x-image-meta-name             SLES 15
x-image-meta-disk_format      vhd
x-image-meta-container_format ovf
x-image-meta-size             5368709120
x-image-meta-checksum         c2e5db72bd7fd153f53ede5da5a06de3
x-image-meta-created_at       2010-02-03 09:34:01
x-image-meta-updated_at       2010-02-03 09:34:01
x-image-meta-deleted_at
x-image-meta-status           available
x-image-meta-is_public        true
x-image-meta-min_ram          256
x-image-meta-min_disk         0
x-image-meta-owner            null
x-image-meta-property-distro  SLES 15
```

Note

All timestamps returned are in UTC.

The `x-image-meta-updated_at` timestamp is the timestamp when an image's metadata was last updated, not its image data, as all image data is immutable once stored in Glance.

There may be multiple headers that begin with the prefix `x-image-meta-property-`. These headers are free-form key and value pairs that have been saved with the image metadata. The key is the string after `x-image-meta-property-` and the value is the value of the header.

The response's `ETag` header will always be equal to the `x-image-meta-checksum` value.

The response's `x-image-meta-is_public` value is a boolean indicating whether the image is publicly available.

The response's `x-image-meta-owner` value is a string which may either be null or which will indicate the owner of the image.

1.9.3.5 Retrieve Raw Image Data

We want to retrieve that actual raw data for a specific virtual machine image that the Glance server knows about.

We have queried the Glance server for a list of images and the data returned includes the `uri` field for each available image. This `uri` field value contains the exact location needed to get the metadata for a specific image.

Continuing the example from above, in order to get metadata about the first image returned, we can issue a `HEAD` request to the Glance server for the image's URI.

We issue a `GET` request to `http://glance.openstack.example.org/v1/images/71c675ab-d94f-49cd-a114-e12490b328d9` to retrieve metadata for that image as well as the image itself encoded into the response body.

The metadata is returned as a set of HTTP headers that begin with the prefix `x-image-meta-`. The following shows an example of the HTTP headers returned from the above `GET` request:

```
x-image-meta-uri              http://glance.openstack.example.org/v1/images/71c675ab-
d94f-49cd-a114-e12490b328d9
x-image-meta-name             SLES 15
x-image-meta-disk_format      vhd
x-image-meta-container_format ovf
x-image-meta-size             5368709120
```

```
x-image-meta-checksum          c2e5db72bd7fd153f53ede5da5a06de3
x-image-meta-created_at        2010-02-03 09:34:01
x-image-meta-updated_at        2010-02-03 09:34:01
x-image-meta-deleted_at
x-image-meta-status            available
x-image-meta-is_public         true
x-image-meta-min_ram           256
x-image-meta-min_disk          5
x-image-meta-owner             null
x-image-meta-property-distro   SLES 15
```

Note

All timestamps returned are in UTC.

The `x-image-meta-updated_at` timestamp is the timestamp when an image's metadata was last updated, not its image data, as all image data is immutable once stored in Glance.

There may be multiple headers that begin with the prefix `x-image-meta-property-`. These headers are free-form key and value pairs that have been saved with the image metadata. The key is the string after `x-image-meta-property-` and the value is the value of the header.

The response's `Content-Length` header shall be equal to the value of the `x-image-meta-size` header.

The response's `ETag` header will always be equal to the `x-image-meta-checksum` value.

The response's `x-image-meta-is_public` value is a boolean indicating whether the image is publicly available.

The response's `x-image-meta-owner` value is a string which may either be null or which will indicate the owner of the image.

The image data itself will be the body of the HTTP response returned from the request, which will have content-type of `application/octet-stream`.

1.9.3.6 Add a New Image

We have created a new virtual machine image in some way (created a snapshot or backed up an existing image) and we wish to do two things:

- Store the disk image data in Glance.
- Store metadata about this image in Glance.

We can do the above two activities in a single call to the Glance API. Assuming, like in the examples above, that a Glance API server is running at `http://glance.openstack.example.org`, we issue a `POST` request to add an image to Glance:

```
POST http://glance.openstack.example.org/v1/images
```

The metadata about the image is sent to Glance in HTTP headers. The body of the HTTP request to the Glance API will be the MIME-encoded disk image data.

1.9.3.7 Reserve a New Image

We can also perform the activities described in *Section 1.9.3.6, "Add a New Image"* using two separate calls to the Image API; the first to register the image metadata, and the second to add the image disk data. This is known as reserving an image.

The first call should be a `POST` to `http://glance.openstack.example.org/v1/images`, which will result in a new image id being registered with a status of `queued`:

```
{'image':
 {'status': 'queued',
  'id': '71c675ab-d94f-49cd-a114-e12490b328d9',
  ...}
 ...}
```

The image data can then be added using a `PUT` to `http://glance.openstack.example.org/v1/images/71c675ab-d94f-49cd-a114-e12490b328d9`. The image status will then be set to `active` by Glance.

Image Metadata in HTTP Headers

Glance will view as image metadata any HTTP header that it receives in a `POST` request where the header key is prefixed with the strings `x-image-meta-` and `x-image-meta-property-`.

The list of metadata headers that Glance accepts are listed below.

- `x-image-meta-name`
  This header is required, unless reserving an image. Its value should be the name of the image.

Note

The name of an image *is not unique to a Glance node*. It would be an unrealistic expectation of users to know all the unique names of all other user's images.

- `x-image-meta-id`
  This header is optional.
  When present, Glance will use the supplied identifier for the image. If the identifier already exists in that Glance node, then a **409 Conflict** will be returned by Glance. The value of the header must be a uuid in hexadecimal string notation (that is 71c675ab-d94f-49cd-a114-e12490b328d9).
  When this header is *not* present, Glance will generate an identifier for the image and return this identifier in the response (see below).

- `x-image-meta-store`
  This header is optional. Valid values are one of `file`, `rbd`, `swift`, `cinder`, `sheepdog` or `vsphere`.
  When present, Glance will attempt to store the disk image data in the backing store indicated by the value of the header. If the Glance node does not support the backing store, Glance will return a **400 Bad Request**.
  When not present, Glance will store the disk image data in the backing store that is marked as default. See the configuration option `default_store` for more information.

- `x-image-meta-disk_format`
  This header is required, unless reserving an image. Valid values are one of `aki`, `ari`, `ami`, `raw`, `iso`, `vhd`, `vhdx`, `vdi`, `qcow2`, `vmdk` or `ploop`.
  For more information, see *Section 1.6, "Disk and Container Formats"*.

- `x-image-meta-container_format`
  This header is required, unless reserving an image. Valid values are one of `aki`, `ari`, `ami`, `bare`, `ova`, `ovf`, or `docker`.
  For more information, see *Section 1.6, "Disk and Container Formats"*.

- `x-image-meta-size`
  This header is optional.
  When present, Glance assumes that the expected size of the request body will be the value of this header. If the length in bytes of the request body *does not match* the value of this header, Glance will return a **400 Bad Request**.

When not present, Glance will calculate the image's size based on the size of the request body.

- `x-image-meta-checksum`
  This header is optional. When present, it specifies the **MD5** checksum of the image file data. When present, Glance will verify the checksum generated from the back-end store while storing your image against this value and return a **400 Bad Request** if the values do not match.

- `x-image-meta-is_public`
  This header is optional.
  When Glance finds the string `true` (case-insensitive), the image is marked as a public one, meaning that any user may view its metadata and may read the disk image from Glance.
  When not present, the image is assumed to be *not public* and owned by a user.

- `x-image-meta-min_ram`
  This header is optional. When present, it specifies the minimum amount of RAM in megabytes required to run this image on a server.
  When not present, the image is assumed to have a minimum RAM requirement of 0.

- `x-image-meta-min_disk`
  This header is optional. When present, it specifies the expected minimum disk space in gigabytes required to run this image on a server.
  When not present, the image is assumed to have a minimum disk space requirement of 0.

- `x-image-meta-owner`
  This header is optional and only meaningful for admins.
  Glance normally sets the owner of an image to be the tenant or user (depending on the `owner_is_tenant` configuration option) of the authenticated user issuing the request. However, if the authenticated user has the Admin role, this default may be overridden by setting this header to null or to a string identifying the owner of the image.

- `x-image-meta-property-*`
  When Glance receives any HTTP header whose key begins with the string prefix `x-image-meta-property-`, Glance adds the key and value to a set of custom, free-form image properties stored with the image. The key is a lower-cased string following the prefix `x-image-meta-property-` with dashes and punctuation replaced with underscores.
  There is no limit on the number of free-form key and value attributes that can be attached to the image. However, keep in mind that the 8K limit on the size of all the HTTP headers sent in a request will effectively limit the number of image properties.

1.9.3.8 Update an Image

Glance will consider any HTTP header that it receives in a `PUT` request as an instance of image metadata. In this case, the header key should be prefixed with the strings `x-image-meta-` and `x-image-meta-property-`.

If an image was previously reserved, and thus is in the `queued` state, then image data can be added by including it as the request body. If the image already has data associated with it (for example, it is not in the `queued` state), then including a request body will result in a **409 Conflict** exception.

On success, the `PUT` request will return the image metadata encoded as HTTP headers.

See more about image statuses here *Section 1.2, "Image Statuses"*.

1.9.3.9 List Image Memberships

We want to see a list of the other system tenants (or users, if `owner_is_tenant` is `False`) that may access a given virtual machine image that the Glance server knows about. We take the `uri` field of the image data, append `/members` to it, and issue a `GET` request on the resulting URL.

Continuing from the example above, in order to get the memberships for the first image returned, we can issue a `GET` request to the Glance server for `http://glance.openstack.example.org/v1/images/71c675ab-d94f-49cd-a114-e12490b328d9/members`. And we will get back JSON data such as the following:

```
{'members': [
 {'member_id': 'tenant1',
  'can_share': false}
 ...]}
```

The `member_id` field identifies a tenant with which the image is shared. If that tenant is authorized to further share the image, the `can_share` field is `true`.

1.9.3.10 List Shared Images

We want to see a list of images which are shared with a given tenant. We issue a `GET` request to `http://glance.openstack.example.org/v1/shared-images/tenant1`. We will get back JSON data such as the following:

```
{'shared_images': [
```

```
 {'image_id': '71c675ab-d94f-49cd-a114-e12490b328d9',
  'can_share': false}
 ...]}
```

The `image_id` field identifies an image shared with the tenant named by *member_id*. If the tenant is authorized to further share the image, the `can_share` field is `true`.

1.9.3.11 Add a Member to an Image

We want to authorize a tenant to access a private image. We issue a `PUT` request to `http://glance.openstack.example.org/v1/images/71c675ab-d94f-49cd-a114-e12490b328d9/members/tenant1`. With no body, this will add the membership to the image, leaving existing memberships unmodified and defaulting new memberships to have `can_share` set to `false`. We may also optionally attach a body of the following form:

```
{'member':
 {'can_share': true}
}
```

If such a body is provided, both existing and new memberships will have `can_share` set to the provided value (either `true` or `false`). This query will return a 204 (`No Content`) status code.

1.9.3.12 Remove a Member from an Image

We want to revoke a tenant's right to access a private image. We issue a `DELETE` request to `http://glance.openstack.example.org/v1/images/1/members/tenant1`. This query will return a 204 (`No Content`) status code.

1.9.3.13 Replace a Membership List for an Image

The full membership list for a given image may be replaced. We issue a `PUT` request to `http://glance.openstack.example.org/v1/images/71c675ab-d94f-49cd-a114-e12490b328d9/members` with a body of the following form:

```
{'memberships': [
 {'member_id': 'tenant1',
  'can_share': false}
 ...]}
```

All existing memberships which are not named in the replacement body are removed, and those which are named have their `can_share` settings changed as specified. (The `can_share` setting may be omitted, which will cause that setting to remain unchanged in the existing memberships.) All new memberships will be created, with `can_share` defaulting to `false` unless it is specified otherwise.

1.9.4 Image Membership Changes in Version 2.0

Version 2.0 of the Images API eliminates the `can_share` attribute of image membership. In the version 2.0 model, image sharing is not transitive.

In version 2.0, image members have a `status` attribute that reflects how the image should be treated with respect to that image member's image-list.

- The `status` attribute may have one of three values: `pending`, `accepted`, or `rejected`.
- By default, only those shared images with status `accepted` are included in an image member's image-list.
- Only an image member may change their membership status.
- Only an image owner may create members on an image. The status of a newly created image member is `pending`. The image owner cannot change the status of a member.

1.9.4.1 Distinctions from Version 1.x API Calls

- The response to a request to list the members of an image has changed.
  call: `GET` on `/v2/images/{imageId}/members`
  response: see the JSON schema at `/v2/schemas/members`
- The request body in the call to create an image member has changed.
  call: `POST` to `/v2/images/{imageId}/members`
  request body:

  ```
  { "member": "<MEMBER_ID>" }
  ```

  Where the `{memberId}` is the tenant ID of the image member.
  The member status of a newly created image member is `pending`.

1.9.4.2 New API Calls

- Change the status of an image member
  call: `PUT` on `/v2/images/{imageId}/members/{memberId}`.
  Request body:

  ```
  { "status": "<STATUS_VALUE>" }
  ```

  Where <STATUS_VALUE> is `pending`, `accepted`, or `rejected`. The `{memberId}` is the tenant ID of the image member.

1.9.5 Images v2 Tasks API

Version 2 of the OpenStack Images API introduces a Task resource that is used to create and monitor long-running asynchronous image-related processes. See the Tasks (https://docs.openstack.org/glance/pike/admin/tasks.html#tasks) section of the Glance documentation for more information.

The following Task calls are available:

1.9.5.1 Create a Task

A user wants to initiate a Task. The user issues a `POST` request to `/v2/tasks`. The request body is of Content-type `application/json` and must contain the following fields:

- `type`: A string specified by the enumeration defined in the Task schema.
- `input`: A JSON object. The content is defined by the cloud provider who has exposed the endpoint being contacted.

The response is a Task entity as defined by the Task schema. It includes an `id` field that can be used in a subsequent call to poll the Task for status changes.

A Task is created in `pending` status.

1.9.5.2 Show a Task

A user wants to see detailed information about a Task the user owns. The user issues a `GET` request to `/v2/tasks/{taskId}`.

The response is in `application/json` format. The exact structure is given by the Task schema located at `/v2/schemas/task`.

1.9.5.3 List Tasks

A user wants to see what Tasks have been created in their project. The user issues a `GET` request to `/v2/tasks`.

The response is in `application/json` format. The exact structure is given by the task schema located at `/v2/schemas/tasks`.

Note

As indicated by the schema, the list of Tasks is provided in a sparse format. To see more information about a particular Task in the list, the user would use the show Task call described above.

1.9.5.4 Filtering and Sorting the Tasks List

The `GET /v2/tasks` request takes query parameters that server to filter the returned list of Tasks. The following list details these query parameters.

- `status={status}`
  Filters the list to display only those tasks in the specified status. See the Task schema or the *Section 1.3, "Task Statuses"* section of this documentation for the legal values to use for `{status}`.
  For example, a request to `GET /v2/tasks?status=pending` would return only those Tasks whose current status is `pending`.
- `type={type}`
  Filters the list to display only those Tasks of the specified type. See the enumeration defined in the Task schema for the legal values to use for `{type}`.
  For example, a request to `GET /v2/tasks?type=import` would return only import Tasks.
- `sort_dir={direction}`
  Sorts the list of tasks according to `updated_at` datetime. Legal values are `asc` (ascending) and `desc` (descending). By default, the Task list is sorted by `created_at` time in descending chronological order.

1.9.6 API Message Localization

Glance supports HTTP message localization. For example, an HTTP client can receive API messages in Chinese even if the locale language of the server is English.

1.9.6.1 How to use it

To receive localized API messages, the HTTP client needs to specify the **Accept-Language** header to indicate the language that will translate the message. For more information about Accept-Language, refer to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

A typical curl API request will be like below:

```
curl -i -X GET -H 'Accept-Language: zh' -H 'Content-Type: application/json'
http://glance.openstack.example.org/v2/images/aaa
```

Then the response will be like the following:

```
HTTP/1.1 404 Not Found
Content-Length: 234
Content-Type: text/html; charset=UTF-8
X-Openstack-Request-Id: req-54d403a0-064e-4544-8faf-4aeef086f45a
Date: Sat, 22 Feb 2014 06:26:26 GMT

<html>
<head>
<title>404 Not Found</title>
</head>
<body>
<h1>404 Not Found</h1>
&#25214;&#19981;&#21040;&#20219;&#20309;&#20855;&#26377;&#26631;&#35782; aaa
 &#30340;&#26144;&#20687;<br /><br />
</body>
</html>
```

Note

Make sure to have a language package under `/usr/share/locale-langpack/` on the target Glance server.

1.10 Using Glance's Client Tools

The command-line tool and python library for Glance are both installed through the python-glanceclient project. Explore the following resources for more information:

- Official Docs (http://docs.openstack.org/developer/python-glanceclient/)
- Pypi Page (http://pypi.python.org/pypi/python-glanceclient)
- GitHub Project (http://github.com/openstack/python-glanceclient)

1.11 Using Glance's Metadata Definitions Catalog Public APIs

A common API hosted by the Glance service for vendors, admins, services, and users to meaningfully define available key and value pair and tag metadata. The intent is to enable better metadata collaboration across artifacts, services, and projects for OpenStack users.

This is about the definition of the available metadata that can be used on different types of resources (images, artifacts, volumes, flavors, aggregates, etc). A definition includes the properties type, its key, its description, and its constraints. This catalog will not store the values for specific instance properties.

For example, a definition of a virtual CPU topology property for number of cores will include the key to use, a description, and value constraints like requiring it to be an integer. So, a user, potentially through Horizon, would be able to search this catalog to list the available properties they can add to a flavor or image. They will see the virtual CPU topology property in the list and know that it must be an integer. In the Horizon example, when the user adds the property, its key and value will be stored in the service that owns that resource (Nova for flavors and in Glance for images).

Diagram: https://wiki.openstack.org/w/images/b/bb/Glance-Metadata-API.png

Glance Metadata Definitions Catalog implementation started with API version v2.

1.11.1 Authentication

Glance depends on Keystone and the OpenStack Identity API to handle authentication of clients. You must obtain an authentication token from Keystone send it along with all API requests to Glance through the `X-Auth-Token` header. Glance will communicate back to Keystone to verify the token validity and obtain your identity credentials.

See Authentication With Keystone (https://docs.openstack.org/glance/pike/admin/authentication.html#authentication) for more information on integrating with Keystone.

1.11.2 Using v2.X

For the purpose of examples, assume there is a Glance API server running at the URL `http://glance.openstack.example.org` on the default port 80.

1.11.2.1 List Available Namespaces

We want to see a list of available namespaces that the authenticated user has access to. This includes namespaces owned by the user, namespaces shared with the user and public namespaces.

We issue a `GET` request to `http://glance.openstack.example.org/v2/metadefs/namespaces` to retrieve this list of available namespaces. The data is returned as a JSON-encoded mapping in the following format:

```
{
  "namespaces": [
      {
          "namespace": "MyNamespace",
          "display_name": "My User Friendly Namespace",
          "description": "My description",
          "visibility": "public",
          "protected": true,
          "owner": "The Test Owner",
          "self": "/v2/metadefs/namespaces/MyNamespace",
          "schema": "/v2/schemas/metadefs/namespace",
          "created_at": "2014-08-28T17:13:06Z",
          "updated_at": "2014-08-28T17:13:06Z",
          "resource_type_associations": [
              {
                  "name": "OS::Nova::Aggregate",
                  "created_at": "2014-08-28T17:13:06Z",
                  "updated_at": "2014-08-28T17:13:06Z"
```

```
            },
            {
                "name": "OS::Nova::Flavor",
                "prefix": "aggregate_instance_extra_specs:",
                "created_at": "2014-08-28T17:13:06Z",
                "updated_at": "2014-08-28T17:13:06Z"
            }
        ]
    }
  ],
  "first": "/v2/metadefs/namespaces?sort_key=created_at&sort_dir=asc",
  "schema": "/v2/schemas/metadefs/namespaces"
}
```


Note

Listing namespaces will only show the summary of each namespace including counts and resource type associations. Detailed responses including all its objects definitions, property definitions will only be available on each individual GET namespace request.

1.11.2.2 Filtering Namespaces Lists

`GET /v2/metadefs/namespaces` requests take query parameters that serve to filter the returned list of namespaces. The following list details these query parameters.

- `resource_types=RESOURCE_TYPES`
  Filters namespaces having a `resource_types` within the list of comma separated `RESOURCE_TYPES`.

GET resource also accepts additional query parameters:

- `sort_key=KEY`
  Results will be ordered by the specified sort attribute `KEY`. Accepted values include `namespace`, `created_at` (default) and `updated_at`.
- `sort_dir=DIR`
  Results will be sorted in the direction `DIR`. Accepted values are `asc` for ascending or `desc` (default) for descending.
- `marker=NAMESPACE`

A namespace identifier marker may be specified. When present only namespaces which occur after the identifier `NAMESPACE` will be listed, for example the namespaces which have a `sort_key` later than that of the marker `NAMESPACE` in the `sort_dir` direction.

- `limit=LIMIT`
  When present the maximum number of results returned will not exceed `LIMIT`.

Note

If the specified `LIMIT` exceeds the operator defined limit (`api_limit_max`) then the number of results returned may be less than `LIMIT`.

- `visibility=PUBLIC`
  An admin user may use the `visibility` parameter to control which results are returned (`PRIVATE` or `PUBLIC`).

1.11.2.3 Retrieve Namespace

We want to see a more detailed information about a namespace that the authenticated user has access to. The detail includes the properties, objects, and resource type associations.

We issue a `GET` request to `http://glance.openstack.example.org/v2/metadefs/namespaces/{namespace}` to retrieve the namespace details. The data is returned as a JSON-encoded mapping in the following format:

```
{
  "namespace": "MyNamespace",
  "display_name": "My User Friendly Namespace",
  "description": "My description",
  "visibility": "public",
  "protected": true,
  "owner": "The Test Owner",
  "schema": "/v2/schemas/metadefs/namespace",
  "resource_type_associations": [
      {
          "name": "OS::Glance::Image",
          "prefix": "hw_",
          "created_at": "2014-08-28T17:13:06Z",
          "updated_at": "2014-08-28T17:13:06Z"
      },
      {
          "name": "OS::Cinder::Volume",
```

```
            "prefix": "hw_",
            "properties_target": "image",
            "created_at": "2014-08-28T17:13:06Z",
            "updated_at": "2014-08-28T17:13:06Z"
        },
        {
            "name": "OS::Nova::Flavor",
            "prefix": "filter1:",
            "created_at": "2014-08-28T17:13:06Z",
            "updated_at": "2014-08-28T17:13:06Z"
        }
    ],
    "properties": {
        "nsprop1": {
            "title": "My namespace property1",
            "description": "More info here",
            "type": "boolean",
            "default": true
        },
        "nsprop2": {
            "title": "My namespace property2",
            "description": "More info here",
            "type": "string",
            "default": "value1"
        }
    },
    "objects": [
        {
            "name": "object1",
            "description": "my-description",
            "self": "/v2/metadefs/namespaces/MyNamespace/objects/object1",
            "schema": "/v2/schemas/metadefs/object",
            "created_at": "2014-08-28T17:13:06Z",
            "updated_at": "2014-08-28T17:13:06Z",
            "required": [],
            "properties": {
                "prop1": {
                    "title": "My object1 property1",
                    "description": "More info here",
                    "type": "array",
                    "items": {
                        "type": "string"
                    }
                }
            }
        },
        {
```

```
            "name": "object2",
            "description": "my-description",
            "self": "/v2/metadefs/namespaces/MyNamespace/objects/object2",
            "schema": "/v2/schemas/metadefs/object",
            "created_at": "2014-08-28T17:13:06Z",
            "updated_at": "2014-08-28T17:13:06Z",
            "properties": {
                "prop1": {
                    "title": "My object2 property1",
                    "description": "More info here",
                    "type": "integer",
                    "default": 20
                }
            }
        }
    ]
}
```

1.11.2.4 Retrieve available Resource Types

We want to see the list of all resource types that are available in Glance.

We issue a `GET` request to `http://glance.openstack.example.org/v2/metadefs/resource_types` to retrieve all resource types.

The data is returned as a JSON-encoded mapping in the following format:

```
{
    "resource_types": [
        {
            "created_at": "2014-08-28T17:13:04Z",
            "name": "OS::Glance::Image",
            "updated_at": "2014-08-28T17:13:04Z"
        },
        {
            "created_at": "2014-08-28T17:13:04Z",
            "name": "OS::Cinder::Volume",
            "updated_at": "2014-08-28T17:13:04Z"
        },
        {
            "created_at": "2014-08-28T17:13:04Z",
            "name": "OS::Nova::Flavor",
            "updated_at": "2014-08-28T17:13:04Z"
        },
        {
            "created_at": "2014-08-28T17:13:04Z",
```

```
            "name": "OS::Nova::Aggregate",
            "updated_at": "2014-08-28T17:13:04Z"
        },
        {
            "created_at": "2014-08-28T17:13:04Z",
            "name": "OS::Nova::Server",
            "updated_at": "2014-08-28T17:13:04Z"
        }
    ]
}
```

1.11.2.5 Retrieve Resource Types associated with a Namespace

We want to see the list of resource types that are associated for a specific namespace.

We issue a `GET` request to `http://glance.openstack.example.org/v2/metadefs/namespaces/{namespace}/resource_types` to retrieve resource types.

The data is returned as a JSON-encoded mapping in the following format:

```
{
  "resource_type_associations" : [
      {
         "name" : "OS::Glance::Image",
         "prefix" : "hw_",
         "created_at": "2014-08-28T17:13:04Z",
         "updated_at": "2014-08-28T17:13:04Z"
      },
      {
         "name" :"OS::Cinder::Volume",
         "prefix" : "hw_",
         "properties_target" : "image",
         "created_at": "2014-08-28T17:13:04Z",
         "updated_at": "2014-08-28T17:13:04Z"
      },
      {
         "name" : "OS::Nova::Flavor",
         "prefix" : "hw:",
         "created_at": "2014-08-28T17:13:04Z",
         "updated_at": "2014-08-28T17:13:04Z"
      }
  ]
}
```

1.11.2.6 Add Namespace

We want to create a new namespace that can contain the properties, objects, etc.

We issue a `POST` request to add an namespace to Glance:

```
POST http://glance.openstack.example.org/v2/metadefs/namespaces/
```

The input data is an JSON-encoded mapping in the following format:

```
{
  "namespace": "MyNamespace",
  "display_name": "My User Friendly Namespace",
  "description": "My description",
  "visibility": "public",
  "protected": true
}
```

Note

Optionally properties, objects and resource type associations could be added in the same input. See GET Namespace output above (input will be similar).

1.11.2.7 Update Namespace

We want to update an existing namespace.

We issue a `PUT` request to update an namespace to Glance:

```
PUT http://glance.openstack.example.org/v2/metadefs/namespaces/{namespace}
```

The input data is similar to Add Namespace.

1.11.2.8 Delete Namespace

We want to delete an existing namespace including all its objects, properties etc.

We issue a `DELETE` request to delete an namespace to Glance:

```
DELETE http://glance.openstack.example.org/v2/metadefs/namespaces/{namespace}
```

1.11.2.9 Associate Resource Type with Namespace

We want to associate a resource type with an existing namespace.

We issue a `POST` request to associate resource type to Glance:

```
POST http://glance.openstack.example.org/v2/metadefs/namespaces/{namespace}/
resource_types
```

The input data is an JSON-encoded mapping in the following format:

```
{
        "name" :"OS::Cinder::Volume",
        "prefix" : "hw_",
        "properties_target" : "image",
        "created_at": "2014-08-28T17:13:04Z",
        "updated_at": "2014-08-28T17:13:04Z"
}
```

1.11.2.10 Remove Resource Type associated with a Namespace

We want to de-associate namespace from a resource type.

We issue a `DELETE` request to de-associate namespace resource type to Glance:

```
DELETE http://glance.openstack.example.org/v2//metadefs/namespaces/{namespace}/
resource_types/{resource_type}
```

1.11.2.11 List Objects in Namespace

We want to see the list of meta definition objects in a specific namespace.

We issue a `GET` request to `http://glance.openstack.example.org/v2/metadefs/namespaces/{namespace}/objects` to retrieve objects.

The data is returned as a JSON-encoded mapping in the following format:

```
{
      "objects": [
      {
          "name": "object1",
          "description": "my-description",
          "self": "/v2/metadefs/namespaces/MyNamespace/objects/object1",
          "schema": "/v2/schemas/metadefs/object",
          "created_at": "2014-08-28T17:13:06Z",
          "updated_at": "2014-08-28T17:13:06Z",
          "required": [],
          "properties": {
              "prop1": {
```

```
                "title": "My object1 property1",
                "description": "More info here",
                "type": "array",
                "items": {
                    "type": "string"
                }
            }
        }
    },
    {
        "name": "object2",
        "description": "my-description",
        "self": "/v2/metadefs/namespaces/MyNamespace/objects/object2",
        "schema": "/v2/schemas/metadefs/object",
        "created_at": "2014-08-28T17:13:06Z",
        "updated_at": "2014-08-28T17:13:06Z",
        "properties": {
            "prop1": {
                "title": "My object2 property1",
                "description": "More info here",
                "type": "integer",
                "default": 20
            }
        }
    }
  ],
  "schema": "/v2/schemas/metadefs/objects"
}
```

1.11.2.12 Add object in a specific namespace

We want to create a new object which can group the properties.

We issue a `POST` request to add object to a namespace in Glance:

```
POST http://glance.openstack.example.org/v2/metadefs/namespaces/{namespace}/objects
```

The input data is an JSON-encoded mapping in the following format:

```
{
  "name": "StorageQOS",
  "description": "Our available storage QOS.",
  "required": [
      "minIOPS"
  ],
  "properties": {
```

```
      "minIOPS": {
          "type": "integer",
          "description": "The minimum IOPs required",
          "default": 100,
          "minimum": 100,
          "maximum": 30000369
      },
      "burstIOPS": {
          "type": "integer",
          "description": "The expected burst IOPs",
          "default": 1000,
          "minimum": 100,
          "maximum": 30000377
      }
  }
}
```

1.11.2.13 Update Object in a specific namespace

We want to update an existing object.

We issue a `PUT` request to update an object to Glance:

```
PUT http://glance.openstack.example.org/v2/metadefs/namespaces/{namespace}/objects/
{object_name}
```

The input data is similar to Add Object.

1.11.2.14 Delete Object in a specific namespace

We want to delete an existing object.

We issue a `DELETE` request to delete object in a namespace to Glance:

```
DELETE http://glance.openstack.example.org/v2/metadefs/namespaces/{namespace}/objects/
{object_name}
```

1.11.2.15 Add property definition in a specific namespace

We want to create a new property definition in a namespace.

We issue a `POST` request to add property definition to a namespace in Glance:

```
POST http://glance.openstack.example.org/v2/metadefs/namespaces/{namespace}/properties
```

The input data is an JSON-encoded mapping in the following format:

```
{
  "name": "hypervisor_type",
  "title" : "Hypervisor",
  "type": "array",
  "description": "The type of hypervisor required",
  "items": {
      "type": "string",
      "enum": [
          "hyperv",
          "qemu",
          "kvm"
      ]
  }
}
```

1.11.2.16 Update property definition in a specific namespace

We want to update an existing object.

We issue a `PUT` request to update an property definition in a namespace to Glance:

```
PUT http://glance.openstack.example.org/v2/metadefs/namespaces/{namespace}/properties/
{property_name}
```

The input data is similar to Add property definition.

1.11.2.17 Delete property definition in a specific namespace

We want to delete an existing object.

We issue a `DELETE` request to delete property definition in a namespace to Glance:

```
DELETE http://glance.openstack.example.org/v2/metadefs/namespaces/{namespace}/properties/
{property_name}
```

1.11.3 API Message Localization

Glance supports HTTP message localization. For example, an HTTP client can receive API messages in Chinese even if the locale language of the server is English.

1.11.3.1 How to use it

To receive localized API messages, the HTTP client needs to specify the **Accept-Language** header to indicate the language to use to translate the message. For more info about Accept-Language, refer http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

A typical curl API request will be like below:

```
curl -i -X GET -H 'Accept-Language: zh' -H 'Content-Type: application/json'
http://glance.openstack.example.org/v2/metadefs/namespaces/{namespace}
```

Then the response will be like the following:

```
HTTP/1.1 404 Not Found
Content-Length: 234
Content-Type: text/html; charset=UTF-8
X-Openstack-Request-Id: req-54d403a0-064e-4544-8faf-4aeef086f45a
Date: Sat, 22 Feb 2014 06:26:26 GMT

<html>
<head>
<title>404 Not Found</title>
</head>
<body>
<h1>404 Not Found</h1>
&#25214;&#19981;&#21040;&#20219;&#20309;&#20855;&#26377;&#26631;&#35782; aaa
 &#30340;&#26144;&#20687;<br /><br />
</body>
</html>
```

Note

Ensure there is the language package under `/usr/share/locale-langpack/` on the target Glance server.

1.12 Image Signature Verification

Glance has the ability to perform image validation using a digital signature and asymmetric cryptography. To trigger this, you must define specific image properties (described below), and have stored a certificate signed with your private key in a local Barbican installation.

When the image properties exist on an image, Glance will validate the uploaded image data against these properties before storing it. If validation is unsuccessful, the upload will fail and the image will be deleted.

Additionally, the image properties may be used by other services (for example, Nova) to perform data verification when the image is downloaded from Glance.

1.12.1 Requirements

Barbican key manager - See http://docs.openstack.org/developer/barbican/setup/devstack.html.

1.12.2 Configuration

The `etc/glance-api.conf` can be modified to change keystone endpoint of barbican. By default barbican will try to connect to keystone at http://localhost:5000/v3 but if keystone is on another host then this should be changed.

In `glance-api.conf` find the following lines:

```
[barbican]
auth_endpoint = http://localhost:5000/v3
```

Then replace http://localhost:5000/v3 with the URL of keystone, also adding `/v3` to the end of it. For example, 'https://192.168.245.9:5000/v3'.

Another option in `etc/glance-api.conf` which can be configured is which key manager to use. By default Glance will use the default key manager defined by the Castellan key manager interface, which is currently the Barbican key manager.

In `glance-api.conf` find the following lines:

```
[key_manager]
api_class = castellan.key_manager.barbican_key_manager.BarbicanKeyManager
```

Then replace the value with the desired key manager class.

Note

If those lines do not exist then simply add them to the end of the file.

1.12.3 Using the Signature Verification

An image will need a few properties for signature verification to be enabled, these are:

```
img_signature
img_signature_hash_method
img_signature_key_type
img_signature_certificate_uuid
```

1.12.3.1 Property img_signature

This is the signature of your image.

Note

The max character limit is 255.

1.12.3.2 Property img_signature_hash_method

Hash methods is the method you hash with.

Current ones you can use are:

- SHA-224
- SHA-256
- SHA-384
- SHA-512

1.12.3.3 Property img_signature_key_type

This is the key_types you can use for your image.

Current ones you can use are:

- RSA-PSS
- DSA
- ECC-CURVES

- SECT571K1
- SECT409K1
- SECT571R1
- SECT409R1
- SECP521R1
- SECP384R1

Note

ECC curves - Only keysizes above 384 are included. Not all ECC curves may be supported by the back end.

1.12.3.4 Property img_signature_certificate_uuid

This is the UUID of the certificate that you upload to Barbican.

Therefore the type passed to glance is:

- UUID

Note

The supported certificate types are:

- X_509

1.12.4 Example Usage

Follow these instructions to create your keys:

```
$ openssl genrsa -out private_key.pem 1024
Generating RSA private key, 1024 bit long modulus
.....................................++++++
..++++++
e is 65537 (0x10001)

$ openssl rsa -pubout -in private_key.pem -out public_key.pem
```

```
writing RSA key

$ openssl req -new -key private_key.pem -out cert_request.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.

$ openssl x509 -req -days 14 -in cert_request.csr -signkey private_key.pem -out
 new_cert.crt
Signature ok
subject=/C=AU/ST=Some-State/O=Internet Widgits Pty Ltd
Getting Private key
```

Upload your certificate. This only has to be done once as you can use the same `Secret href` for many images until it expires:

```
$ openstack secret store --name test --algorithm RSA --expiration 2016-06-29 --secret-
type certificate --payload-content-type "application/octet-stream" --payload-content-
encoding base64 --payload "$(base64 new_cert.crt)"
+---------------+------------------------------------------------------------------------+
| Field         | Value                                                                  |
+---------------+------------------------------------------------------------------------+
| Secret href   | http://127.0.0.1:9311/v1/secrets/cd7cc675-e573-419c-8fff-33a72734a243 |

$ cert_uuid=cd7cc675-e573-419c-8fff-33a72734a243
```

Get an image and create the signature:

```
$ echo This is a dodgy image > myimage

$ openssl dgst -sha256 -sign private_key.pem -sigopt rsa_padding_mode:pss -out
 myimage.signature myimage

$ base64 -w 0 myimage.signature > myimage.signature.b64

$ image_signature=$(cat myimage.signature.b64)
```

Note

Using Glance v1 requires `-w 0` due to not supporting multiline image properties. Glance v2 does support multiline image properties and does not require `-w 0` but may still be used.

Create the image:

```
$ glance image-create --name mySignedImage --container-format bare
 --disk-format qcow2 --property img_signature="$image_signature"
```

```
--property img_signature_certificate_uuid="$cert_uuid" --property
img_signature_hash_method='SHA-256' --property img_signature_key_type='RSA-PSS' <
myimage
```

Note

Creating the image can fail if validation does not succeed. This will cause the image to be deleted.

1.12.5 Other Links

- https://etherpad.openstack.org/p/mitaka-glance-image-signing-instructions
- http://docs.openstack.org/ops-guide/ops_user_facing_operations.html

2 Ironic User Guide

Ironic is an OpenStack project which provisions bare metal (as opposed to virtual) machines. It may be used independently or as part of an OpenStack Cloud, and integrates with the OpenStack Identity (keystone), Compute (nova), Network (neutron), Image (glance) and Object (swift) services.

When the Bare Metal service is appropriately configured with the Compute and Network services, it is possible to provision both virtual and physical machines through the Compute service's API. However, the set of instance actions is limited, arising from the different characteristics of physical servers and switch hardware. For example, live migration can not be performed on a bare metal instance.

The community maintains reference drivers that leverage open-source technologies (for example, PXE and IPMI) to cover a wide range of hardware. Ironic's pluggable driver architecture also allows hardware vendors to write and contribute drivers that may improve performance or add functionality not provided by the community drivers.

2.1 Why Provision Bare Metal

The following are a few use-cases for bare metal (physical server) provisioning in cloud:

- High-performance computing clusters
- Computing tasks that require access to hardware devices which can't be virtualized
- Database hosting (some databases run poorly in a hypervisor)
- Single tenant, dedicated hardware for performance, security, dependability and other regulatory requirements
- Rapidly deploying a cloud infrastructure

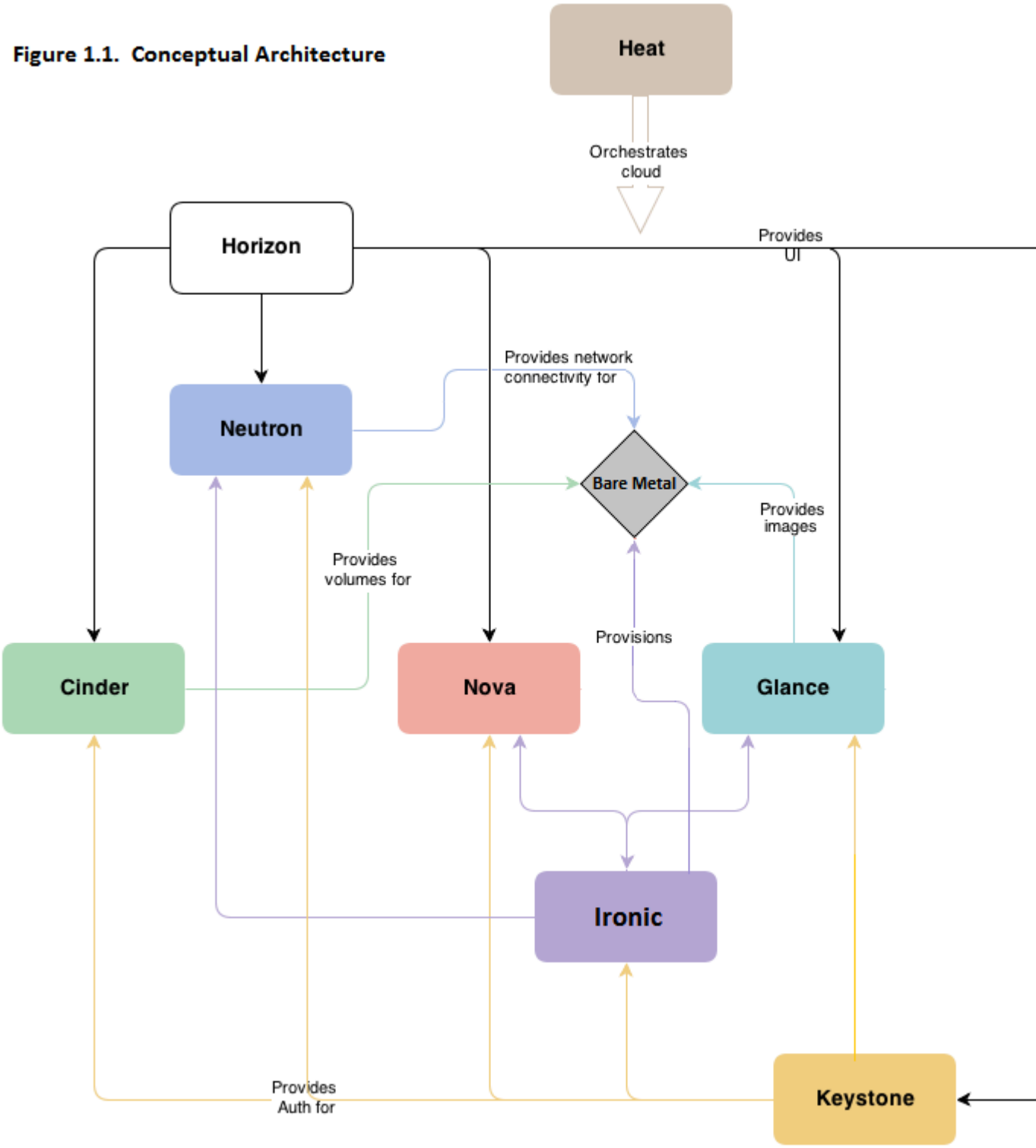
2.2 Conceptual Architecture

The following diagram shows the relationships and how all services come into play during the provisioning of a physical server.

Note

Ceilometer and Swift can be used with Ironic, but are missing from this diagram.

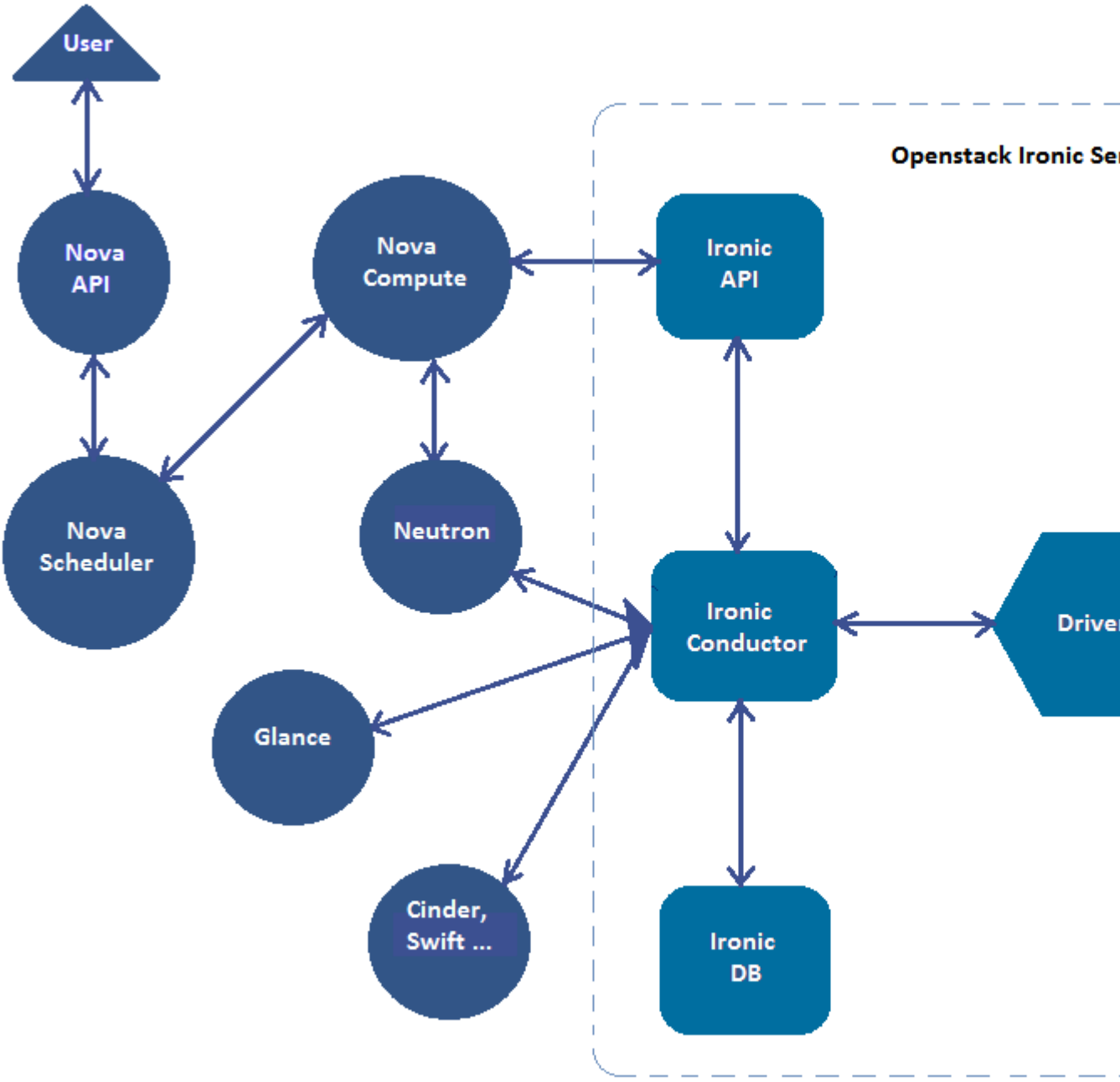
Figure 1.1. Conceptual Architecture



2.3 Logical Architecture

The diagram below shows the logical architecture. It shows the basic components that form the Ironic service, the relation of Ironic service with other OpenStack services and the logical flow of a boot instance request resulting in the provisioning of a physical server.

Figure 1.2. Logical Architecture



The Ironic service is composed of the following components:

1. A RESTful API service, by which operators and other services may interact with the managed bare metal servers.
2. A conductor service, which does the bulk of the work. Functionality is exposed via the API service. The Conductor and API services communicate via RPC.
3. Various drivers that support heterogeneous hardware
4. A message queue
5. A database for storing information about the resources. Among other things, this includes the state of the conductors, nodes (physical servers), and drivers.

As in Figure 1.2. Logical Architecture, a user request to boot an instance is passed to the Nova Compute service via Nova API and Nova Scheduler. The Compute service hands over this request to the Ironic service, where the request passes from the Ironic API, to the Conductor, to a driver to successfully provision a physical server for the user.

Just as the Nova Compute service talks to various OpenStack services like Glance, Neutron, or Swift to provision a virtual machine instance, here the Ironic service talks to the same OpenStack services for image, network and other resource needs to provision a bare metal instance.

2.4 Key Technologies for Bare Metal Hosting

2.4.1 Preboot Execution Environment (PXE)

PXE is part of the Wired for Management (WfM) specification developed by Intel and Microsoft. The PXE enables system's BIOS and network interface card (NIC) to bootstrap a computer from the network in place of a disk. Bootstrapping is the process by which a system loads the OS into local memory so that it can be executed by the processor. This capability of allowing a system to boot over a network simplifies server deployment and server management for administrators.

2.4.2 Dynamic Host Configuration Protocol (DHCP)

DHCP is a standardized networking protocol used on Internet Protocol (IP) networks for dynamically distributing network configuration parameters, such as IP addresses for interfaces and services. Using PXE, the BIOS uses DHCP to obtain an IP address for the network interface and to locate the server that stores the network bootstrap program (NBP).

2.4.3 Network Bootstrap Program (NBP)

NBP is equivalent to GRUB (GRand Unified Bootloader) or LILO (LInux LOader) - loaders which are traditionally used in local booting. Like the boot program in a hard drive environment, the NBP is responsible for loading the OS kernel into memory so that the OS can be bootstrapped over a network.

2.4.4 Trivial File Transfer Protocol (TFTP)

TFTP is a simple file transfer protocol that is generally used for automated transfer of configuration or boot files between machines in a local environment. In a PXE environment, TFTP is used to download NBP over the network using information from the DHCP server.

2.4.5 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. It is a method to manage systems that may be unresponsive or powered off by using only a network connection to the hardware rather than to an operating system.

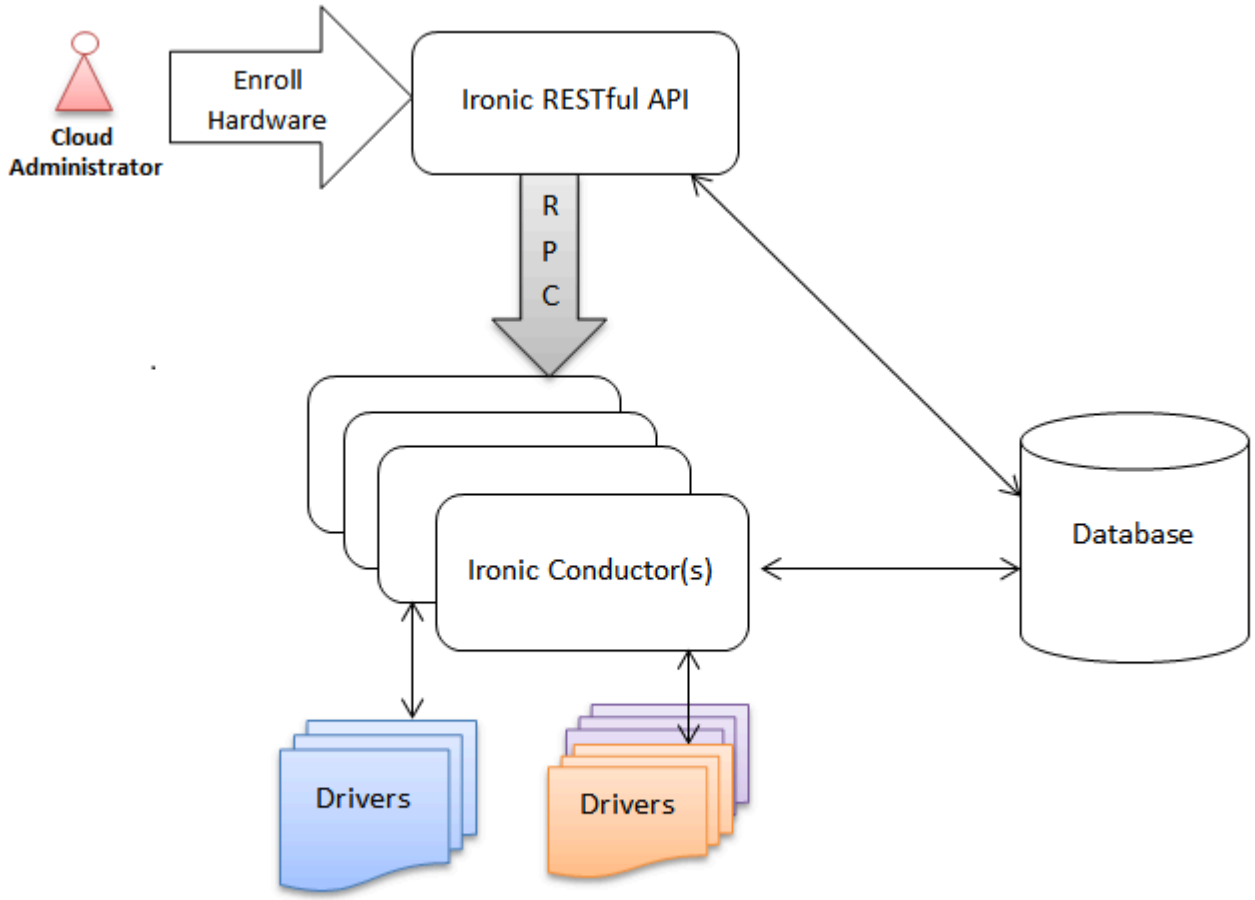
2.5 Ironic Deployment Architecture

The Ironic RESTful API service is used to enroll hardware that Ironic will manage. A cloud administrator usually registers the hardware, specifying their attributes such as MAC addresses and IPMI credentials. There can be multiple instances of the API service.

The Ironic conductor service does the bulk of the work. For security reasons, it is advisable to place the conductor service on an isolated host, since it is the only service that requires access to both the data plane and IPMI control plane.

There can be multiple instances of the conductor service to support various class of drivers and also to manage fail over. Instances of the conductor service should be on separate nodes. Each conductor can itself run many drivers to operate heterogeneous hardware. This is depicted in the following figure.

The API exposes a list of supported drivers and the names of conductor hosts servicing them.



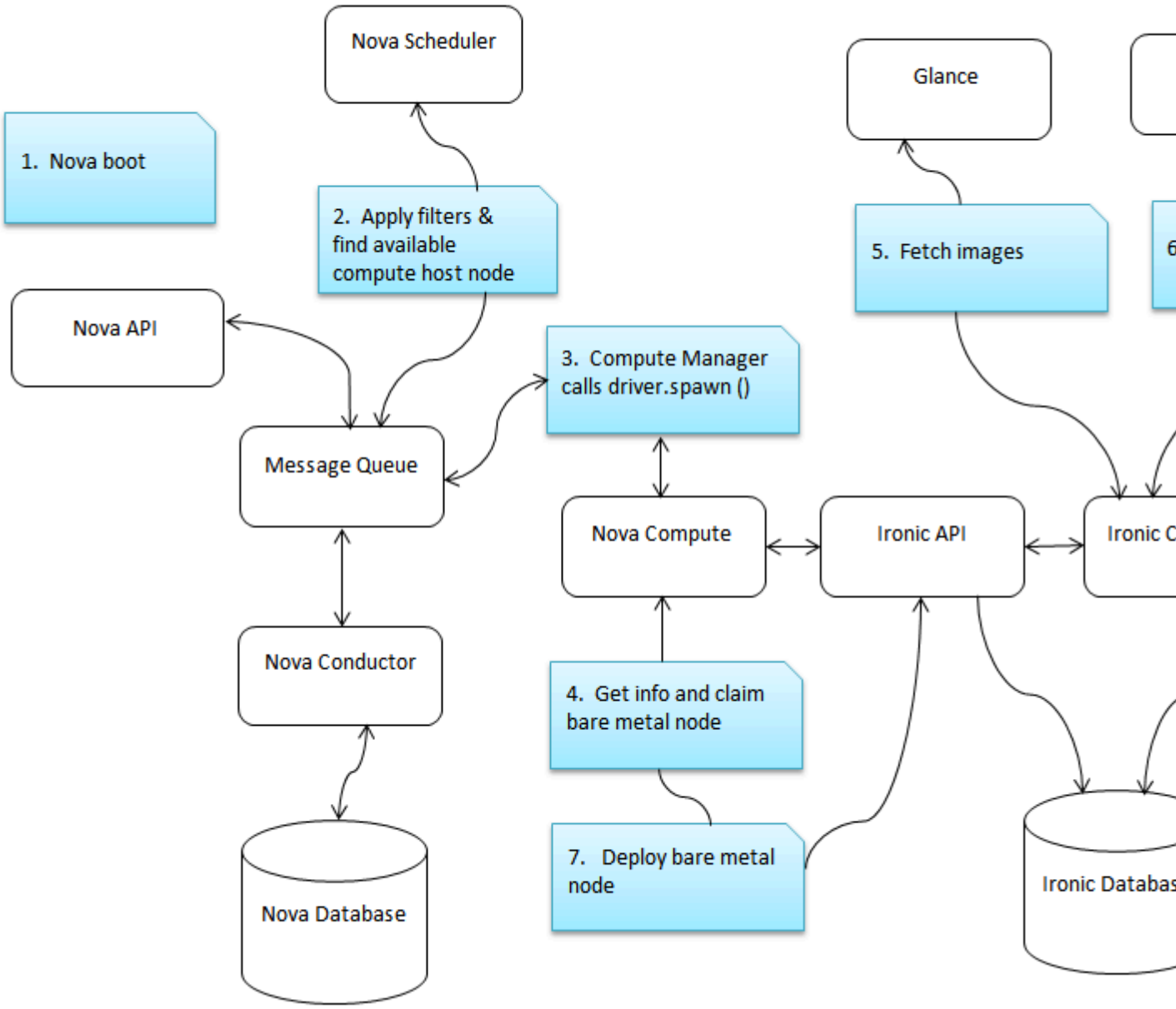
2.6 Understanding Bare Metal Deployment

What happens when a boot instance request comes in? The below diagram walks through the steps involved during the provisioning of a bare metal instance.

These pre-requisites must be met before the deployment process:

- Dependent packages to be configured on the Bare Metal service node(s) where ironic-conductor is running like tftp-server, ipmi, or syslinux for bare metal provisioning.
- Nova must be configured to make use of the bare metal service endpoint and compute driver should be configured to use ironic driver on the Nova compute node(s).
- Flavors to be created for the available hardware. Nova must know the flavor to boot from.
- Images to be made available in Glance. Listed below are some image types required for successful bare metal deployment:
  - `bm-deploy-kernel`
  - `bm-deploy-ramdisk`
  - `user-image`
  - `user-image-vmlinuz`
  - `user-image-initrd`
- Hardware to be enrolled via Ironic RESTful API service.

Figure 1.3.3. Bare Metal Deployment Steps



2.6.1 Deploy Process

1. A boot instance request comes in via the Nova API, through the message queue to the Nova scheduler.

2. Nova scheduler applies filter and finds the eligible compute node. Nova scheduler uses flavor `extra_specs` detail such as `cpu_arch`, `baremetal:deploy_kernel_id` or `baremetal:deploy_ramdisk_id` to match the target physical node.
3. A spawn task is placed by the driver which contains all information such as which image to boot from. It invokes the `driver.spawn` from the virtual layer of Nova compute.
4. Information about the bare metal node is retrieved from the bare metal database and the node is reserved.
5. Images from Glance are pulled down to the local disk of the Ironic conductor servicing the bare metal node.
    a. For `pxe_*` drivers these include all images: both the deploy ramdisk and user instance images.
    b. For `agent_*` drivers only the deploy ramdisk is stored locally. Temporary URLs in OpenStack's Object Storage service are created for user instance images.
6. Virtual interfaces are plugged in and Neutron API updates DHCP port to support PXE/TFTP options.
7. Nova's ironic driver issues a deploy request via the Ironic API to the Ironic conductor servicing the bare metal node.
8. PXE driver prepares tftp bootloader.
9. The IPMI driver issues command to enable network boot of a node and power it on.
10. The DHCP boots the deploy ramdisk. Next, depending on the exact driver used, either the conductor copies the image over iSCSI to the physical node (`pxe_*` group of drivers) or the deploy ramdisk downloads the image from a temporary URL (`agent_*` group of drivers), which can be generated by a variety of object stores. For example, *swift* or *radosgw*, and uploaded to OpenStack's Object Storage service. In the former case, the conductor connects to the iSCSI end point, partitions volume, `dd` the image and closes the iSCSI connection. The deployment is done. The Ironic conductor will switch pxe config to service mode and notify ramdisk agent on the successful deployment.
11. The IPMI driver reboots the bare metal node. There are 2 power cycles during bare metal deployment; the first time when powered-on, the images get deployed as mentioned in step 9. The second time as in this case, after the images are deployed, the node is powered up.

12. The bare metal node status is updated and the node instance is made available.

2.6.2 Example 1: PXE Boot and iSCSI Deploy Process

This process is used with `pxe_*` family of drivers.

(From a talk (https://www.openstack.org/summit/vancouver-2015/summit-videos/presentation/isn-and-039t-it-ironic-the-bare-metal-cloud) and slides (http://www.slideshare.net/devananda1/isnt-it-ironic-managing-a-bare-metal-cloud-osl-tes-2015)).

2.6.3 Example 2: PXE Boot and Direct Deploy Process

This process is used with `agent_*` family of drivers.

(From a talk (https://www.openstack.org/summit/vancouver-2015/summit-videos/presentation/isn-and-039t-it-ironic-the-bare-metal-cloud) and slides (http://www.slideshare.net/devananda1/isnt-it-ironic-managing-a-bare-metal-cloud-osl-tes-2015)).

3 Horizon User Guide

How to use Horizon in your own projects.

3.1 OpenStack Dashboard User Documentation

As a cloud end user, you can use the OpenStack dashboard to provision your own resources within the limits set by administrators. You can modify the examples provided in this section to create other types and sizes of server instances.

3.1.1 Log in to the dashboard

The dashboard is generally installed on the controller node.

1. Ask the cloud operator for the host name or public IP address from which you can access the dashboard, and for your user name and password. If the cloud supports multi-domain model, you also need to ask for your domain name.
2. Open a web browser that has JavaScript and cookies enabled.

   Note

   To use the Virtual Network Computing (VNC) client for the dashboard, your browser must support HTML5 Canvas and HTML5 WebSockets. The VNC client is based on noVNC. For details, see noVNC: HTML5 VNC Client (https://github.com/kanaka/noVNC/blob/master/README.md). For a list of supported browsers, see Browser support (https://github.com/kanaka/noVNC/wiki/Browser-support).

3. In the address bar, enter the host name or IP address for the dashboard, for example, `https://ipAddressOrHostName/`.

   Note

   If a certificate warning appears when you try to access the URL for the first time, a self-signed certificate is in use, which is not considered trustworthy by default. Verify the certificate or add an exception in the browser to bypass the warning.

4. On the *Log In* page, enter your user name and password, and click *Sign In*. If the cloud supports multi-domain model, you also need to enter your domain name.
   The top of the window displays your user name. You can also access the *Settings* tab (*Section 3.1.1.4, "OpenStack dashboard — Settings tab"*) or sign out of the dashboard.
   The visible tabs and functions in the dashboard depend on the access permissions, or roles, of the user you are logged in as.

   - If you are logged in as an end user, the *Project* tab (*Section 3.1.1.1, "OpenStack dashboard — Project tab"*) and *Identity* tab (*Section 3.1.1.3, "OpenStack dashboard — Identity tab"*) are displayed.
   - If you are logged in as an administrator, the *Project* tab (*Section 3.1.1.1, "OpenStack dashboard — Project tab"*) and *Admin* tab (*Section 3.1.1.2, "OpenStack dashboard — Admin tab"*) and *Identity* tab (*Section 3.1.1.3, "OpenStack dashboard — Identity tab"*) are displayed.

Note

Some tabs, such as *Orchestration* and *Firewalls*, only appear on the dashboard if they are properly configured.

3.1.1.1 OpenStack dashboard — Project tab

Projects are organizational units in the cloud and are also known as tenants or accounts. Each user is a member of one or more projects. Within a project, a user creates and manages instances.

From the *Project* tab, you can view and manage the resources in a selected project, including instances and images. You can select the project from the drop-down menu at the top left. If the cloud supports multi-domain model, you can also select the domain from this menu.

FIGURE 3.1: FIGURE: PROJECT TAB

From the *Project* tab, you can access the following categories:

3.1.1.1.1 Compute tab

- *Overview*: View reports for the project.
- *Instances*: View, launch, create a snapshot from, stop, pause, or reboot instances, or connect to them through VNC.
- *Volumes*: Use the following tabs to complete these tasks:
  - *Volumes*: View, create, edit, and delete volumes.
  - *Volume Snapshots*: View, create, edit, and delete volume snapshots.
- *Images*: View images and instance snapshots created by project users, plus any images that are publicly available. Create, edit, and delete images, and launch instances from images and snapshots.
- *Access & Security*: Use the following tabs to complete these tasks:
  - *Security Groups*: View, create, edit, and delete security groups and security group rules.
  - *Key Pairs*: View, create, edit, import, and delete key pairs.
  - *Floating IPs*: Allocate an IP address to or release it from a project.
  - *API Access*: View API endpoints.
- *Shares*: Use the following tabs to complete these tasks:
  - *Shares*: View, create, manage, and delete shares.
  - *Snapshots*: View, manage, and delete volume snapshots.
  - *Share Networks*: View, manage, and delete share networks.
  - *Security Services*: View, manage, and delete security services.

3.1.1.1.2 Network tab

- *Network Topology*: View the network topology.
- *Networks*: Create and manage public and private networks.
- *Routers*: Create and manage routers.
- *Load Balancers*: Create and manage load balancers.
    - *Pools*: Add and manage pools.
    - *Members*: Add and manage members.
    - *Monitors*: Add and manage monitors.
- *Firewalls*: Create and manage firewalls.
    - *Firewalls*: Create and manage firewalls.
    - *Firewall Policies*: Add and manage firewall policies.
    - *Firewall Rules*: Add and manage firewall rules.

3.1.1.1.3 Orchestration tab

- *Stacks*: Use the REST API to orchestrate multiple composite cloud applications.
- *Resource Types*: Show a list of all the supported resource types for HOT templates.

3.1.1.1.4 Object Store tab

- *Containers*: Create and manage containers and objects.

3.1.1.2 OpenStack dashboard — Admin tab

Administrative users can use the *Admin* tab to view usage and to manage instances, volumes, flavors, images, networks, and so on.

FIGURE 3.2: FIGURE: ADMIN TAB

From the *Admin* tab, you can access the following category to complete these tasks:

3.1.1.2.1 System tab

- *Overview*: View basic reports.
- *Resource Usage*: Use the following tabs to view the following usages:
  - *Usage Report*: View the usage report.
  - *Stats*: View the statistics of all resources.
- *Hypervisors*: View the hypervisor summary.
- *Host Aggregates*: View, create, and edit host aggregates. View the list of availability zones.
- *Instances*: View, pause, resume, suspend, migrate, soft or hard reboot, and delete running instances that belong to users of some, but not all, projects. Also, view the log for an instance or access an instance through VNC.
- *Volumes*: Use the following tabs to complete these tasks:
  - *Volumes*: View, create, manage, and delete volumes.
  - *Volume Types*: View, create, manage, and delete volume types.
  - *Volume Snapshots*: View, manage, and delete volume snapshots.
- *Flavors*: View, create, edit, view extra specifications for, and delete flavors. A flavor is the size of an instance.
- *Images*: View, create, edit properties for, and delete custom images.
- *Networks*: View, create, edit properties for, and delete networks.
- *Routers*: View, create, edit properties for, and delete routers.
- *Defaults*: View default quota values. Quotas are hard-coded in OpenStack Compute and define the maximum allowable size and number of resources.
- *Metadata Definitions*: Import namespace and view the metadata information.

- *System Information*: Use the following tabs to view the service information:
  - *Services*: View a list of the services.
  - *Compute Services*: View a list of all Compute services.
  - *Block Storage Services*: View a list of all Block Storage services.
  - *Network Agents*: View the network agents.
  - *Orchestration Services*: View a list of all Orchestration services.
- *Shares*: Use the following tabs to complete these tasks:
  - *Shares*: View, create, manage, and delete shares.
  - *Snapshots*: View, manage, and delete volume snapshots.
  - *Share Networks*: View, manage, and delete share networks.
  - *Security Services*: View, manage, and delete security services.
  - *Share Types*: View, create, manage, and delete share types.
  - *Share Servers*: View, manage, and delete share servers.

3.1.1.3 OpenStack dashboard — Identity tab

FIGURE 3.3: FIGURE:IDENTITY TAB

- *Projects*: View, create, assign users to, remove users from, and delete projects.
- *Users*: View, create, enable, disable, and delete users.

3.1.1.4 OpenStack dashboard — Settings tab

FIGURE 3.4: FIGURE:SETTINGS TAB

Click the *Settings* button from the user drop down menu at the top right of any page, you will see the *Settings* tab.

- *User Settings*: View and manage dashboard settings.
- *Change Password*: Change the password of the user.

3.1.2 Upload and manage images

A virtual machine image, referred to in this document simply as an image, is a single file that contains a virtual disk that has a bootable operating system installed on it. Images are used to create virtual machine instances within the cloud. For information about creating image files, see the OpenStack Virtual Machine Image Guide (https://docs.openstack.org/image-guide/).

Depending on your role, you may have permission to upload and manage virtual machine images. Operators might restrict the upload and management of images to cloud administrators or operators only. If you have the appropriate privileges, you can use the dashboard to upload and manage images in the admin project.

Note

You can also use the `openstack` and `glance` command-line clients or the Image service to manage images.

3.1.2.1 Upload an image

Follow this procedure to upload an image to a project:

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Images* category.
4. Click *Create Image*.
   The *Create An Image* dialog box appears.

FIGURE 3.5: DASHBOARD — CREATE IMAGE

5. Enter the following values:

Image Name	Enter a name for the image.
Image Description	Enter a brief description of the image.
Image Source	Choose the image source from the drop-down list. Your choices are *Image*

	Location and *Image File*.
Image File or *Image Location*	Based on your selection for *Image Source*, you either enter the location URL of the image in the *Image Location* field,

	or browse for the image file on your file system and add it.
Format	Select the image format (for example, QCOW2) for the image.
Architecture	Specify the ar-

	chitecture. For example, `i386` for a 32-bit architecture or `x86_64` for a 64-bit architecture.
Minimum Disk (GB)	Leave this field empty.
Minimum RAM (MB)	Leave this field empty.

Copy Data	Specify this option to copy image data to the Image service.
Visibility	The access permission for the image. `Public` or `Private`.
Protected	Select this check box

		to ensure that only users with permissions can delete the image. `Yes` or `No`.
	Image Metadata	Specify this option to add resource metadata. The glance Metadata Cat-

	alog provides a list of metadata image definitions. (Note: Not all cloud providers enable this feature.)

6. Click *Create Image*.
   The image is queued to be uploaded. It might take some time before the status changes from Queued to Active.

3.1.2.2 Update an image

Follow this procedure to update an existing image.

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. Select the image that you want to edit.

4. In the *Actions* column, click the menu button and then select *Edit Image* from the list.
5. In the *Edit Image* dialog box, you can perform various actions. For example:
    - Change the name of the image.
    - Select the *Public* check box to make the image public.
    - Clear the *Public* check box to make the image private.
6. Click *Edit Image*.

3.1.2.3 Delete an image

Deletion of images is permanent and **cannot** be reversed. Only users with the appropriate permissions can delete images.

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Images* category.
4. Select the images that you want to delete.
5. Click *Delete Images*.
6. In the *Confirm Delete Images* dialog box, click *Delete Images* to confirm the deletion.

3.1.3 Configure access and security for instances

Before you launch an instance, you should add security group rules to enable users to ping and use SSH to connect to the instance. Security groups are sets of IP filter rules that define networking access and are applied to all instances within a project. To do so, you either add rules to the default security group *Section 3.1.3.1, "Add a rule to the default security group"* or add a new security group with rules.

Key pairs are SSH credentials that are injected into an instance when it is launched. To use key pair injection, the image that the instance is based on must contain the `cloud-init` package. Each project should have at least one key pair. For more information, see the section *Section 3.1.3.2, "Add a key pair"*.

If you have generated a key pair with an external tool, you can import it into OpenStack. The key pair can be used for multiple instances that belong to a project. For more information, see the section *Section 3.1.3.3, "Import a key pair"*.

Note

A key pair belongs to an individual user, not to a project. To share a key pair across multiple users, each user needs to import that key pair.

When an instance is created in OpenStack, it is automatically assigned a fixed IP address in the network to which the instance is assigned. This IP address is permanently associated with the instance until the instance is terminated. However, in addition to the fixed IP address, a floating IP address can also be attached to an instance. Unlike fixed IP addresses, floating IP addresses are able to have their associations modified at any time, regardless of the state of the instances involved.

3.1.3.1 Add a rule to the default security group

This procedure enables SSH and ICMP (ping) access to instances. The rules apply to all instances within a given project, and should be set for every project unless there is a reason to prohibit SSH or ICMP access to the instances.

This procedure can be adjusted as necessary to add additional security group rules to a project, if your cloud requires them.

Note

When adding a rule, you must specify the protocol used with the destination port or source port.

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Access & Security* category. The *Security Groups* tab shows the security groups that are available for this project.
4. Select the default security group and click *Manage Rules*.

5. To allow SSH access, click *Add Rule*.
6. In the *Add Rule* dialog box, enter the following values:
   - **Rule**: `SSH`
   - **Remote**: `CIDR`
   - **CIDR**: `0.0.0.0/0`

   Note

   To accept requests from a particular range of IP addresses, specify the IP address block in the *CIDR* box.
7. Click *Add*.
   Instances will now have SSH port 22 open for requests from any IP address.
8. To add an ICMP rule, click *Add Rule*.
9. In the *Add Rule* dialog box, enter the following values:
   - **Rule**: `All ICMP`
   - **Direction**: `Ingress`
   - **Remote**: `CIDR`
   - **CIDR**: `0.0.0.0/0`
10. Click *Add*.
    Instances will now accept all incoming ICMP packets.

3.1.3.2 Add a key pair

Create at least one key pair for each project.

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Access & Security* category.

4. Click the *Key Pairs* tab, which shows the key pairs that are available for this project.
5. Click *Create Key Pair*.
6. In the *Create Key Pair* dialog box, enter a name for your key pair, and click *Create Key Pair*.
7. Respond to the prompt to download the key pair.

3.1.3.3 Import a key pair

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Access & Security* category.
4. Click the *Key Pairs* tab, which shows the key pairs that are available for this project.
5. Click *Import Key Pair*.
6. In the *Import Key Pair* dialog box, enter the name of your key pair, copy the public key into the *Public Key* box, and then click *Import Key Pair*.
7. Save the `*.pem` file locally.
8. To change its permissions so that only you can read and write to the file, run the following command:

```
$ chmod 0600 yourPrivateKey.pem
```

Note

If you are using the Dashboard from a Windows computer, use PuTTYgen to load the `*.pem` file and convert and save it as `*.ppk`. For more information see the WinSCP web page for PuTTYgen (http://winscp.net/eng/docs/ui_puttygen).

9. To make the key pair known to SSH, run the **`ssh-add`** command.

```
$ ssh-add yourPrivateKey.pem
```

The Compute database registers the public key of the key pair.

The Dashboard lists the key pair on the *Access & Security* tab.

3.1.3.4 Allocate a floating IP address to an instance

When an instance is created in OpenStack, it is automatically assigned a fixed IP address in the network to which the instance is assigned. This IP address is permanently associated with the instance until the instance is terminated.

However, in addition to the fixed IP address, a floating IP address can also be attached to an instance. Unlike fixed IP addresses, floating IP addresses can have their associations modified at any time, regardless of the state of the instances involved. This procedure details the reservation of a floating IP address from an existing pool of addresses and the association of that address with a specific instance.

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Access & Security* category.
4. Click the *Floating IPs* tab, which shows the floating IP addresses allocated to instances.
5. Click *Allocate IP To Project*.
6. Choose the pool from which to pick the IP address.
7. Click *Allocate IP*.
8. In the *Floating IPs* list, click *Associate*.
9. In the *Manage Floating IP Associations* dialog box, choose the following options:
   - The *IP Address* field is filled automatically, but you can add a new IP address by clicking the + button.
   - In the *Port to be associated* field, select a port from the list.
     The list shows all the instances with their fixed IP addresses.
10. Click *Associate*.

Note

To disassociate an IP address from an instance, click the *Disassociate* button.

To release the floating IP address back into the floating IP pool, click the *Release Floating IP* option in the *Actions* column.

3.1.4 Launch and manage instances

Instances are virtual machines that run inside the cloud. You can launch an instance from the following sources:

- Images uploaded to the Image service.
- Image that you have copied to a persistent volume. The instance launches from the volume, which is provided by the `cinder-volume` API through iSCSI.
- Instance snapshot that you took.

3.1.4.1 Launch an instance

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Instances* category.
   The dashboard shows the instances with its name, its private and floating IP addresses, size, status, task, power state, and so on.
4. Click *Launch Instance*.
5. In the *Launch Instance* dialog box, specify the following values:
   Details tab

   Instance Name

   Assign a name to the virtual machine.

   Availability Zone

   By default, this value is set to the availability zone given by the cloud provider (for example, `us-west` or `apac-south`). For some cases, it could be `nova`.

   Note

   The name you assign here becomes the initial host name of the server. If the name is longer than 63 characters, the Compute service truncates it automatically to ensure dnsmasq works correctly.

   After the server is built, if you change the server name in the API or change the host name directly, the names are not updated in the dashboard.

Server names are not guaranteed to be unique when created so you could have two instances with the same host name.

Count

To launch multiple instances, enter a value greater than `1`. The default is `1`.

Source tab

Instance Boot Source

Your options are:

Boot from image

If you choose this option, a new field for *Image Name* displays. You can select the image from the list.

Boot from snapshot

If you choose this option, a new field for *Instance Snapshot* displays. You can select the snapshot from the list.

Boot from volume

If you choose this option, a new field for *Volume* displays. You can select the volume from the list.

Boot from image (creates a new volume)

With this option, you can boot from an image and create a volume by entering the *Device Size* and *Device Name* for your volume. Click the *Delete Volume on Instance Delete* option to delete the volume on deleting the instance.

Boot from volume snapshot (creates a new volume)

Using this option, you can boot from a volume snapshot and create a new volume by choosing *Volume Snapshot* from a list and adding a *Device Name* for your volume. Click the *Delete Volume on Instance Delete* option to delete the volume on deleting the instance.

Image Name

This field changes based on your previous selection. If you have chosen to launch an instance using an image, the *Image Name* field displays. Select the image name from the dropdown list.

Instance Snapshot

This field changes based on your previous selection. If you have chosen to launch an instance using a snapshot, the *Instance Snapshot* field displays. Select the snapshot name from the dropdown list.

Volume

This field changes based on your previous selection. If you have chosen to launch an instance using a volume, the *Volume* field displays. Select the volume name from the dropdown list. If you want to delete the volume on instance delete, check the *Delete Volume on Instance Delete* option.

Flavor tab

Flavor

Specify the size of the instance to launch.

Note

The flavor is selected based on the size of the image selected for launching an instance. For example, while creating an image, if you have entered the value in the *Minimum RAM (MB)* field as 2048, then on selecting the image, the default flavor is `m1.small`.

Networks tab

Selected Networks

To add a network to the instance, click the + in the *Available* field.

Network Ports tab

Ports

Activate the ports that you want to assign to the instance.

Security Groups tab

Security Groups

Activate the security groups that you want to assign to the instance.

Security groups are a kind of cloud firewall that define which incoming network traffic is forwarded to instances.

If you have not created any security groups, you can assign only the default security group to the instance.

Key Pair tab

Key Pair

Specify a key pair.

If the image uses a static root password or a static key set (neither is recommended), you do not need to provide a key pair to launch the instance.

Configuration tab

Customization Script Source

Specify a customization script that runs after your instance launches.

Metadata tab

Available Metadata

Add Metadata items to your instance.

6. Click *Launch Instance*.

   The instance starts on a compute node in the cloud.

Note

If you did not provide a key pair, security groups, or rules, users can access the instance only from inside the cloud through VNC. Even pinging the instance is not possible without an ICMP rule configured.

You can also launch an instance from the *Images* or *Volumes* category when you launch an instance from an image or a volume respectively.

When you launch an instance from an image, OpenStack creates a local copy of the image on the compute node where the instance starts.

For details on creating images, see Creating images manually (https://docs.openstack.org/image-guide/create-images-manually.html) in the *OpenStack Virtual Machine Image Guide*.

When you launch an instance from a volume, note the following steps:

- To select the volume from which to launch, launch an instance from an arbitrary image on the volume. The arbitrary image that you select does not boot. Instead, it is replaced by the image on the volume that you choose in the next steps.
  To boot a Xen image from a volume, the image you launch in must be the same type, fully virtualized or paravirtualized, as the one on the volume.
- Select the volume or volume snapshot from which to boot. Enter `vda` for KVM images or `xvda` for Xen images.

Note

When running QEMU without support for the hardware virtualization, set `cpu_mode="none"` alongside `virt_type=qemu` in `/etc/nova/nova-compute.conf` to solve the following error:

```
libvirtError: unsupported configuration: CPU mode 'host-model'
for ``x86_64`` qemu domain on ``x86_64`` host is not supported by hypervisor
```

3.1.4.2 Connect to your instance by using SSH

To use SSH to connect to your instance, use the downloaded keypair file.

1. Copy the IP address for your instance.
2. Use the **`ssh`** command to make a secure connection to the instance.
3. At the prompt, type `yes`.

It is also possible to SSH into an instance without an SSH keypair, if the administrator has enabled root password injection. For more information about root password injection, see Injecting the administrator password (https://docs.openstack.org/nova/latest/admin/admin-password-injection.html) in the *OpenStack Administrator Guide*.

3.1.4.3 Track usage for instances

You can track usage for instances for each project. You can track costs per month by showing meters like number of vCPUs, disks, RAM, and uptime for all your instances.

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Overview* category.
4. To query the instance usage for a month, select a month and click *Submit*.
5. To download a summary, click *Download CSV Summary*.

3.1.4.4 Create an instance snapshot

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click the *Instances* category.
4. Select the instance from which to create a snapshot.
5. In the actions column, click *Create Snapshot*.
6. In the *Create Snapshot* dialog box, enter a name for the snapshot, and click *Create Snapshot*. The *Images* category shows the instance snapshot.

To launch an instance from the snapshot, select the snapshot and click *Launch*. Proceed with launching an instance.

3.1.4.5 Manage an instance

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Instances* category.
4. Select an instance.
5. In the menu list in the actions column, select the state.
   You can resize or rebuild an instance. You can also choose to view the instance console log, edit instance or the security groups. Depending on the current state of the instance, you can pause, resume, suspend, soft or hard reboot, or terminate it.

3.1.5 Create and manage networks

The OpenStack Networking service provides a scalable system for managing the network connectivity within an OpenStack cloud deployment. It can easily and quickly react to changing network needs (for example, creating and assigning new IP addresses).

Networking in OpenStack is complex. This section provides the basic instructions for creating a network and a router. For detailed information about managing networks, refer to the OpenStack Networking Guide (https://docs.openstack.org/neutron/latest/admin/).

3.1.5.1 Create a network

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Network* tab and click *Networks* category.
4. Click *Create Network*.
5. In the *Create Network* dialog box, specify the following values.
   Network tab
   Network Name: Specify a name to identify the network.
   Shared: Share the network with other projects. Non admin users are not allowed to set shared option.
   Admin State: The state to start the network in.
   Create Subnet: Select this check box to create a subnet
   You do not have to specify a subnet when you create a network, but if you do not specify a subnet, the network can not be attached to an instance.
   Subnet tab
   Subnet Name: Specify a name for the subnet.
   Network Address: Specify the IP address for the subnet.
   IP Version: Select IPv4 or IPv6.
   Gateway IP: Specify an IP address for a specific gateway. This parameter is optional.
   Disable Gateway: Select this check box to disable a gateway IP address.
   Subnet Details tab
   Enable DHCP: Select this check box to enable DHCP.
   Allocation Pools: Specify IP address pools.

DNS Name Servers: Specify a name for the DNS server.

Host Routes: Specify the IP address of host routes.

6. Click *Create*.
   The dashboard shows the network on the *Networks* tab.

3.1.5.2 Create a router

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Network* tab and click *Routers* category.
4. Click *Create Router*.
5. In the *Create Router* dialog box, specify a name for the router and *External Network*, and click *Create Router*.
   The new router is now displayed in the *Routers* tab.
6. To connect a private network to the newly created router, perform the following steps:
   a. On the *Routers* tab, click the name of the router.
   b. On the *Router Details* page, click the *Interfaces* tab, then click *Add Interface*.
   c. In the *Add Interface* dialog box, select a *Subnet*.
      Optionally, in the *Add Interface* dialog box, set an *IP Address* for the router interface for the selected subnet.
      If you choose not to set the *IP Address* value, then by default OpenStack Networking uses the first host IP address in the subnet.
      The *Router Name* and *Router ID* fields are automatically updated.
7. Click *Add Interface*.

You have successfully created the router. You can view the new topology from the *Network Topology* tab.

3.1.5.3 Create a port

Warning

Creating and managing ports requires administrator privileges. Contact an administrator before adding or changing ports.

1. Log in to the dashboard.
2. Select the appropriate project from the drop-down menu at the top left.
3. On the *Admin* tab, click *Networks* category.
4. Click on the *Network Name* of the network in which the port has to be created.
5. In the *Create Port* dialog box, specify the following values.
   Name: Specify name to identify the port.
   Device ID: Device ID attached to the port.
   Device Owner: Device owner attached to the port.
   Binding Host: The ID of the host where the port is allocated.
   Binding VNIC Type: Select the VNIC type that is bound to the neutron port.
6. Click *Create Port*.
   The new port is now displayed in the *Ports* list.

3.1.6 Create and manage object containers

OpenStack Object Storage (swift) is used for redundant, scalable data storage using clusters of standardized servers to store petabytes of accessible data. It is a long-term storage system for large amounts of static data which can be retrieved and updated.

OpenStack Object Storage provides a distributed, API-accessible storage platform that can be integrated directly into an application or used to store any type of file, including VM images, backups, archives, or media files. In the OpenStack dashboard, you can only manage containers and objects.

In OpenStack Object Storage, containers provide storage for objects in a manner similar to a Windows folder or Linux file directory, though they cannot be nested. An object in OpenStack consists of the file to be stored in the container and any accompanying metadata.

3.1.6.1 Create a container

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Object Store* tab and click *Containers* category.
4. Click *Container*.
5. In the *Create Container* dialog box, enter a name for the container, and then click *Create*.

You have successfully created a container.

Note

To delete a container, click the *More* button and select *Delete Container*.

3.1.6.2 Upload an object

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Object Store* tab and click *Containers* category.
4. Select the container in which you want to store your object.
5. Click the *Upload File* icon.
   The *Upload File To Container: <name>* dialog box appears. `<name>` is the name of the container to which you are uploading the object.
6. Enter a name for the object.
7. Browse to and select the file that you want to upload.
8. Click *Upload File*.

You have successfully uploaded an object to the container.

Note

To delete an object, click the *More button* and select *Delete Object*.

3.1.6.3 Manage an object

To edit an object

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Object Store* tab and click *Containers* category.
4. Select the container in which you want to store your object.
5. Click the menu button and choose *Edit* from the dropdown list.
   The *Edit Object* dialog box is displayed.
6. Browse to and select the file that you want to upload.
7. Click *Update Object*.

Note

To delete an object, click the menu button and select *Delete Object*.

To copy an object from one container to another

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Object Store* tab and click *Containers* category.
4. Select the container in which you want to store your object.
5. Click the menu button and choose *Copy* from the dropdown list.
6. In the *Copy Object* launch dialog box, enter the following values:
   - *Destination Container*: Choose the destination container from the list.
   - *Path*: Specify a path in which the new copy should be stored inside of the selected container.
   - *Destination object name*: Enter a name for the object in the new container.

7. Click *Copy Object*.

To create a metadata-only object without a file

You can create a new object in container without a file available and can upload the file later when it is ready. This temporary object acts a place-holder for a new object, and enables the user to share object metadata and URL info in advance.

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Object Store* tab and click *Containers* category.
4. Select the container in which you want to store your object.
5. Click *Upload Object*.
   The *Upload Object To Container*: `<name>` dialog box is displayed.
   `<name>` is the name of the container to which you are uploading the object.
6. Enter a name for the object.
7. Click *Update Object*.

To create a pseudo-folder

Pseudo-folders are similar to folders in your desktop operating system. They are virtual collections defined by a common prefix on the object's name.

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Object Store* tab and click *Containers* category.
4. Select the container in which you want to store your object.
5. Click *Create Pseudo-folder*.
   The *Create Pseudo-Folder in Container* `<name>` dialog box is displayed. `<name>` is the name of the container to which you are uploading the object.
6. Enter a name for the pseudo-folder.
   A slash (/) character is used as the delimiter for pseudo-folders in Object Storage.
7. Click *Create*.

3.1.7 Create and manage volumes

Volumes are block storage devices that you attach to instances to enable persistent storage. You can attach a volume to a running instance or detach a volume and attach it to another instance at any time. You can also create a snapshot from or delete a volume. Only administrative users can create volume types.

3.1.7.1 Create a volume

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Volumes* category.
4. Click *Create Volume*.
   In the dialog box that opens, enter or select the following values.
   Volume Name: Specify a name for the volume.
   Description: Optionally, provide a brief description for the volume.
   Volume Source: Select one of the following options:

   - No source, empty volume: Creates an empty volume. An empty volume does not contain a file system or a partition table.
   - Snapshot: If you choose this option, a new field for *Use snapshot as a source* displays. You can select the snapshot from the list.
   - Image: If you choose this option, a new field for *Use image as a source* displays. You can select the image from the list.
   - Volume: If you choose this option, a new field for *Use volume as a source* displays. You can select the volume from the list. Options to use a snapshot or a volume as the source for a volume are displayed only if there are existing snapshots or volumes.

   Type: Leave this field blank.
   Size (GB): The size of the volume in gibibytes (GiB).
   Availability Zone: Select the Availability Zone from the list. By default, this value is set to the availability zone given by the cloud provider (for example, `us-west` or `apac-south`). For some cases, it could be `nova`.
5. Click *Create Volume*.

The dashboard shows the volume on the *Volumes* tab.

3.1.7.2 Attach a volume to an instance

After you create one or more volumes, you can attach them to instances. You can attach a volume to one instance at a time.

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Volumes* category.
4. Select the volume to add to an instance and click *Manage Attachments*.
5. In the *Manage Volume Attachments* dialog box, select an instance.
6. Enter the name of the device from which the volume is accessible by the instance.

   Note

   The actual device name might differ from the volume name because of hypervisor settings.

7. Click *Attach Volume*.
   The dashboard shows the instance to which the volume is now attached and the device name.

You can view the status of a volume in the Volumes tab of the dashboard. The volume is either Available or In-Use.

Now you can log in to the instance and mount, format, and use the disk.

3.1.7.3 Detach a volume from an instance

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click the *Volumes* category.
4. Select the volume and click *Manage Attachments*.

5. Click *Detach Volume* and confirm your changes.

A message indicates whether the action was successful.

3.1.7.4 Create a snapshot from a volume

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Volumes* category.
4. Select a volume from which to create a snapshot.
5. In the *Actions* column, click *Create Snapshot*.
6. In the dialog box that opens, enter a snapshot name and a brief description.
7. Confirm your changes.
   The dashboard shows the new volume snapshot in Volume Snapshots tab.

3.1.7.5 Edit a volume

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Volumes* category.
4. Select the volume that you want to edit.
5. In the *Actions* column, click *Edit Volume*.
6. In the *Edit Volume* dialog box, update the name and description of the volume.
7. Click *Edit Volume*.

Note

You can extend a volume by using the *Extend Volume* option available in the *More* dropdown list and entering the new value for volume size.

3.1.7.6 Delete a volume

When you delete an instance, the data in its attached volumes is not deleted.

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Volumes* category.
4. Select the check boxes for the volumes that you want to delete.
5. Click *Delete Volumes* and confirm your choice.
   A message indicates whether the action was successful.

3.1.8 Create and manage shares

Shares are file storage that you provide access to instances. You can allow access to a share to a running instance or deny access to a share and allow access to it to another instance at any time. You can also delete a share. You can create snapshot from a share if the driver supports it. Only administrative users can create share types.

3.1.8.1 Create a share

1. Log in to the dashboard, choose a project, and click *Shares*.
2. Click *Create Share*.
   In the dialog box that opens, enter or select the following values.
   Share Name: Specify a name for the share.
   Description: Optionally, provide a brief description for the share.
   Share Type: Choose a share type.
   Size (GB): The size of the share in gibibytes (GiB).
   Share Protocol: Select NFS, CIFS, GlusterFS, or HDFS.
   Share Network: Choose a share network.
   Metadata: Enter metadata for the share creation if needed.
3. Click *Create Share*.

The dashboard shows the share on the *Shares* tab.

3.1.8.2 Delete a share

1. Log in to the dashboard, choose a project, and click *Shares*.
2. Select the check boxes for the shares that you want to delete.
3. Click *Delete Shares* and confirm your choice.
   A message indicates whether the action was successful.

3.1.8.3 Allow access

1. Log in to the dashboard, choose a project, and click *Shares*.
2. Go to the share that you want to allow access and choose *Manage Rules* from Actions.
3. Click *Add rule*.
   Access Type: Choose ip, user, or cert.
   Access Level: Choose read-write or read-only.
   Access To: Fill in Access To field.
4. Click *Add Rule*.
   A message indicates whether the action was successful.

3.1.8.4 Deny access

1. Log in to the dashboard, choose a project, and click *Shares*.
2. Go to the share that you want to deny access and choose *Manage Rules* from Actions.
3. Choose the rule you want to delete.
4. Click *Delete rule* and confirm your choice.
   A message indicates whether the action was successful.

3.1.8.5 Edit share metadata

1. Log in to the dashboard, choose a project, and click *Shares*.
2. Go to the share that you want to edit and choose *Edit Share Metadata* from Actions.
3. *Metadata*: To add share metadata, use key=value. To unset metadata, use key.

4. Click *Edit Share Metadata*.
   A message indicates whether the action was successful.

3.1.8.6 Edit share

1. Log in to the dashboard, choose a project, and click *Shares*.
2. Go to the share that you want to edit and choose *Edit Share* from Actions.
3. *Share Name*: Enter a new share name.
4. *Description*: Enter a new description.
5. Click *Edit Share*.
   A message indicates whether the action was successful.

3.1.8.7 Extend share

1. Log in to the dashboard, choose a project, and click *Shares*.
2. Go to the share that you want to edit and choose *Extend Share* from Actions.
3. *New Size (GB)*: Enter new size.
4. Click *Extend Share*.
   A message indicates whether the action was successful.

3.1.8.8 Create share network

1. Log in to the dashboard, choose a project, click *Shares*, and click *Share Networks*.
2. Click *Create Share Network*.
   In the dialog box that opens, enter or select the following values.
   Name: Specify a name for the share network.
   Description: Optionally, provide a brief description for the share network.
   Neutron Net: Choose a neutron network.
   Neutron Subnet: Choose a neutron subnet.
3. Click *Create Share Network*.

The dashboard shows the share network on the *Share Networks* tab.

3.1.8.9 Delete a share network

1. Log in to the dashboard, choose a project, click *Shares*, and click *Share Networks*.
2. Select the check boxes for the share networks that you want to delete.
3. Click *Delete Share Networks* and confirm your choice.
   A message indicates whether the action was successful.

3.1.8.10 Edit share network

1. Log in to the dashboard, choose a project, click *Shares*, and click *Share Networks*.
2. Go to the share network that you want to edit and choose *Edit Share Network* from Actions.
3. *Name*: Enter a new share network name.
4. *Description*: Enter a new description.
5. Click *Edit Share Network*.
   A message indicates whether the action was successful.

3.1.8.11 Create security service

1. Log in to the dashboard, choose a project, click *Shares*, and click *Security Services*.
2. Click *Create Security Service*.
   In the dialog box that opens, enter or select the following values.
   Name: Specify a name for the security service.
   DNS IP: Enter the DNS IP address.
   Server: Enter the server name.
   Domain: Enter the domain name.
   User: Enter the user name.
   Password: Enter the password.
   Confirm Password: Enter the password again to confirm.
   Type: Choose the type from Active Directory, LDAP, or Kerberos.
   Description: Optionally, provide a brief description for the security service.
3. Click *Create Security Service*.

The dashboard shows the security service on the *Security Services* tab.

3.1.8.12 Delete a security service

1. Log in to the dashboard, choose a project, click *Shares*, and click *Security Services*.
2. Select the check boxes for the security services that you want to delete.
3. Click *Delete Security Services* and confirm your choice.
   A message indicates whether the action was successful.

3.1.8.13 Edit security service

1. Log in to the dashboard, choose a project, click *Shares*, and click *Security Services*.
2. Go to the security service that you want to edit and choose *Edit Security Service* from Actions.
3. *Name*: Enter a new security service name.
4. *Description*: Enter a new description.
5. Click *Edit Security Service*.
   A message indicates whether the action was successful.

3.1.9 Launch and manage stacks

OpenStack Orchestration is a service that you can use to orchestrate multiple composite cloud applications. This service supports the use of both the Amazon Web Services (AWS) CloudFormation template format through both a Query API that is compatible with CloudFormation and the native OpenStack Heat Orchestration Template (HOT) format through a REST API.

These flexible template languages enable application developers to describe and automate the deployment of infrastructure, services, and applications. The templates enable creation of most OpenStack resource types, such as instances, floating IP addresses, volumes, security groups, and users. Once created, the resources are referred to as stacks.

The template languages are described in the Template Guide (https://docs.openstack.org/heat/latest/template_guide/).

3.1.9.1 Launch a stack

1. Log in to the dashboard.

2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Orchestration* tab and click *Stacks* category.
4. Click *Launch Stack*.
5. In the *Select Template* dialog box, specify the following values:

Template Source	Choose the source of the template from the list.
Template URL/ File/ Data	Depending on the source that you select, enter the URL, browse to the file lo-

	cation, or directly include the template.
Environment Source	Choose the source of the environment from the list. The environment files contain additional

	settings for the stack.
Environment File/ Data	Depending on the source that you select, browse to the file location, directly include the environment

6. Click *Next*.

7. In the *Launch Stack* dialog box, specify the following values:

Stack Name	Enter a name to identify the stack.
Creation Timeout (minutes)	Specify the number of minutes that can elapse before the launch of the stack times out.

Rollback On Failure	Select this check box if you want the service to roll back changes if the stack fails to launch.
Password for user "demo"	Specify the password that the default user uses when the stack is

	created.
DBUsername	Specify the name of the database user.
LinuxDistribution	Specify the Linux distribution that is used in the stack.
DBRootPassword	Specify the root password for

	the database.
KeyName	Specify the name of the key pair to use to log in to the stack.
DBName	Specify the name of the database.
DBPassword	Specify the password of

	the database.
InstanceType	Specify the flavor for the instance.

8. Click *Launch* to create a stack. The *Stacks* tab shows the stack.

After the stack is created, click on the stack name to see the following details:

Topology

The topology of the stack.

Overview

The parameters and details of the stack.

Resources

The resources used by the stack.

Events

The events related to the stack.

Template

The template for the stack.

3.1.9.2 Manage a stack

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Orchestration* tab and click *Stacks* category.

4. Select the stack that you want to update.
5. Click *Change Stack Template*.
6. In the *Select Template* dialog box, select the new template source or environment source.
7. Click *Next*.
   The *Update Stack Parameters* window appears.
8. Enter new values for any parameters that you want to update.
9. Click *Update*.

3.1.9.3 Delete a stack

When you delete a stack, you cannot undo this action.

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Orchestration* tab and click *Stacks* category.
4. Select the stack that you want to delete.
5. Click *Delete Stack*.
6. In the confirmation dialog box, click *Delete Stack* to confirm the deletion.

3.1.10 Create and manage databases

The Database service provides scalable and reliable cloud provisioning functionality for both relational and non-relational database engines. Users can quickly and easily use database features without the burden of handling complex administrative tasks.

3.1.10.1 Create a database instance

Prerequisites. Before you create a database instance, you need to configure a default datastore and make sure you have an appropriate flavor for the type of database instance you want.

1. **Configure a default datastore.**
   Because the dashboard does not let you choose a specific datastore to use with an instance, you need to configure a default datastore. The dashboard then uses the default datastore to create the instance.

   a. Add the following line to `/etc/trove/trove.conf`:

   ```
   default_datastore = DATASTORE_NAME
   ```

   Replace `DATASTORE_NAME` with the name that the administrative user set when issuing the **`trove-manage`** command to create the datastore. You can use the trove **`datastore-list`** command to display the datastores that are available in your environment.
   For example, if your MySQL data store name is set to `mysql`, your entry would look like this:

   ```
   default_datastore = mysql
   ```

   b. Restart Database services on the controller node:

   ```
   # service trove-api restart
   # service trove-taskmanager restart
   # service trove-conductor restart
   ```

2. **Verify flavor.**
   Make sure an appropriate flavor exists for the type of database instance you want.

Create database instance. Once you have configured a default datastore and verified that you have an appropriate flavor, you can create a database instance.

1. Log in to the dashboard.
2. From the CURRENT PROJECT on the *Project* tab, select the appropriate project.
3. On the *Project* tab, open the *Database* tab and click *Instances* category. This lists the instances that already exist in your environment.
4. Click *Launch Instance*.
5. In the *Launch Database* dialog box, specify the following values.
   Details
   Database Name: Specify a name for the database instance.

Flavor: Select an appropriate flavor for the instance.
Volume Size: Select a volume size. Volume size is expressed in GB.
Initialize Databases: Initial Database
Optionally provide a comma separated list of databases to create, for example:
`database1, database2, database3`
Initial Admin User: Create an initial admin user. This user will have access to all the databases you create.
Password: Specify a password associated with the initial admin user you just named.
Host: Optionally, allow the user to connect only from this host. If you do not specify a host, this user will be allowed to connect from anywhere.

6. Click the *Launch* button. The new database instance appears in the databases list.

3.1.10.2 Backup and restore a database

You can use Database services to backup a database and store the backup artifact in the Object Storage service. Later on, if the original database is damaged, you can use the backup artifact to restore the database. The restore process creates a database instance.

This example shows you how to back up and restore a MySQL database.

3.1.10.2.1 To backup the database instance

1. Log in to the dashboard.
2. From the CURRENT PROJECT on the *Project* tab, select the appropriate project.
3. On the *Project* tab, open the *Database* tab and click *Instances* category. This displays the existing instances in your system.
4. Click *Create Backup*.
5. In the *Backup Database* dialog box, specify the following values:
   Name
   Specify a name for the backup.
   Database Instance
   Select the instance you want to back up.
6. Click *Backup*. The new backup appears in the backup list.

3.1.10.2.2 To restore a database instance

Now assume that your original database instance is damaged and you need to restore it. You do the restore by using your backup to create a new database instance.

1. Log in to the dashboard.
2. From the CURRENT PROJECT on the *Project* tab, select the appropriate project.
3. On the *Project* tab, open the *Database* tab and click *Backups* category. This lists the available backups.
4. Check the backup you want to use and click *Restore Backup*.
5. In the *Launch Database* dialog box, specify the values you want for the new database instance.
6. Click the *Restore From Database* tab and make sure that this new instance is based on the correct backup.
7. Click *Launch*.
   The new instance appears in the database instances list.

3.1.10.3 Update a database instance

You can change various characteristics of a database instance, such as its volume size and flavor.

3.1.10.3.1 To change the volume size of an instance

1. Log in to the dashboard.
2. From the CURRENT PROJECT on the *Project* tab, select the appropriate project.
3. On the *Project* tab, open the *Database* tab and click *Instances* category. This displays the existing instances in your system.
4. Check the instance you want to work with. In the *Actions* column, expand the drop down menu and select *Resize Volume*.
5. In the *Resize Database Volume* dialog box, fill in the *New Size* field with an integer indicating the new size you want for the instance. Express the size in GB, and note that the new size must be larger than the current size.
6. Click *Resize Database Volume*.

3.1.10.3.2 To change the flavor of an instance

1. Log in to the dashboard.
2. From the CURRENT PROJECT on the *Project* tab, select the appropriate project.
3. On the *Project* tab, open the *Database* tab and click *Instances* category. This displays the existing instances in your system.
4. Check the instance you want to work with. In the *Actions* column, expand the drop down menu and select *Resize Instance*.
5. In the *Resize Database Instance* dialog box, expand the drop down menu in the *New Flavor* field. Select the new flavor you want for the instance.
6. Click *Resize Database Instance*.

3.1.11 View and manage load balancers v2

Load-Balancer-as-a-Service (LBaaS) enables networking to distribute incoming requests evenly among designated instances. This distribution ensures that the workload is shared predictably among instances and enables more effective use of system resources. Use one of these load-balancing methods to distribute incoming requests:

- Round robin: Rotates requests evenly between multiple instances.
- Source IP: Requests from a unique source IP address are consistently directed to the same instance.
- Least connections: Allocates requests to the instance with the least number of active connections.

As an end user, you can create and manage load balancers and related objects for users in various projects. You can also delete load balancers and related objects.

LBaaS v2 has several new concepts to understand:

Load balancer

The load balancer occupies a neutron network port and has an IP address assigned from a subnet.

Listener

Each port that listens for traffic on a particular load balancer is configured separately and tied to the load balancer. Multiple listeners can be associated with the same load balancer.

Pool

A pool is a group of hosts that sits behind the load balancer and serves traffic through the load balancer.

Member

Members are the actual IP addresses that receive traffic from the load balancer. Members are associated with pools.

Health monitor

Members may go offline from time to time and health monitors diverts traffic away from members that are not responding properly. Health monitors are associated with pools.

3.1.11.1 View existing load balancers

1. Log in to the OpenStack dashboard.
2. On the *Project* tab, open the *Network* tab, and click the *Load Balancers* category.
   This view shows the list of existing load balancers. To view details of any of the load balancers, click on the specific load balancer.

3.1.11.2 Create a load balancer

1. Log in to the OpenStack dashboard.
2. On the *Project* tab, open the *Network* tab, and click the *Load Balancers* category.
3. Click the *Create Load Balancer* button.
   Use the concepts described in the overview section to fill in the necessary information about the load balancer you want to create.
   Keep in mind, the health checks routinely run against each instance within a target load balancer and the result of the health check is used to determine if the instance receives new connections.

Note

A message indicates whether the action succeeded.

3.1.11.3 Delete a load balancer

- Select the load balancer you want to delete and click the *Delete Load Balancer* button. To be deleted successfully, a load balancer must not have any listeners or pools associated with it. The delete action is also available in the *Actions* column for the individual load balancers.

3.1.12 Supported Browsers

Horizon is primarily tested and supported on the latest version of Firefox, the latest version of Chrome, and IE9+. Issues related to Safari and Opera will also be considered.

This page aims to informally document what that means for different releases, everyone is warmly encouraged to update this page based on the versions they've tested with.

Legend:

- Very good: Very well tested, should work as expected
- Good: Moderately tested, should look nice and work fine, maybe a few visual hiccups
- Poor: Doesn't look good
- Broken: Essential functionality not working (link to bug in the notes)
- No: Not supported

3.1.12.1 Kilo

	Status	Notes
Firefox	Very good	31+. (Earlier versions?)
Firefox ESR	Very good	31+

	Status	Notes
Chrome	Very good	43.0.2357.81
IE 11	Good?	
IE 10	Good?	
IE 9	?	
IE 8 and below	Not supported.	
Safari	?	
Opera	?	

3.1.12.2 Juno

	Status	Notes
Firefox	Very good	31 + . (Earlier versions?)

	Status	Notes
Firefox ESR	?	
Chrome	Very good	Versions?
IE 11	Good	? Open IE Bugs (https://bugs.launchpad.net/horizon/+bugs?field.tag=ie).
IE 10	Good	? Open IE Bugs (https://bugs.launchpad.net/horizon/+bugs?field.tag=ie).
IE 9	?	Open IE Bugs (https://bugs.launchpad.net/horizon/+bugs?field.tag=ie).

	Status	Notes
IE 8 and below	Not supported.	No.
Safari	?	
Opera	?	

3.1.12.3 Icehouse

	Status	Notes
Firefox	Very good	Versions?
Firefox ESR	Very good	Windows 24.7.0 ESR
Chrome	Very good	Windows Version 36, RHEL version 27.0

	Status	Notes
Chromium	Very good	version 31.0
IE 11	?	
IE 10	?	
IE 9	?	
IE 8 and below	?	
Safari	?	
Opera	?	

4 Keystone User Guide

This section contains the documentation for end-users of keystone.

4.1 User Documentation

An end user can find the specific API documentation here, OpenStack's Identity API (https://developer.openstack.org/api-ref/identity/v3).

Note

Following are some API examples using curl. These examples are not automatically generated. They can be outdated as things change and are subject to regular updates and changes.

4.1.1 API Examples using Curl

4.1.1.1 v3 API Examples Using Curl

4.1.1.1.1 Tokens

4.1.1.1.1.1 Default scope

Get a token with default scope (may be unscoped):

```
curl -i \
  -H "Content-Type: application/json" \
  -d '
{ "auth": {
    "identity": {
      "methods": ["password"],
      "password": {
        "user": {
          "name": "admin",
```

```
          "domain": { "id": "default" },
          "password": "adminpwd"
        }
      }
    }
  }
}' \
  "http://localhost:5000/v3/auth/tokens" ; echo
```

Example response:

```
HTTP/1.1 201 Created
X-Subject-Token: MIIFvgY...
Vary: X-Auth-Token
Content-Type: application/json
Content-Length: 1025
Date: Tue, 10 Jun 2014 20:55:16 GMT

{
  "token": {
    "methods": ["password"],
    "roles": [{
      "id": "9fe2ff9ee4384b1894a90878d3e92bab",
      "name": "_member_"
    }, {
      "id": "c703057be878458588961ce9a0ce686b",
      "name": "admin"
    }],
    "expires_at": "2014-06-10T2:55:16.806001Z",
    "project": {
      "domain": {
        "id": "default",
        "name": "Default"
      },
      "id": "8538a3f13f9541b28c2620eb19065e45",
      "name": "admin"
    },
    "catalog": [{
      "endpoints": [{
        "url": "http://localhost:3537/v2.0",
        "region": "RegionOne",
        "interface": "admin",
        "id": "29beb2f1567642eb810b042b6719ea88"
      }, {
        "url": "http://localhost:5000/v2.0",
        "region": "RegionOne",
        "interface": "internal",
        "id": "8707e3735d4415c97ae231b4841eb1c"
```

```
      }, {
        "url": "http://localhost:5000/v2.0",
        "region": "RegionOne",
        "interface": "public",
        "id": "ef303187fc8d41668f25199c298396a5"
      }],
      "type": "identity",
      "id": "bd73972c0e14fb69bae8ff76e112a90",
      "name": "keystone"
    }],
    "extras": {},
    "user": {
      "domain": {
        "id": "default",
        "name": "Default"
      },
      "id": "3ec3164f750146be97f21559ee4d9c51",
      "name": "admin"
    },
    "audit_ids": ["yRt0UrxJSs6-WYJgwEMMmg"],
    "issued_at": "201406-10T20:55:16.806027Z"
  }
}
```

4.1.1.1.1.2 Project-scoped

Get a project-scoped token:

```
curl -i \
  -H "Content-Type: application/json" \
  -d '
{ "auth": {
    "identity": {
      "methods": ["password"],
      "password": {
        "user": {
          "name": "admin",
          "domain": { "id": "default" },
          "password": "adminpwd"
        }
      }
    },
    "scope": {
      "project": {
        "name": "demo",
        "domain": { "id": "default" }
```

```
      }
    }
  }
}' \
  "http://localhost:5000/v3/auth/tokens" ; echo
```

Example response:

```
HTTP/1.1 201 Created
X-Subject-Token: MIIFfQ...
Vary: X-Auth-Token
Content-Type: application/json
Content-Length: 960
Date: Tue, 10 Jun 2014 20:40:14 GMT

{
  "token": {
    "audit_ids": ["ECwrVNWbSCqmEgPnu0YCRw"],
    "methods": ["password"],
    "roles": [{
      "id": "c703057be878458588961ce9a0ce686b",
      "name": "admin"
    }],
    "expires_at": "2014-06-10T21:40:14.360795Z",
    "project": {
      "domain": {
        "id": "default",
        "name": "Default"
      },
      "id": "3d4c2c82bd5948f0bcab0cf3a7c9b48c",
      "name": "demo"
    },
    "catalog": [{
      "endpoints": [{
        "url": "http://localhost:35357/v2.0",
        "region": "RegionOne",
        "interface": "admin",
        "id": "29beb2f1567642eb810b042b6719ea88"
      }, {
        "url": "http://localhost:5000/v2.0",
        "region": "RegionOne",
        "interface": "internal",
        "id": "87057e3735d4415c97ae231b4841eb1c"
      }, {
        "url": "http://localhost:5000/v2.0",
        "region": "RegionOne",
        "interface": "public",
        "id": "ef303187fc8d41668f25199c298396a5"
```

```
      }],
      "type": "identity",
      "id": "bd7397d2c0e14fb69bae8ff76e112a90",
      "name": "keystone"
    }],
    "extras": {},
    "user": {
      "domain": {
        "id": "default",
        "name": "Default"
      },
      "id": "3ec3164f750146be97f21559ee4d9c51",
      "name": "admin"
    },
    "issued_at": "2014-06-10T20:40:14.360822Z"
  }
}
```

4.1.1.1.1.3 Domain-Scoped

Get a domain-scoped token:

Note

Ensure you have a role-assignment on the domain first.

```
curl -i \
  -H "Content-Type: application/json" \
  -d '
{ "auth": {
    "identity": {
      "methods": ["password"],
      "password": {
        "user": {
          "name": "admin",
          "domain": { "id": "default" },
          "password": "adminpwd"
        }
      }
    },
    "scope": {
      "domain": {
        "id": "default"
      }
```

```
    }
  }
}' \
  "http://localhost:5000/v3/auth/tokens" ; echo
```

Example response:

```
HTTP/1.1 201 Created
X-Subject-Token: MIIFNg...
Vary: X-Auth-Token
Content-Type: application/json
Content-Length: 889
Date: Tue, 10 Jun 2014 20:52:59 GMT

{
  "token": {
    "domain": {
      "id": "default",
      "name": "Default"
    },
    "methods": ["password"],
    "roles": [{
      "id": "c703057be878458588961ce9a0ce686b",
      "name": "admin"
    }],
    "expires_at": "2014-06-10T21:52:58.852167Z",
    "catalog": [{
      "endpoints": [{
        "url": "http://localhost:35357/v2.0",
        "region": "RegionOne",
        "interface": "admin",
        "id": "29beb2f1567642eb810b042b6719ea88"
      }, {
        "url": "http://localhost:5000/v2.0",
        "region": "RegionOne",
        "interface": "internal",
        "id": "87057e3735d4415c97ae231b4841eb1c"
      }, {
        "url": "http://localhost:5000/v2.0",
        "region": "RegionOne",
        "interface": "public",
        "id": "ef303187fc8d41668f25199c298396a5"
      }],
      "type": "identity",
      "id": "bd7397d2c0e14fb69bae8ff76e112a90",
      "name": "keystone"
    }],
    "extras": {},
```

```
    "user": {
      "domain": {
        "id": "default",
        "name": "Default"
      },
      "id": "3ec3164f750146be97f21559ee4d9c51",
      "name": "admin"
    },
    "audit_ids": ["Xpa6Uyn-T9S6mTREudUH3w"],
    "issued_at": "2014-06-10T20:52:58.852194Z"
  }
}
```

4.1.1.1.1.4 Getting a token from a token

Get a token from a token:

```
curl -i \
  -H "Content-Type: application/json" \
  -d '
{ "auth": {
    "identity": {
      "methods": ["token"],
      "token": {
        "id": "'$OS_TOKEN'"
      }
    }
  }
}' \
  "http://localhost:5000/v3/auth/tokens" ; echo
```

Example response:

```
HTTP/1.1 201 Created
X-Subject-Token: MIIFxw...
Vary: X-Auth-Token
Content-Type: application/json
Content-Length: 1034
Date: Tue, 10 Jun 2014 21:00:05 GMT

{
  "token": {
    "methods": ["token", "password"],
    "expires_at": "2015-05-28T07:43:44.808209Z",
    "extras": {},
```

```
    "user": {
      "domain": {
        "id": "default",
        "name": "Default"
      },
      "id": "753867c25c3340ffad1abc22d488c31a",
      "name": "admin"
    },
    "audit_ids": ["ZE0OPSuzTmCXHo0eIOYltw",
      "xxIQCkHOQOywL0oY6CTppQ"
    ],
    "issued_at": "2015-05-28T07:19:23.763532Z"
  }
}
```

Note

If a scope was included in the request body, then this would get a token with the new scope.

4.1.1.1.1.5 DELETE /v3/auth/tokens

Revoke a token:

```
curl -i -X DELETE \
  -H "X-Auth-Token: $OS_TOKEN" \
  -H "X-Subject-Token: $OS_TOKEN" \
  "http://localhost:5000/v3/auth/tokens"
```

If there's no error then the response is empty.

4.1.1.1.2 Domains

4.1.1.1.2.1 GET /v3/domains

List domains:

```
curl -s \
  -H "X-Auth-Token: $OS_TOKEN" \
  "http://localhost:5000/v3/domains" | python -mjson.tool
```

Example response:

```
{
    "domains": [
        {
            "description": "Owns users and tenants (i.e. projects) available on Identity
 API v2.",
            "enabled": true,
            "id": "default",
            "links": {
                "self": "http://identity-server:5000/v3/domains/default"
            },
            "name": "Default"
        }
    ],
    "links": {
        "next": null,
        "previous": null,
        "self": "http://identity-server:5000/v3/domains"
    }
}
```

4.1.1.1.2.2 POST /v3/domains

Create a domain:

```
curl -s \
  -H "X-Auth-Token: $OS_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{ "domain": { "name": "newdomain"}}' \
  "http://localhost:5000/v3/domains" | python -mjson.tool
```

Example response:

```
{
    "domain": {
        "enabled": true,
        "id": "3a5140aecd974bf08041328b53a62458",
        "links": {
            "self": "http://identity-server:5000/v3/
domains/3a5140aecd974bf08041328b53a62458"
        },
        "name": "newdomain"
    }
}
```

4.1.1.1.3 Projects

4.1.1.1.3.1 GET /v3/projects

List projects:

```
curl -s \
 -H "X-Auth-Token: $OS_TOKEN" \
 "http://localhost:5000/v3/projects" | python -mjson.tool
```

Example response:

```
{
    "links": {
        "next": null,
        "previous": null,
        "self": "http://localhost:5000/v3/projects"
    },
    "projects": [
        {
            "description": null,
            "domain_id": "default",
            "enabled": true,
            "id": "3d4c2c82bd5948f0bcab0cf3a7c9b48c",
            "links": {
                "self": "http://localhost:5000/v3/
projects/3d4c2c82bd5948f0bcab0cf3a7c9b48c"
            },
            "name": "demo"
        }
    ]
}
```

4.1.1.1.3.2 PATCH /v3/projects/{id}

Disable a project:

```
curl -s -X PATCH \
  -H "X-Auth-Token: $OS_TOKEN" \
  -H "Content-Type: application/json" \
  -d '
{
  "project": {
      "enabled": false
    }
```

```
}'\
  "http://localhost:5000/v3/projects/$PROJECT_ID"  | python -mjson.tool
```

Example response:

```
{
    "project": {
        "description": null,
        "domain_id": "default",
        "enabled": false,
        "extra": {},
        "id": "3d4c2c82bd5948f0bcab0cf3a7c9b48c",
        "links": {
            "self": "http://localhost:5000/v3/projects/3d4c2c82bd5948f0bcab0cf3a7c9b48c"
        },
        "name": "demo"
    }
}
```

4.1.1.1.4 GET /v3/services

List the services:

```
curl -s \
  -H "X-Auth-Token: $OS_TOKEN" \
  "http://localhost:5000/v3/services" | python -mjson.tool
```

Example response:

```
{
    "links": {
        "next": null,
        "previous": null,
        "self": "http://localhost:5000/v3/services"
    },
    "services": [
        {
            "description": "Keystone Identity Service",
            "enabled": true,
            "id": "bd7397d2c0e14fb69bae8ff76e112a90",
            "links": {
                "self": "http://localhost:5000/v3/services/
bd7397d2c0e14fb69bae8ff76e112a90"
            },
            "name": "keystone",
            "type": "identity"
        }
```

```
    ]
}
```

4.1.1.1.5 GET /v3/endpoints

List the endpoints:

```
curl -s \
 -H "X-Auth-Token: $OS_TOKEN" \
 "http://localhost:5000/v3/endpoints" | python -mjson.tool
```

Example response:

```
{
    "endpoints": [
        {
            "enabled": true,
            "id": "29beb2f1567642eb810b042b6719ea88",
            "interface": "admin",
            "links": {
                "self": "http://localhost:5000/v3/
endpoints/29beb2f1567642eb810b042b6719ea88"
            },
            "region": "RegionOne",
            "service_id": "bd7397d2c0e14fb69bae8ff76e112a90",
            "url": "http://localhost:35357/v2.0"
        }
    ],
    "links": {
        "next": null,
        "previous": null,
        "self": "http://localhost:5000/v3/endpoints"
    }
}
```

4.1.1.1.6 Users

4.1.1.1.6.1 GET /v3/users

List users:

```
curl -s \
 -H "X-Auth-Token: $OS_TOKEN" \
```

```
 "http://localhost:5000/v3/users" | python -mjson.tool
```

4.1.1.1.6.2 POST /v3/users

Create a user:

```
curl -s \
 -H "X-Auth-Token: $OS_TOKEN" \
 -H "Content-Type: application/json" \
 -d '{"user": {"name": "newuser", "password": "changeme"}}' \
 "http://localhost:5000/v3/users" | python -mjson.tool
```

Example response:

```
{
    "user": {
        "domain_id": "default",
        "enabled": true,
        "id": "ec8fc20605354edd91873f2d66bf4fc4",
        "links": {
            "self": "http://identity-server:5000/v3/users/
ec8fc20605354edd91873f2d66bf4fc4"
        },
        "name": "newuser"
    }
}
```

4.1.1.1.6.3 GET /v3/users/{user_id}

Show details for a user:

```
USER_ID=ec8fc20605354edd91873f2d66bf4fc4

curl -s \
 -H "X-Auth-Token: $OS_TOKEN" \
 "http://localhost:5000/v3/users/$USER_ID" | python -mjson.tool
```

Example response:

```
{
    "user": {
        "domain_id": "default",
        "enabled": true,
        "id": "ec8fc20605354edd91873f2d66bf4fc4",
        "links": {
            "self": "http://localhost:5000/v3/users/ec8fc20605354edd91873f2d66bf4fc4"
```

```
        },
        "name": "newuser"
    }
}
```

4.1.1.1.6.4 POST /v3/users/{user_id}/password

Change password using the default policy; this can be done as the user:

```
USER_ID=b7793000f8d84c79af4e215e9da78654
ORIG_PASS=userpwd
NEW_PASS=newuserpwd

curl \
 -H "X-Auth-Token: $OS_TOKEN" \
 -H "Content-Type: application/json" \
 -d '{ "user": {"password": "'$NEW_PASS'", "original_password": "'$ORIG_PASS'"} }' \
 "http://localhost:5000/v3/users/$USER_ID/password"
```

Note

This command will not print anything if the request was successful.

4.1.1.1.6.5 PATCH /v3/users/{user_id}

Reset password using the default policy; this requires admin:

```
USER_ID=b7793000f8d84c79af4e215e9da78654
NEW_PASS=newuserpwd

curl -s -X PATCH \
 -H "X-Auth-Token: $OS_TOKEN" \
 -H "Content-Type: application/json" \
 -d '{ "user": {"password": "'$NEW_PASS'"} }' \
 "http://localhost:5000/v3/users/$USER_ID" | python -mjson.tool
```

Example response:

```
{
    "user": {
        "default_project_id": "3d4c2c82bd5948f0bcab0cf3a7c9b48c",
        "domain_id": "default",
        "email": "demo@example.com",
        "enabled": true,
```

```
        "extra": {
            "email": "demo@example.com"
        },
        "id": "269348fdd9374b8885da1418e0730af1",
        "links": {
            "self": "http://localhost:5000/v3/users/269348fdd9374b8885da1418e0730af1"
        },
        "name": "demo"
    }
}
```

4.1.1.1.7 PUT /v3/projects/{project_id}/groups/{group_id}/roles/{role_id}

Create group role assignment on project:

```
curl -s -X PUT \
 -H "X-Auth-Token: $OS_TOKEN" \
 "http://localhost:5000/v3/projects/$PROJECT_ID/groups/$GROUP_ID/roles/$ROLE_ID" |
   python -mjson.tool
```

There's no data in the response if the operation is successful.

4.1.1.1.8 POST /v3/OS-TRUST/trusts

Create a trust:

```
curl -s \
 -H "X-Auth-Token: $OS_TOKEN" \
 -H "Content-Type: application/json" \
 -d '
{ "trust": {
    "expires_at": "2014-12-30T23:59:59.999999Z",
    "impersonation": false,
    "project_id": "'$PROJECT_ID'",
    "roles": [
        { "name": "admin" }
      ],
    "trustee_user_id": "'$DEMO_USER_ID'",
    "trustor_user_id": "'$ADMIN_USER_ID'"
}}'\
 "http://localhost:5000/v3/OS-TRUST/trusts" | python -mjson.tool
```

Example response:

```
{
    "trust": {
```

```
        "expires_at": "2014-12-30T23:59:59.999999Z",
        "id": "394998fa61f14736b1f0c1f322882949",
        "impersonation": false,
        "links": {
            "self": "http://localhost:5000/v3/OS-TRUST/
trusts/394998fa61f14736b1f0c1f322882949"
        },
        "project_id": "3d4c2c82bd5948f0bcab0cf3a7c9b48c",
        "remaining_uses": null,
        "roles": [
            {
                "id": "c703057be878458588961ce9a0ce686b",
                "links": {
                    "self": "http://localhost:5000/v3/roles/
c703057be878458588961ce9a0ce686b"
                },
                "name": "admin"
            }
        ],
        "roles_links": {
            "next": null,
            "previous": null,
            "self": "http://localhost:5000/v3/OS-TRUST/
trusts/394998fa61f14736b1f0c1f322882949/roles"
        },
        "trustee_user_id": "269348fdd9374b8885da1418e0730af1",
        "trustor_user_id": "3ec3164f750146be97f21559ee4d9c51"
    }
}
```

4.1.1.2 Service API Examples Using Curl

The service API is defined to be a subset of the Admin API and runs on port 5000 by default.

4.1.1.2.1 GET /

This call is identical to that documented for the Admin API, except that it uses port 5000 by default instead of port 35357:

```
$ curl "http://0.0.0.0:5000"
```

or:

```
$ curl "http://0.0.0.0:5000/v2.0/"
```

See the *Section 4.1.1.3, "Admin API Examples Using Curl"* for more info.

4.1.1.2.2 GET /extensions

This call is identical to that documented for the Admin API.

4.1.1.2.3 POST /tokens

This call is identical to that documented for the Admin API.

4.1.1.2.4 GET /tenants

List all of the tenants your token can access:

```
$ curl -H "X-Auth-Token:887665443383838" \
   "http://localhost:5000/v2.0/tenants"
```

Returns:

```
{
    "tenants_links": [],
    "tenants": [
        {
            "enabled": true,
            "description": "None",
            "name": "customer-x",
            "id": "1"
        }
    ]
}
```

4.1.1.3 Admin API Examples Using Curl

These examples assume a default port value of 35357, and depend on the `sampledata` bundled with keystone.

4.1.1.3.1 GET /

Discover API version information, links to documentation (PDF, HTML, WADL), and supported media types:

```
$ curl "http://0.0.0.0:35357"
```

```
{
    "versions": {
        "values": [
            {
                "id": "v3.4",
                "links": [
                    {
                        "href": "http://127.0.0.1:35357/v3/",
                        "rel": "self"
                    }
                ],
                "media-types": [
                    {
                        "base": "application/json",
                        "type": "application/vnd.openstack.identity-v3+json"
                    }
                ],
                "status": "stable",
                "updated": "2015-03-30T00:00:00Z"
            },
            {
                "id": "v2.0",
                "links": [
                    {
                        "href": "http://127.0.0.1:35357/v2.0/",
                        "rel": "self"
                    },
                    {
                        "href": "https://docs.openstack.org/",
                        "rel": "describedby",
                        "type": "text/html"
                    }
                ],
                "media-types": [
                    {
                        "base": "application/json",
                        "type": "application/vnd.openstack.identity-v2.0+json"
                    }
                ],
                "status": "stable",
```

```
                "updated": "2014-04-17T00:00:00Z"
            }
        ]
    }
}
```

```
$ curl "http://0.0.0.0:35357/v2.0/"
```

Returns:

```
{
    "version": {
        "id": "v2.0",
        "links": [
            {
                "href": "http://127.0.0.1:35357/v2.0/",
                "rel": "self"
            },
            {
                "href": "https://docs.openstack.org/",
                "rel": "describedby",
                "type": "text/html"
            }
        ],
        "media-types": [
            {
                "base": "application/json",
                "type": "application/vnd.openstack.identity-v2.0+json"
            }
        ],
        "status": "stable",
        "updated": "2014-04-17T00:00:00Z"
    }
}
```

4.1.1.3.2 GET /extensions

Discover the API extensions enabled at the endpoint:

```
$ curl "http://localhost:35357/v2.0/extensions/"
```

Returns:

```
{
    "extensions":{
        "values":[]
    }
```

```
}
```

4.1.1.3.3 POST /tokens

Authenticate by exchanging credentials for an access token:

```
$ curl -d '
   {"auth": {
      "tenantName": "customer-x",
      "passwordCredentials": {
        "username": "joeuser",
        "password": "secret"
       }
     }
   }' \
   -H "Content-type: application/json" \
   "http://localhost:35357/v2.0/tokens"
```

Returns:

```
{
    "access":{
        "token":{
            "expires":"2012-02-05T00:00:00",
            "id":"887665443383838",
            "tenant":{
                "id":"1",
                "name":"customer-x"
            }
        },
        "serviceCatalog":[
            {
                "endpoints":[
                {
                    "adminURL":"http://swift.admin-nets.local:8080/",
                    "region":"RegionOne",
                    "internalURL":"http://127.0.0.1:8080/v1/AUTH_1",
                    "publicURL":"http://swift.publicinternets.com/v1/AUTH_1"
                }
                ],
                "type":"object-store",
                "name":"swift"
            },
            {
                "endpoints":[
                {
```

```
                    "adminURL":"http://cdn.admin-nets.local/v1.1/1",
                    "region":"RegionOne",
                    "internalURL":"http://127.0.0.1:7777/v1.1/1",
                    "publicURL":"http://cdn.publicinternets.com/v1.1/1"
                }
                ],
                "type":"object-store",
                "name":"cdn"
            }
        ],
        "user":{
            "id":"1",
            "roles":[
                {
                "tenantId":"1",
                "id":"3",
                "name":"Member"
                }
            ],
            "name":"joeuser"
        }
    }
}
```

Note

Take note of the value `['access']['token']['id']` value produced here (`887665443383838`, above), as you can use it in the calls below.

4.1.1.3.4 GET /tokens/{token_id}

Note

This call refers to a token known to be valid, `887665443383838` in this case.

Validate a token:

```
$ curl -H "X-Auth-Token:999888777666" \
   "http://localhost:35357/v2.0/tokens/887665443383838"
```

If the token is valid, returns:

```
{
```

```
    "access":{
        "token":{
            "expires":"2012-02-05T00:00:00",
            "id":"887665443383838",
            "tenant":{
                "id":"1",
                "name":"customer-x"
            }
        },
        "user":{
            "name":"joeuser",
            "tenantName":"customer-x",
            "id":"1",
            "roles":[
                {
                    "serviceId":"1",
                    "id":"3",
                    "name":"Member"
                }
            ],
            "tenantId":"1"
        }
    }
}
```

4.1.1.3.5 HEAD /tokens/{token_id}

This is a high-performance variant of the GET call documented above, which by definition, returns no response body:

```
$ curl -I -H "X-Auth-Token:999888777666" \
   "http://localhost:35357/v2.0/tokens/887665443383838"
```

This returns `200`, indicating the token is valid:

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: None
Date: Tue, 08 Nov 2011 23:07:44 GMT
```

4.1.1.3.6 GET /tokens/{token_id}/endpoints

List all endpoints for a token:

```
$ curl -H "X-Auth-Token:999888777666" \
```

```
   "http://localhost:35357/v2.0/tokens/887665443383838/endpoints"
```

Returns:

```
{
    "endpoints_links": [
        {
            "href": "http://127.0.0.1:35357/tokens/887665443383838/
endpoints?'marker=5&limit=10'",
            "rel": "next"
        }
    ],
    "endpoints": [
        {
            "internalURL": "http://127.0.0.1:8080/v1/AUTH_1",
            "name": "swift",
            "adminURL": "http://swift.admin-nets.local:8080/",
            "region": "RegionOne",
            "tenantId": 1,
            "type": "object-store",
            "id": 1,
            "publicURL": "http://swift.publicinternets.com/v1/AUTH_1"
        },
        {
            "internalURL": "http://localhost:8774/v1.0",
            "name": "nova_compat",
            "adminURL": "http://127.0.0.1:8774/v1.0",
            "region": "RegionOne",
            "tenantId": 1,
            "type": "compute",
            "id": 2,
            "publicURL": "http://nova.publicinternets.com/v1.0/"
        },
        {
            "internalURL": "http://localhost:8774/v1.1",
            "name": "nova",
            "adminURL": "http://127.0.0.1:8774/v1.1",
            "region": "RegionOne",
            "tenantId": 1,
            "type": "compute",
            "id": 3,
            "publicURL": "http://nova.publicinternets.com/v1.1/
        },
        {
            "internalURL": "http://127.0.0.1:9292/v1.1/",
            "name": "glance",
            "adminURL": "http://nova.admin-nets.local/v1.1/",
            "region": "RegionOne",
```

```
            "tenantId": 1,
            "type": "image",
            "id": 4,
            "publicURL": "http://glance.publicinternets.com/v1.1/"
        },
        {
            "internalURL": "http://127.0.0.1:7777/v1.1/1",
            "name": "cdn",
            "adminURL": "http://cdn.admin-nets.local/v1.1/1",
            "region": "RegionOne",
            "tenantId": 1,
            "type": "object-store",
            "id": 5,
            "publicURL": "http://cdn.publicinternets.com/v1.1/1"
        }
    ]
}
```

4.1.1.3.7 GET /tenants

List all of the tenants in the system (requires an Admin `X-Auth-Token`):

```
$ curl -H "X-Auth-Token:999888777666" \
   "http://localhost:35357/v2.0/tenants"
```

Returns:

```
{
    "tenants_links": [],
    "tenants": [
        {
            "enabled": false,
            "description": "None",
            "name": "project-y",
            "id": "3"
        },
        {
            "enabled": true,
            "description": "None",
            "name": "ANOTHER:TENANT",
            "id": "2"
        },
        {
            "enabled": true,
            "description": "None",
```

```
            "name": "customer-x",
            "id": "1"
        }
    ]
}
```

4.1.1.3.8 GET /tenants/{tenant_id}

Retrieve information about a tenant, by tenant ID:

```
$ curl -H "X-Auth-Token:999888777666" \
   "http://localhost:35357/v2.0/tenants/1"
```

Returns:

```
{
    "tenant":{
        "enabled":true,
        "description":"None",
        "name":"customer-x",
        "id":"1"
    }
}
```

4.1.1.3.9 GET /tenants/{tenant_id}/users/{user_id}/roles

List the roles a user has been granted on a tenant:

```
$ curl -H "X-Auth-Token:999888777666" \
   "http://localhost:35357/v2.0/tenants/1/users/1/roles"
```

Returns:

```
{
    "roles_links":[],
    "roles":[
        {
            "id":"3",
            "name":"Member"
        }
    ]
}
```

4.1.1.3.10 GET /users/{user_id}

Retrieve information about a user, by user ID:

```
$ curl -H "X-Auth-Token:999888777666" \
   "http://localhost:35357/v2.0/users/1"
```

Returns:

```
{
    "user":{
        "tenantId":"1",
        "enabled":true,
        "id":"1",
        "name":"joeuser"
    }
}
```

4.1.1.3.11 GET /tokens/revoked

Get the revocation list:

```
curl -s -H "X-Auth-Token: $OS_TOKEN" \
  "http://localhost:35357/v2.0/tokens/revoked" |
 jq -r .signed |
 openssl cms -verify \
  -certfile /etc/keystone/ssl/certs/signing_cert.pem \
  -CAfile /etc/keystone/ssl/certs/ca.pem \
  -inform PEM \
  -nosmimecap -nodetach -nocerts -noattr 2>/dev/null |
 python -m json.tool
```

Example response:

```
{
    "revoked": [
        {
            "expires": "2014-06-10T21:40:14Z",
            "id": "e6e2b5c9092751f88d2bcd30b09777a9"
        },
        {
            "expires": "2014-06-10T21:47:29Z",
            "id": "883ef5d610bd1c68fbaa8ac528aa9f17"
        },
        {
            "expires": "2014-06-10T21:51:52Z",
```

```
            "id": "41775ff4838f8f406b7bad28bea0dde6"
        }
    ]
}
```

5 Magnum User Documentation

5.1 Introduction

This guide is intended for users who use Magnum to deploy and manage clusters of hosts for a Container Orchestration Engine. It describes the infrastructure that Magnum creates and how to work with them.

Section 1-3 describe Magnum itself, including an overview, the CLI and Horizon interface. Section 4-9 describe the Container Orchestration Engine (COE) supported along with a guide on how to select one that best meets your needs and how to develop a driver for a new COE. Section 10-15 describe the low level OpenStack infrastructure that is created and managed by Magnum to support the COE's.

5.2 Terminology

Cluster (previously Bay)

A cluster is the construct in which Magnum launches container orchestration engines. After a cluster has been created the user is able to add containers to it either directly, or in the case of the Kubernetes container orchestration engine within pods - a logical construct specific to that implementation. A cluster is created based on a ClusterTemplate.

ClusterTemplate (previously BayModel)

A ClusterTemplate in Magnum is roughly equivalent to a flavor in Nova. It acts as a template that defines options such as the container orchestration engine, keypair and image for use when Magnum is creating clusters using the given ClusterTemplate.

Container Orchestration Engine (COE)

A container orchestration engine manages the lifecycle of one or more containers, logically represented in Magnum as a cluster. Magnum supports a number of container orchestration engines, each with their own pros and cons, including Docker Swarm, Kubernetes, and Mesos.

5.3 Overview

Magnum is an OpenStack API service developed by the OpenStack Containers Team making container orchestration engines (COE) such as Docker Swarm, Kubernetes and Apache Mesos available as the first class resources in OpenStack.

Magnum uses Heat to orchestrate an OS image which contains Docker and COE and runs that image in either virtual machines or bare metal in a cluster configuration.

Magnum offers complete life-cycle management of COEs in an OpenStack environment, integrated with other OpenStack services for a seamless experience for OpenStack users who wish to run containers in an OpenStack environment.

Following are few salient features of Magnum:

- Standard API based complete life-cycle management for Container Clusters
- Multi-tenancy for container clusters
- Choice of COE: Kubernetes, Swarm, Mesos, DC/OS
- Choice of container cluster deployment model: VM or Bare-metal
- Keystone-based multi-tenant security and auth management
- Neutron based multi-tenant network control and isolation
- Cinder based volume service for containers
- Integrated with OpenStack: SSO experience for cloud users
- Secure container cluster access (TLS enabled)

More details: Magnum Project Wiki (https://wiki.openstack.org/wiki/Magnum)

5.4 ClusterTemplate

A ClusterTemplate (previously known as BayModel) is a collection of parameters to describe how a cluster can be constructed. Some parameters are relevant to the infrastructure of the cluster, while others are for the particular COE. In a typical workflow, a user would create a ClusterTemplate, then create one or more clusters using the ClusterTemplate. A cloud provider can also define a number of ClusterTemplates and provide them to the users. A ClusterTemplate cannot be updated or deleted if a cluster using this ClusterTemplate still exists.

The definition and usage of the parameters of a ClusterTemplate are as follows. They are loosely grouped as: mandatory, infrastructure, COE specific.

<name>

Name of the ClusterTemplate to create. The name does not have to be unique. If multiple ClusterTemplates have the same name, you will need to use the UUID to select the ClusterTemplate when creating a cluster or updating, deleting a ClusterTemplate. If a name is not specified, a random name will be generated using a string and a number, for example “pi-13-model”.

–coe <coe>

Specify the Container Orchestration Engine to use. Supported COE’s include ‘kubernetes’, ‘swarm’, ‘mesos’. If your environment has additional cluster drivers installed, refer to the cluster driver documentation for the new COE names. This is a mandatory parameter and there is no default value.

–image <image>

The name or UUID of the base image in Glance to boot the servers for the cluster. The image must have the attribute ‘os_distro’ defined as appropriate for the cluster driver. For the currently supported images, the os_distro names are:

C	os-distro
Kubernetes	Fedora-atomic, CoreOS
Swarm	Fedora-atomic

C	os-distro
Mesos	Ubuntu

This is a mandatory parameter and there is no default value.

–keypair <keypair>

The name of the SSH keypair to configure in the cluster servers for ssh access. You will need the key to be able to ssh to the servers in the cluster. The login name is specific to the cluster driver. If keypair is not provided in template it will be required at Cluster create. This value will be overridden by any keypair value that is provided during Cluster create.

–external-network <external-network>

The name or network ID of a Neutron network to provide connectivity to the external internet for the cluster. This network must be an external network, i.e. its attribute 'router:external' must be 'True'. The servers in the cluster will be connected to a private network and Magnum will create a router between this private network and the external network. This will allow the servers to download images, access discovery service, etc, and the containers to install packages, etc. In the opposite direction, floating IP's will be allocated from the external network to provide access from the external internet to servers and the container services hosted in the cluster. This is a mandatory parameter and there is no default value.

`--public`

Access to a ClusterTemplate is normally limited to the admin, owner or users within the same tenant as the owners. Setting this flag makes the ClusterTemplate public and accessible by other users. The default is not public.

–server-type <server-type>

The servers in the cluster can be VM or baremetal. This parameter selects the type of server to create for the cluster. The default is 'vm'. Possible values are 'vm', 'bm'.

–network-driver <network-driver>

The name of a network driver for providing the networks for the containers. Note that this is different and separate from the Neutron network for the cluster. The operation and networking model are specific to the particular driver. Supported network drivers and the default driver are:

COE	Network Driver	Default Driver
Kubernetes	Flannel	Flannel
Swarm	Docker, Flannel	Flannel
Mesos	Docker	Docker

–volume-driver <volume-driver>

The name of a volume driver for managing the persistent storage for the containers. The functionality supported are specific to the driver. Supported volume drivers and the default driver are:

COE	Volume Driver	Default Driver
Kubernetes	Cinder	No Driver
Swarm	Rexray	No Driver

C	V u ve	De- fau	Dri-
Me	Ro	No	ay Dri- ver

–dns-nameserver <dns-nameserver>

The DNS nameserver for the servers and containers in the cluster to use. This is configured in the private Neutron network for the cluster. The default is ‘8.8.8.8’.

–flavor <flavor>

The nova flavor id for booting the node servers. The default is ‘m1.small’.

–master-flavor <master-flavor>

The nova flavor id for booting the master or manager servers. The default is ‘m1.small’.

–http-proxy <http-proxy>

The IP address for a proxy to use when direct http access from the servers to sites on the external internet is blocked. This may happen in certain countries or enterprises, and the proxy allows the servers and containers to access these sites. The format is a URL including a port number. The default is ‘None’.

–https-proxy <https-proxy>

The IP address for a proxy to use when direct https access from the servers to sites on the external internet is blocked. This may happen in certain countries or enterprises, and the proxy allows the servers and containers to access these sites. The format is a URL including a port number. The default is ‘None’.

–no-proxy <no-proxy>

When a proxy server is used, some sites should not go through the proxy and should be accessed normally. In this case, you can specify these sites as a comma separated list of IP’s. The default is ‘None’.

–docker-volume-size <docker-volume-size>

If specified, container images will be stored in a cinder volume of the specified size in GB. Each cluster node will have a volume attached of the above size. If not specified, images will be stored in the compute instance's local disk. For the 'devicemapper' storage driver, the minimum value is 3GB. For the 'overlay' storage driver, the minimum value is 1GB. This value can be overridden at cluster creation.

–docker-storage-driver <docker-storage-driver>

The name of a driver to manage the storage for the images and the container's writable layer. The supported drivers are 'devicemapper' and 'overlay'. The default is 'devicemapper'.

–labels <KEY1=VALUE1,KEY2=VALUE2;KEY3=VALUE3...>

Arbitrary labels in the form of key=value pairs. The accepted keys and valid values are defined in the cluster drivers. They are used as a way to pass additional parameters that are specific to a cluster driver. Refer to the subsection on labels for a list of the supported key/value pairs and their usage.

`--tls-disabled`

Transport Layer Security (TLS) is normally enabled to secure the cluster. In some cases, users may want to disable TLS in the cluster, for instance during development or to troubleshoot certain problems. Specifying this parameter will disable TLS so that users can access the COE endpoints without a certificate. The default is TLS enabled.

`--registry-enabled`

Docker images by default are pulled from the public Docker registry, but in some cases, users may want to use a private registry. This option provides an alternative registry based on the Registry V2: Magnum will create a local registry in the cluster backed by swift to host the images. Refer to Docker Registry 2.0 (https://github.com/docker/distribution) for more details. The default is to use the public registry.

`--master-lb-enabled`

Since multiple masters may exist in a cluster, a load balancer is created to provide the API endpoint for the cluster and to direct requests to the masters. In some cases, such as when the LBaaS service is not available, this option can be set to 'false' to create a cluster without the load balancer. In this case, one of the masters will serve as the API endpoint. The default is 'true', i.e. to create the load balancer for the cluster.

5.4.1 Labels

Labels is a general method to specify supplemental parameters that are specific to certain COE or associated with certain options. Their format is key/value pair and their meaning is interpreted by the drivers that uses them. The drivers do validate the key/value pairs. Their usage is explained in details in the appropriate sections, however, since there are many possible labels, the following table provides a summary to help give a clearer picture. The label keys in the table are linked to more details elsewhere in the user guide.

label key	label value	default
`flannel-network-cidr`	IPv4 CIDR	10.100.0.0/16
`flannel-backend`		• udp • vxlan • host-gw
`flannel-network-subnetlen`	size of subnet to assign to node	24
`rexray-preempt`		• true • false

label key	label value	default
`mesos-slave-isolation`		• filesystem/posix • filesystem/linux • filesystem/shared • posix/cpu • posix/mem • posix/disk • cgroups/cpu • cgroups/mem • docker/runtime • namespaces/pid

label key	label value	default
`mesos-slave-image-providers`		• "" (empty string) • appc • docker • appc,docker
`mesos-slave-work-dir`	(directory name)	""
`mesos-slave-executor-env-variables`	(file name)	""
`swarm-strategy`		• spread spread • binpack • random

label key	label value	default
`admission-control-list`	see below	see below
`prometheus-monitoring`		• true • false
`grafana-admin-passwd`	(any string)	"admin"
`kube-tag`	see below	see below
`kube-dashboard-enabled`		• true • false
`"docker-volume-type"`	see below	see below
`etcd-volume-size`	etcd storage	0

label key	label value	default
	volume size	

5.5 Cluster

A cluster (previously known as bay) is an instance of the ClusterTemplate of a COE. Magnum deploys a cluster by referring to the attributes defined in the particular ClusterTemplate as well as a few additional parameters for the cluster. Magnum deploys the orchestration templates provided by the cluster driver to create and configure all the necessary infrastructure. When ready, the cluster is a fully operational COE that can host containers.

5.5.1 Infrastructure

The infrastructure of the cluster consists of the resources provided by the various OpenStack services. Existing infrastructure, including infrastructure external to OpenStack, can also be used by the cluster, such as DNS, public network, public discovery service, Docker registry. The actual resources created depends on the COE type and the options specified; therefore you need to refer to the cluster driver documentation of the COE for specific details. For instance, the option '–master-lb-enabled' in the ClusterTemplate will cause a load balancer pool along with the health monitor and floating IP to be created. It is important to distinguish resources in the IaaS level from resources in the PaaS level. For instance, the infrastructure networking in OpenStack IaaS is different and separate from the container networking in Kubernetes or Swarm PaaS.

Typical infrastructure includes the following.

Servers

The servers host the containers in the cluster and these servers can be VM or bare metal. VM's are provided by Nova. Since multiple VM's are hosted on a physical server, the VM's provide the isolation needed for containers between different tenants running on the same physical server. Bare metal servers are provided by Ironic and are used when peak performance with virtually no overhead is needed for the containers.

Identity

Keystone provides the authentication and authorization for managing the cluster infrastructure.

Network

Networking among the servers is provided by Neutron. Since COE currently are not multi-tenant, isolation for multi-tenancy on the networking level is done by using a private network for each cluster. As a result, containers belonging to one tenant will not be accessible to containers or servers of another tenant. Other networking resources may also be used, such as load balancer and routers. Networking among containers can be provided by Kuryr if needed.

Storage

Cinder provides the block storage that can be used to host the containers and as persistent storage for the containers.

Security

Barbican provides the storage of secrets such as certificates used for Transport Layer Security (TLS) within the cluster.

5.5.2 Life cycle

The set of life cycle operations on the cluster is one of the key value that Magnum provides, enabling clusters to be managed painlessly on OpenStack. The current operations are the basic CRUD operations, but more advanced operations are under discussion in the community and will be implemented as needed.

NOTE The OpenStack resources created for a cluster are fully accessible to the cluster owner. Care should be taken when modifying or reusing these resources to avoid impacting Magnum operations in unexpected manners. For instance, if you launch your own Nova instance on the cluster private network, Magnum would not be aware of this instance. Therefore, the cluster-delete operation will fail because Magnum would not delete the extra Nova instance and the private Neutron network cannot be removed while a Nova instance is still attached.

NOTE Currently Heat nested templates are used to create the resources; therefore if an error occurs, you can troubleshoot through Heat. For more help on Heat stack troubleshooting, refer to the Troubleshooting Guide (https://github.com/openstack/magnum/blob/master/doc/source/troubleshooting-guide.rst#heat-stacks)↗.

5.5.2.1 Create

NOTE bay-<command> are the deprecated versions of these commands and are still support in current release. They will be removed in a future version. Any references to the term bay will be replaced in the parameters when using the 'bay' versions of the commands. For example, in 'bay-create' –baymodel is used as the baymodel parameter for this command instead of –cluster-template.

The 'cluster-create' command deploys a cluster, for example:

```
magnum cluster-create mycluster \
                  --cluster-template mytemplate \
                  --node-count 8 \
                  --master-count 3
```

The 'cluster-create' operation is asynchronous; therefore you can initiate another 'cluster-create' operation while the current cluster is being created. If the cluster fails to be created, the infrastructure created so far may be retained or deleted depending on the particular orchestration engine. As a common practice, a failed cluster is retained during development for troubleshooting, but they are automatically deleted in production. The current cluster drivers use Heat templates and the resources of a failed 'cluster-create' are retained.

The definition and usage of the parameters for 'cluster-create' are as follows:

<name>

Name of the cluster to create. If a name is not specified, a random name will be generated using a string and a number, for example "gamma-7-cluster".

–cluster-template <cluster-template>

The ID or name of the ClusterTemplate to use. This is a mandatory parameter. Once a ClusterTemplate is used to create a cluster, it cannot be deleted or modified until all clusters that use the ClusterTemplate have been deleted.

–keypair <keypair>

The name of the SSH keypair to configure in the cluster servers for ssh access. You will need the key to be able to ssh to the servers in the cluster. The login name is specific to the cluster driver. If keypair is not provided it will attempt to use the value in the ClusterTemplate. If the ClusterTemplate is also missing a keypair value then an error will be returned. The keypair value provided here will override the keypair value from the ClusterTemplate.

–node-count <node-count>

The number of servers that will serve as node in the cluster. The default is 1.

–master-count <master-count>

The number of servers that will serve as master for the cluster. The default is 1. Set to more than 1 master to enable High Availability. If the option '–master-lb-enabled' is specified in the ClusterTemplate, the master servers will be placed in a load balancer pool.

–discovery-url <discovery-url>

The custom discovery url for node discovery. This is used by the COE to discover the servers that have been created to host the containers. The actual discovery mechanism varies with the COE. In some cases, Magnum fills in the server info in the discovery service. In other cases, if the discovery-url is not specified, Magnum will use the public discovery service at:

```
https://discovery.etcd.io
```

In this case, Magnum will generate a unique url here for each cluster and store the info for the servers.

–timeout <timeout>

The timeout for cluster creation in minutes. The value expected is a positive integer and the default is 60 minutes. If the timeout is reached during cluster-create, the operation will be aborted and the cluster status will be set to 'CREATE_FAILED'.

5.5.2.2 List

The 'cluster-list' command lists all the clusters that belong to the tenant, for example:

```
magnum cluster-list
```

5.5.2.3 Show

The ‘cluster-show’ command prints all the details of a cluster, for example:

```
magnum cluster-show mycluster
```

The properties include those not specified by users that have been assigned default values and properties from new resources that have been created for the cluster.

5.5.2.4 Update

A cluster can be modified using the ‘cluster-update’ command, for example:

```
magnum cluster-update mycluster replace node_count=8
```

The parameters are positional and their definition and usage are as follows.

<cluster>
: This is the first parameter, specifying the UUID or name of the cluster to update.

<op>
: This is the second parameter, specifying the desired change to be made to the cluster attributes. The allowed changes are ‘add’, ‘replace’ and ‘remove’.

<attribute=value>
: This is the third parameter, specifying the targeted attributes in the cluster as a list separated by blank space. To add or replace an attribute, you need to specify the value for the attribute. To remove an attribute, you only need to specify the name of the attribute. Currently the only attribute that can be replaced or removed is ‘node_count’. The attributes ‘name’, ‘master_count’ and ‘discovery_url’ cannot be replaced or delete. The table below summarizes the possible change to a cluster.

Attribute	replace	remove
node_count	add/replace node_count to remove nodes	set node_count to default

Attribute	replace	remove
		fault of 1
master_count	no	no
name	no	no
discovery_url	no	no

The ‘cluster-update’ operation cannot be initiated when another operation is in progress.

NOTE: The attribute names in cluster-update are slightly different from the corresponding names in the cluster-create command: the dash ‘-‘ is replaced by an underscore ‘_’. For instance, ‘node-count’ in cluster-create is ‘node_count’ in cluster-update.

5.5.2.5 Scale

Scaling a cluster means adding servers to or removing servers from the cluster. Currently, this is done through the ‘cluster-update’ operation by modifying the node-count attribute, for example:

```
magnum cluster-update mycluster replace node_count=2
```

When some nodes are removed, Magnum will attempt to find nodes with no containers to remove. If some nodes with containers must be removed, Magnum will log a warning message.

5.5.2.6 Delete

The ‘cluster-delete’ operation removes the cluster by deleting all resources such as servers, network, storage; for example:

```
magnum cluster-delete mycluster
```

The only parameter for the cluster-delete command is the ID or name of the cluster to delete. Multiple clusters can be specified, separated by a blank space.

If the operation fails, there may be some remaining resources that have not been deleted yet. In this case, you can troubleshoot through Heat. If the templates are deleted manually in Heat, you can delete the cluster in Magnum to clean up the cluster from Magnum database.

The 'cluster-delete' operation can be initiated when another operation is still in progress.

5.6 Python Client

5.6.1 Installation

Follow the instructions in the OpenStack Installation Guide to enable the repositories for your distribution:

- RHEL/CentOS/Fedora (http://docs.openstack.org/liberty/install-guide-rdo/)
- Ubuntu/Debian (http://docs.openstack.org/liberty/install-guide-ubuntu/)
- openSUSE/SUSE Linux Enterprise (http://docs.openstack.org/liberty/install-guide-obs/)

Install using distribution packages for RHEL/CentOS/Fedora:

```
$ sudo yum install python-magnumclient
```

Install using distribution packages for Ubuntu/Debian:

```
$ sudo apt-get install python-magnumclient
```

Install using distribution packages for OpenSuSE and SuSE Enterprise Linux:

```
$ sudo zypper install python-magnumclient
```

5.6.2 Verifying installation

Execute the `magnum` command with the `–version` argument to confirm that the client is installed and in the system path:

```
$ magnum --version
1.1.0
```

Note that the version returned may differ from the above, 1.1.0 was the latest available version at the time of writing.

5.6.3 Using the command-line client

Refer to the OpenStack Command-Line Interface Reference (http://docs.openstack.org/cli-reference/magnum.html) for a full list of the commands supported by the `magnum` command-line client.

5.7 Horizon Interface

Magnum provides a Horizon plugin so that users can access the Container Infrastructure Management service through the OpenStack browser-based graphical UI. The plugin is available from magnum-ui (https://github.com/openstack/magnum-ui). It is not installed by default in the standard Horizon service, but you can follow the instruction for installing a Horizon plugin (http://docs.openstack.org/developer/horizon/tutorials/plugin.html#installing-your-plugin).

In Horizon, the container infrastructure panel is part of the 'Project' view and it currently supports the following operations:

- View list of cluster templates
- View details of a cluster template
- Create a cluster template
- Delete a cluster template
- View list of clusters
- View details of a cluster
- Create a cluster
- Delete a cluster
- Get the Certificate Authority for a cluster
- Sign a user key and obtain a signed certificate for accessing the secured COE API endpoint in a cluster.

Other operations are not yet supported and the CLI should be used for these.

5.8 Cluster Drivers

A cluster driver is a collection of python code, heat templates, scripts, images, and documents for a particular COE on a particular distro. Magnum presents the concept of ClusterTemplates and clusters. The implementation for a particular cluster type is provided by the cluster driver. In other words, the cluster driver provisions and manages the infrastructure for the COE. Magnum includes default drivers for the following COE and distro pairs:

COE	distro
Kubernetes	Fedora Atomic
Kubernetes	CoreOS
Swarm	Fedora Atomic
Mesos	Ubuntu

Magnum is designed to accommodate new cluster drivers to support custom COE's and this section describes how a new cluster driver can be constructed and enabled in Magnum.

5.8.1 Directory structure

Magnum expects the components to be organized in the following directory structure under the directory ‘drivers’:

```
COE_Distro/
   image/
   templates/
   api.py
   driver.py
   monitor.py
   scale.py
   template_def.py
   version.py
```

The minimum required components are:

driver.py

Python code that implements the controller operations for the particular COE. The driver must implement: Currently supported: `cluster_create`, `cluster_update`, `cluster_delete`.

templates

A directory of orchestration templates for managing the lifecycle of clusters, including creation, configuration, update, and deletion. Currently only Heat templates are supported, but in the future other orchestration mechanism such as Ansible may be supported.

template_def.py

Python code that maps the parameters from the ClusterTemplate to the input parameters for the orchestration and invokes the orchestration in the templates directory.

version.py

Tracks the latest version of the driver in this directory. This is defined by a `version` attribute and is represented in the form of `1.0.0`. It should also include a `Driver` attribute with descriptive name such as `fedora_swarm_atomic`.

The remaining components are optional:

image

Instructions for obtaining or building an image suitable for the COE.

api.py

Python code to interface with the COE.

monitor.py

Python code to monitor the resource utilization of the cluster.

scale.py

Python code to scale the cluster by adding or removing nodes.

5.8.2 Sample cluster driver

To help developers in creating new COE drivers, a minimal cluster driver is provided as an example. The 'docker' cluster driver will simply deploy a single VM running Ubuntu with the latest Docker version installed. It is not a true cluster, but the simplicity will help to illustrate the key concepts.

To be filled in

5.8.3 Installing a cluster driver

To be filled in

5.9 Cluster Type Definition

There are three key pieces to a Cluster Type Definition:

1. Heat Stack template - The HOT file that Magnum will use to generate a cluster using a Heat Stack.
2. Template definition - Magnum's interface for interacting with the Heat template.
3. Definition Entry Point - Used to advertise the available Cluster Types.

5.9.1 The Heat Stack Template

The Heat Stack Template is where most of the real work happens. The result of the Heat Stack Template should be a full Container Orchestration Environment.

5.9.2 The Template Definition

Template definitions are a mapping of Magnum object attributes and Heat template parameters, along with Magnum consumable template outputs. A Cluster Type Definition indicates which Cluster Types it can provide. Cluster Types are how Magnum determines which of the enabled Cluster Type Definitions it will use for a given cluster.

5.9.3 The Definition Entry Point

Entry points are a standard discovery and import mechanism for Python objects. Each Template Definition should have an Entry Point in the `magnum.template_definitions` group. This example exposes it's Template Definition as `example_template = example_template:ExampleTemplate` in the `magnum.template_definitions` group.

5.9.4 Installing Cluster Templates

Because Cluster Type Definitions are basically Python projects, they can be worked with like any other Python project. They can be cloned from version control and installed or uploaded to a package index and installed via utilities such as pip.

Enabling a Cluster Type is as simple as adding it's Entry Point to the `enabled_definitions` config option in magnum.conf.:

```
# Setup python environment and install Magnum

$ virtualenv .venv
$ source .venv/bin/active
(.venv)$ git clone https://github.com/openstack/magnum.git
(.venv)$ cd magnum
(.venv)$ python setup.py install

# List installed templates, notice default templates are enabled

(.venv)$ magnum-template-manage list-templates
Enabled Templates
  magnum_vm_atomic_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster.yaml
  magnum_vm_coreos_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster-coreos.yaml
Disabled Templates

# Install example template
```

```
(.venv)$ cd contrib/templates/example
(.venv)$ python setup.py install

# List installed templates, notice example template is disabled

(.venv)$ magnum-template-manage list-templates
Enabled Templates
  magnum_vm_atomic_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster.yaml
  magnum_vm_coreos_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster-coreos.yaml
Disabled Templates
  example_template: /home/example/.venv/local/lib/python2.7/site-packages/
ExampleTemplate-0.1-py2.7.egg/example_template/example.yaml

# Enable example template by setting enabled_definitions in magnum.conf

(.venv)$ sudo mkdir /etc/magnum
(.venv)$ sudo bash -c "cat > /etc/magnum/magnum.conf << END_CONF
[bay]
enabled_definitions=magnum_vm_atomic_k8s,magnum_vm_coreos_k8s,example_template
END_CONF"

# List installed templates, notice example template is now enabled

(.venv)$ magnum-template-manage list-templates
Enabled Templates
  example_template: /home/example/.venv/local/lib/python2.7/site-packages/
ExampleTemplate-0.1-py2.7.egg/example_template/example.yaml
  magnum_vm_atomic_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster.yaml
  magnum_vm_coreos_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster-coreos.yaml
Disabled Templates

# Use --details argument to get more details about each template

(.venv)$ magnum-template-manage list-templates --details
Enabled Templates
  example_template: /home/example/.venv/local/lib/python2.7/site-packages/
ExampleTemplate-0.1-py2.7.egg/example_template/example.yaml
     Server_Type  OS       CoE
     vm         example  example_coe
  magnum_vm_atomic_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster.yaml
     Server_Type   OS             CoE
```

```
     vm        fedora-atomic  kubernetes
  magnum_vm_coreos_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster-coreos.yaml
     Server_Type  OS      CoE
     vm         coreos  kubernetes
Disabled Templates
```

5.10 Heat Stack Templates

Heat Stack Templates are what Magnum passes to Heat to generate a cluster. For each ClusterTemplate resource in Magnum, a Heat stack is created to arrange all of the cloud resources needed to support the container orchestration environment. These Heat stack templates provide a mapping of Magnum object attributes to Heat template parameters, along with Magnum consumable stack outputs. Magnum passes the Heat Stack Template to the Heat service to create a Heat stack. The result is a full Container Orchestration Environment.

5.11 Choosing a COE

Magnum supports a variety of COE options, and allows more to be added over time as they gain popularity. As an operator, you may choose to support the full variety of options, or you may want to offer a subset of the available choices. Given multiple choices, your users can run one or more clusters, and each may use a different COE. For example, I might have multiple clusters that use Kubernetes, and just one cluster that uses Swarm. All of these clusters can run concurrently, even though they use different COE software.

Choosing which COE to use depends on what tools you want to use to manage your containers once you start your app. If you want to use the Docker tools, you may want to use the Swarm cluster type. Swarm will spread your containers across the various nodes in your cluster automatically. It does not monitor the health of your containers, so it can't restart them for you if they stop. It will not automatically scale your app for you (as of Swarm version 1.2.2). You may view this as a plus. If you prefer to manage your application yourself, you might prefer swarm over the other COE options.

Kubernetes (as of v1.2) is more sophisticated than Swarm (as of v1.2.2). It offers an attractive YAML file description of a pod, which is a grouping of containers that run together as part of a distributed application. This file format allows you to model your application deployment using

a declarative style. It has support for auto scaling and fault recovery, as well as features that allow for sophisticated software deployments, including canary deploys and blue/green deploys. Kubernetes is very popular, especially for web applications.

Apache Mesos is a COE that has been around longer than Kubernetes or Swarm. It allows for a variety of different frameworks to be used along with it, including Marathon, Aurora, Chronos, Hadoop, and a number of others. (http://mesos.apache.org/documentation/latest/frameworks/)

The Apache Mesos framework design can be used to run alternate COE software directly on Mesos. Although this approach is not widely used yet, it may soon be possible to run Mesos with Kubernetes and Swarm as frameworks, allowing you to share the resources of a cluster between multiple different COEs. Until this option matures, we encourage Magnum users to create multiple clusters, and use the COE in each cluster that best fits the anticipated workload.

Finding the right COE for your workload is up to you, but Magnum offers you a choice to select among the prevailing leading options. Once you decide, see the next sections for examples of how to create a cluster with your desired COE.

5.12 Native Clients

Magnum preserves the native user experience with a COE and does not provide a separate API or client. This means you will need to use the native client for the particular cluster type to interface with the clusters. In the typical case, there are two clients to consider:

COE level

This is the orchestration or management level such as Kubernetes, Swarm, Mesos and its frameworks.

Container level

This is the low level container operation. Currently it is Docker for all clusters.

The clients can be CLI and/or browser-based. You will need to refer to the documentation for the specific native client and appropriate version for details, but following are some pointers for reference.

Kubernetes CLI is the tool ‘kubectl’, which can be simply copied from a node in the cluster or downloaded from the Kubernetes release. For instance, if the cluster is running Kubernetes release 1.2.0, the binary for ‘kubectl’ can be downloaded as and set up locally as follows:

```
curl -O https://storage.googleapis.com/kubernetes-release/release/v1.2.0/bin/linux/amd64/
kubectl
```

```
chmod +x kubectl
sudo mv kubectl /usr/local/bin/kubectl
```

Kubernetes also provides a browser UI. If the cluster has the Kubernetes Dashboard running; it can be accessed using:

```
eval $(magnum cluster-config <cluster-name>)
kubectl proxy

The browser can be accessed at http://localhost:8001/ui
```

For Swarm, the main CLI is 'docker', along with associated tools such as 'docker-compose', etc. Specific version of the binaries can be obtained from the Docker Engine installation (https://docs.docker.com/engine/installation/binaries/).

Mesos cluster uses the Marathon framework.

Depending on the client requirement, you may need to use a version of the client that matches the version in the cluster. To determine the version of the COE and container, use the command 'cluster-show' and look for the attribute *coe_version* and *container_version*:

```
magnum cluster-show k8s-cluster
+--------------------+------------------------------------------------------------+
| Property           | Value                                                      |
+--------------------+------------------------------------------------------------+
| status             | CREATE_COMPLETE                                            |
| uuid               | 04952c60-a338-437f-a7e7-d016d1d00e65                       |
| stack_id           | b7bf72ce-b08e-4768-8201-e63a99346898                       |
| status_reason      | Stack CREATE completed successfully                        |
| created_at         | 2016-07-25T23:14:06+00:00                                  |
| updated_at         | 2016-07-25T23:14:10+00:00                                  |
| create_timeout     | 60                                                         |
| coe_version        | v1.2.0                                                     |
| api_address        | https://192.168.19.86:6443                                 |
| cluster_template_id| da2825a0-6d09-4208-b39e-b2db666f1118                       |
| master_addresses   | ['192.168.19.87']                                          |
| node_count         | 1                                                          |
| node_addresses     | ['192.168.19.88']                                          |
| master_count       | 1                                                          |
| container_version  | 1.9.1                                                      |
| discovery_url      | https://discovery.etcd.io/3b7fb09733429d16679484673ba3bfd5 |
| name               | k8s-cluster                                                |
+--------------------+------------------------------------------------------------+
```

5.13 Kubernetes

Kubernetes uses a range of terminology that we refer to in this guide. We define these common terms for your reference:

Pod

When using the Kubernetes container orchestration engine, a pod is the smallest deployable unit that can be created and managed. A pod is a co-located group of application containers that run with a shared context. When using Magnum, pods are created and managed within clusters. Refer to the pods section (http://kubernetes.io/v1.0/docs/user-guide/pods.html) in the Kubernetes User Guide (http://kubernetes.io/v1.0/docs/user-guide/) for more information.

Replication controller

A replication controller is used to ensure that at any given time a certain number of replicas of a pod are running. Pods are automatically created and deleted by the replication controller as necessary based on a template to ensure that the defined number of replicas exist. Refer to the replication controller section (http://kubernetes.io/v1.0/docs/user-guide/replication-controller.html) in the Kubernetes User Guide (http://kubernetes.io/v1.0/docs/user-guide/) for more information.

Service

A service is an additional layer of abstraction provided by the Kubernetes container orchestration engine which defines a logical set of pods and a policy for accessing them. This is useful because pods are created and deleted by a replication controller, for example, other pods needing to discover them can do so via the service abstraction. Refer to the services section (http://kubernetes.io/v1.0/docs/user-guide/services.html) in the Kubernetes User Guide (http://kubernetes.io/v1.0/docs/user-guide/) for more information.

When Magnum deploys a Kubernetes cluster, it uses parameters defined in the ClusterTemplate and specified on the cluster-create command, for example:

```
magnum cluster-template-create k8s-cluster-template \
                           --image fedora-atomic-latest \
                           --keypair testkey \
                           --external-network public \
                           --dns-nameserver 8.8.8.8 \
                           --flavor m1.small \
                           --docker-volume-size 5 \
                           --network-driver flannel \
                           --coe kubernetes
```

```
magnum cluster-create k8s-cluster \
                      --cluster-template k8s-cluster-template \
                      --master-count 3 \
                      --node-count 8
```

Following are further details relevant to a Kubernetes cluster:

Number of masters (master-count)

Specified in the cluster-create command to indicate how many servers will run as master in the cluster. Having more than one will provide high availability. The masters will be in a load balancer pool and the virtual IP address (VIP) of the load balancer will serve as the Kubernetes API endpoint. For external access, a floating IP associated with this VIP is available and this is the endpoint shown for Kubernetes in the 'cluster-show' command.

Number of nodes (node-count)

Specified in the cluster-create command to indicate how many servers will run as node in the cluster to host the users' pods. The nodes are registered in Kubernetes using the Nova instance name.

Network driver (network-driver)

Specified in the ClusterTemplate to select the network driver. The supported and default network driver is 'flannel', an overlay network providing a flat network for all pods.

Volume driver (volume-driver)

Specified in the ClusterTemplate to select the volume driver. The supported volume driver is 'cinder', allowing Cinder volumes to be mounted in containers for use as persistent storage. Data written to these volumes will persist after the container exits and can be accessed again from other containers, while data written to the union file system hosting the container will be deleted.

Storage driver (docker-storage-driver)

Specified in the ClusterTemplate to select the Docker storage driver. The supported storage drivers are 'devicemapper' and 'overlay', with 'devicemapper' being the default.

Image (image)

Specified in the ClusterTemplate to indicate the image to boot the servers. The image binary is loaded in Glance with the attribute 'os_distro = fedora-atomic'. Current supported images are Fedora Atomic (download from Fedora (https://alt.fedoraproject.org/pub/alt/atomic/stable/Cloud-Images/x86_64/Images)) and CoreOS (download from CoreOS (http://beta.release.core-os.net/amd64-usr/current/coreos_production_openstack_image.img.bz2))

TLS (tls-disabled)

Transport Layer Security is enabled by default, so you need a key and signed certificate to access the Kubernetes API and CLI. Magnum handles its own key and certificate when interfacing with the Kubernetes cluster. In development mode, TLS can be disabled. Refer to the 'Transport Layer Security'_ section for more details.

What runs on the servers

The servers for Kubernetes master host containers in the 'kube-system' name space to run the Kubernetes proxy, scheduler and controller manager. The masters will not host users' pods. Kubernetes API server, docker daemon, etcd and flannel run as systemd services. The servers for Kubernetes node also host a container in the 'kube-system' name space to run the Kubernetes proxy, while Kubernetes kubelet, docker daemon and flannel run as systemd services.

Log into the servers

You can log into the master servers using the login 'fedora' and the keypair specified in the ClusterTemplate.

In addition to the common attributes in the ClusterTemplate, you can specify the following attributes that are specific to Kubernetes by using the labels attribute.

This label corresponds to Kubernetes parameter for the API server '–admission-control'. For more details, refer to the Admission Controllers (https://kubernetes.io/docs/admin/admission-controllers//). The default value corresponds to the one recommended in this doc for our current Kubernetes version.

This label sets the size of a volume holding the etcd storage data. The default value is 0, meaning the etcd data is not persisted (no volume).

This label allows users to select a specific Kubernetes release, based on its container tag (https://hub.docker.com/r/openstackmagnum/kubernetes-apiserver/tags/). If unset, the current Magnum version's default Kubernetes release is installed.

This label triggers the deployment of the kubernetes dashboard. The default value is 1, meaning it will be enabled.

5.13.1 External load balancer for services

All Kubernetes pods and services created in the cluster are assigned IP addresses on a private container network so they can access each other and the external internet. However, these IP addresses are not accessible from an external network.

To publish a service endpoint externally so that the service can be accessed from the external network, Kubernetes provides the external load balancer feature. This is done by simply specifying in the service manifest the attribute “type: LoadBalancer”. Magnum enables and configures the Kubernetes plugin for OpenStack so that it can interface with Neutron and manage the necessary networking resources.

When the service is created, Kubernetes will add an external load balancer in front of the service so that the service will have an external IP address in addition to the internal IP address on the container network. The service endpoint can then be accessed with this external IP address. Kubernetes handles all the life cycle operations when pods are modified behind the service and when the service is deleted.

Refer to the document Kubernetes external load balancer (https://github.com/openstack/magnum/blob/master/doc/source/dev/kubernetes-load-balancer.rst) for more details.

5.14 Swarm

A Swarm cluster is a pool of servers running Docker daemon that is managed as a single Docker host. One or more Swarm managers accepts the standard Docker API and manage this pool of servers. Magnum deploys a Swarm cluster using parameters defined in the ClusterTemplate and specified on the ‘cluster-create’ command, for example:

```
magnum cluster-template-create swarm-cluster-template \
                           --image fedora-atomic-latest \
                           --keypair testkey \
                           --external-network public \
                           --dns-nameserver 8.8.8.8 \
                           --flavor m1.small \
                           --docker-volume-size 5 \
                           --coe swarm

magnum cluster-create swarm-cluster \
                   --cluster-template swarm-cluster-template \
                   --master-count 3 \
                   --node-count 8
```

Following are further details relevant to Swarm:

What runs on the servers

There are two types of servers in the Swarm cluster: managers and nodes. The Docker daemon runs on all servers. On the servers for manager, the Swarm manager is run as a Docker container on port 2376 and this is initiated by the systemd service swarm-manager. Etcd is also run on the manager servers for discovery of the node servers in the cluster. On the servers for node, the Swarm agent is run as a Docker container on port 2375 and this is initiated by the systemd service swarm-agent. On start up, the agents will register themselves in etcd and the managers will discover the new node to manage.

Number of managers (master-count)

Specified in the cluster-create command to indicate how many servers will run as managers in the cluster. Having more than one will provide high availability. The managers will be in a load balancer pool and the load balancer virtual IP address (VIP) will serve as the Swarm API endpoint. A floating IP associated with the load balancer VIP will serve as the external Swarm API endpoint. The managers accept the standard Docker API and perform the corresponding operation on the servers in the pool. For instance, when a new container is created, the managers will select one of the servers based on some strategy and schedule the containers there.

Number of nodes (node-count)

Specified in the cluster-create command to indicate how many servers will run as nodes in the cluster to host your Docker containers. These servers will register themselves in etcd for discovery by the managers, and interact with the managers. Docker daemon is run locally to host containers from users.

Network driver (network-driver)

Specified in the ClusterTemplate to select the network driver. The supported drivers are ‘docker’ and ‘flannel’, with ‘docker’ as the default. With the ‘docker’ driver, containers are connected to the ‘docker0’ bridge on each node and are assigned local IP address. With the ‘flannel’ driver, containers are connected to a flat overlay network and are assigned IP address by Flannel.

Volume driver (volume-driver)

Specified in the ClusterTemplate to select the volume driver to provide persistent storage for containers. The supported volume driver is 'rexray'. The default is no volume driver. When 'rexray' or other volume driver is deployed, you can use the Docker 'volume' command to create, mount, unmount, delete volumes in containers. Cinder block storage is used as the backend to support this feature.

Storage driver (docker-storage-driver)

Specified in the ClusterTemplate to select the Docker storage driver. The supported storage driver are 'devicemapper' and 'overlay', with 'devicemapper' being the default.

Image (image)

Specified in the ClusterTemplate to indicate the image to boot the servers for the Swarm manager and node. The image binary is loaded in Glance with the attribute 'os_distro = fedora-atomic'. Current supported image is Fedora Atomic (download from Fedora (https://alt.fedoraproject.org/pub/alt/atomic/stable/Cloud-Images/x86_64/Images))

TLS (tls-disabled)

Transport Layer Security is enabled by default to secure the Swarm API for access by both the users and Magnum. You will need a key and a signed certificate to access the Swarm API and CLI. Magnum handles its own key and certificate when interfacing with the Swarm cluster. In development mode, TLS can be disabled. Refer to the 'Transport Layer Security'_ section for details on how to create your key and have Magnum sign your certificate.

Log into the servers

You can log into the manager and node servers with the account 'fedora' and the keypair specified in the ClusterTemplate.

In addition to the common attributes in the ClusterTemplate, you can specify the following attributes that are specific to Swarm by using the labels attribute.

This label corresponds to Swarm parameter for master '–strategy'. For more details, refer to the Swarm Strategy (https://docs.docker.com/swarm/scheduler/strategy/). Valid values for this label are:

- spread
- binpack
- random

5.15 Mesos

A Mesos cluster consists of a pool of servers running as Mesos slaves, managed by a set of servers running as Mesos masters. Mesos manages the resources from the slaves but does not itself deploy containers. Instead, one of more Mesos frameworks running on the Mesos cluster would accept user requests on their own endpoint, using their particular API. These frameworks would then negotiate the resources with Mesos and the containers are deployed on the servers where the resources are offered.

Magnum deploys a Mesos cluster using parameters defined in the ClusterTemplate and specified on the 'cluster-create' command, for example:

```
magnum cluster-template-create mesos-cluster-template \
                        --image ubuntu-mesos \
                        --keypair testkey \
                        --external-network public \
                        --dns-nameserver 8.8.8.8 \
                        --flavor m1.small \
                        --coe mesos

magnum cluster-create mesos-cluster \
                    --cluster-template mesos-cluster-template \
                    --master-count 3 \
                    --node-count 8
```

Following are further details relevant to Mesos:

What runs on the servers

There are two types of servers in the Mesos cluster: masters and slaves. The Docker daemon runs on all servers. On the servers for master, the Mesos master is run as a process on port 5050 and this is initiated by the upstart service 'mesos-master'. Zookeeper is also run on the master servers, initiated by the upstart service 'zookeeper'. Zookeeper is used by the master servers for electing the leader among the masters, and by the slave servers and frameworks to determine the current leader. The framework Marathon is run as a process on port 8080 on the master servers, initiated by the upstart service 'marathon'. On the servers for slave, the Mesos slave is run as a process initiated by the upstart service 'mesos-slave'.

Number of master (master-count)

Specified in the cluster-create command to indicate how many servers will run as masters in the cluster. Having more than one will provide high availability. If the load balancer option is specified, the masters will be in a load balancer pool and the load balancer virtual IP address (VIP) will serve as the Mesos API endpoint. A floating IP associated with the load balancer VIP will serve as the external Mesos API endpoint.

Number of agents (node-count)

Specified in the cluster-create command to indicate how many servers will run as Mesos slave in the cluster. Docker daemon is run locally to host containers from users. The slaves report their available resources to the master and accept request from the master to deploy tasks from the frameworks. In this case, the tasks will be to run Docker containers.

Network driver (network-driver)

Specified in the ClusterTemplate to select the network driver. Currently 'docker' is the only supported driver: containers are connected to the 'docker0' bridge on each node and are assigned local IP address.

Volume driver (volume-driver)

Specified in the ClusterTemplate to select the volume driver to provide persistent storage for containers. The supported volume driver is 'rexray'. The default is no volume driver. When 'rexray' or other volume driver is deployed, you can use the Docker 'volume' command to create, mount, unmount, delete volumes in containers. Cinder block storage is used as the backend to support this feature.

Storage driver (docker-storage-driver)

This is currently not supported for Mesos.

Image (image)

Specified in the ClusterTemplate to indicate the image to boot the servers for the Mesos master and slave. The image binary is loaded in Glance with the attribute 'os_distro = ubuntu'. You can download the ready-built image (https://fedorapeople.org/groups/magnum/ubuntu-mesos-latest.qcow2).

TLS (tls-disabled)

Transport Layer Security is currently not implemented yet for Mesos.

Log into the servers

You can log into the manager and node servers with the account 'ubuntu' and the keypair specified in the ClusterTemplate.

In addition to the common attributes in the baymodel, you can specify the following attributes that are specific to Mesos by using the labels attribute.

When the volume driver 'rexray' is used, you can mount a data volume backed by Cinder to a host to be accessed by a container. In this case, the label 'rexray_preempt' can optionally be set to True or False to enable any host to take control of the volume regardless of whether other hosts are using the volume. This will in effect unmount the volume from the current host and remount it on the new host. If this label is set to false, then rexray will ensure data safety for locking the volume before remounting. The default value is False.

This label corresponds to the Mesos parameter for slave '–isolation'. The isolators are needed to provide proper isolation according to the runtime configurations specified in the container image. For more details, refer to the Mesos configuration (http://mesos.apache.org/documentation/latest/configuration/) and the Mesos container image support (http://mesos.apache.org/documentation/latest/container-image/). Valid values for this label are:

- filesystem/posix
- filesystem/linux
- filesystem/shared
- posix/cpu
- posix/mem
- posix/disk
- cgroups/cpu
- cgroups/mem
- docker/runtime
- namespaces/pid

This label corresponds to the Mesos parameter for agent '–image_providers', which tells Mesos containerizer what types of container images are allowed. For more details, refer to the Mesos configuration (http://mesos.apache.org/documentation/latest/configuration/) and the Mesos container image support (http://mesos.apache.org/documentation/latest/container-image/). Valid values are:

- appc
- docker
- appc,docker

This label corresponds to the Mesos parameter '–work_dir' for slave. For more details, refer to the Mesos configuration (http://mesos.apache.org/documentation/latest/configuration/). Valid value is a directory path to use as the work directory for the framework, for example:

```
mesos_slave_work_dir=/tmp/mesos
```

This label corresponds to the Mesos parameter for slave '–executor_environment_variables', which passes additional environment variables to the executor and subsequent tasks. For more details, refer to the Mesos configuration (http://mesos.apache.org/documentation/latest/configuration/). Valid value is the name of a JSON file, for example:

```
mesos_slave_executor_env_variables=/home/ubuntu/test.json
```

The JSON file should contain environment variables, for example:

```
{
   "PATH": "/bin:/usr/bin",
   "LD_LIBRARY_PATH": "/usr/local/lib"
}
```

By default the executor will inherit the slave's environment variables.

5.15.1 Building Mesos image

The boot image for Mesos cluster is an Ubuntu 14.04 base image with the following middleware pre-installed:

- `docker`
- `zookeeper`

- `mesos`
- `marathon`

The cluster driver provides two ways to create this image, as follows.

5.15.1.1 Diskimage-builder

To run the diskimage-builder (http://docs.openstack.org/developer/diskimage-builder) tool manually, use the provided elements (http://git.openstack.org/cgit/openstack/magnum/tree/magnum/drivers/mesos_ubuntu_v1/image/mesos/). Following are the typical steps to use the diskimage-builder tool on an Ubuntu server:

```
$ sudo apt-get update
$ sudo apt-get install git qemu-utils python-pip
$ sudo pip install diskimage-builder

$ git clone https://git.openstack.org/openstack/magnum
$ git clone https://git.openstack.org/openstack/dib-utils.git
$ git clone https://git.openstack.org/openstack/tripleo-image-elements.git
$ git clone https://git.openstack.org/openstack/heat-templates.git
$ export PATH="${PWD}/dib-utils/bin:$PATH"
$ export ELEMENTS_PATH=tripleo-image-elements/elements:heat-templates/hot/software-
config/elements:magnum/magnum/drivers/mesos_ubuntu_v1/image/mesos
$ export DIB_RELEASE=trusty

$ disk-image-create ubuntu vm docker mesos \
    os-collect-config os-refresh-config os-apply-config \
    heat-config heat-config-script \
    -o ubuntu-mesos.qcow2
```

5.15.1.2 Dockerfile

To build the image as above but within a Docker container, use the provided Dockerfile (http://git.openstack.org/cgit/openstack/magnum/tree/magnum/drivers/mesos_ubuntu_v1/image/Dockerfile). The output image will be saved as '/tmp/ubuntu-mesos.qcow2'. Following are the typical steps to run a Docker container to build the image:

```
$ git clone https://git.openstack.org/openstack/magnum
$ cd magnum/magnum/drivers/mesos_ubuntu_v1/image
$ sudo docker build -t magnum/mesos-builder .
```

```
$ sudo docker run -v /tmp:/output --rm -ti --privileged magnum/mesos-builder
...
Image file /output/ubuntu-mesos.qcow2 created...
```

5.15.2 Using Marathon

Marathon is a Mesos framework for long running applications. Docker containers can be deployed via Marathon's REST API. To get the endpoint for Marathon, run the cluster-show command and look for the property 'api_address'. Marathon's endpoint is port 8080 on this IP address, so the web console can be accessed at:

```
http://<api_address>:8080/
```

Refer to Marathon documentation for details on running applications. For example, you can 'post' a JSON app description to `http://<api_address>:8080/apps` to deploy a Docker container:

```
$ cat > app.json << END
{
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "libmesos/ubuntu"
    }
  },
  "id": "ubuntu",
  "instances": 1,
  "cpus": 0.5,
  "mem": 512,
  "uris": [],
  "cmd": "while sleep 10; do date -u +%T; done"
}
END
$ API_ADDRESS=$(magnum cluster-show mesos-cluster | awk '/ api_address /{print $4}')
$ curl -X POST -H "Content-Type: application/json" \
    http://${API_ADDRESS}:8080/v2/apps -d@app.json
```

5.16 Transport Layer Security

Magnum uses TLS to secure communication between a cluster's services and the outside world. TLS is a complex subject, and many guides on it exist already. This guide will not attempt to fully describe TLS, but instead will only cover the necessary steps to get a client set up to talk to a cluster with TLS. A more in-depth guide on TLS can be found in the OpenSSL Cookbook (https://www.feistyduck.com/books/openssl-cookbook/) by Ivan Ristić.

TLS is employed at 3 points in a cluster:

1. By Magnum to communicate with the cluster API endpoint
2. By the cluster worker nodes to communicate with the master nodes
3. By the end-user when they use the native client libraries to interact with the cluster. This applies to both a CLI or a program that uses a client for the particular cluster. Each client needs a valid certificate to authenticate and communicate with a cluster.

The first two cases are implemented internally by Magnum and are not exposed to the users, while the last case involves the users and is described in more details below.

5.16.1 Deploying a secure cluster

Current TLS support is summarized below:

C	TLS support
Kubernetes	yes
Swarm	yes
Mesos	no

For cluster type with TLS support, e.g. Kubernetes and Swarm, TLS is enabled by default. To disable TLS in Magnum, you can specify the parameter '–tls-disabled' in the ClusterTemplate. Please note it is not recommended to disable TLS due to security reasons.

In the following example, Kubernetes is used to illustrate a secure cluster, but the steps are similar for other cluster types that have TLS support.

First, create a ClusterTemplate; by default TLS is enabled in Magnum, therefore it does not need to be specified via a parameter:

```
magnum cluster-template-create secure-kubernetes \
                           --keypair default \
                           --external-network public \
                           --image fedora-atomic-latest \
                           --dns-nameserver 8.8.8.8 \
                           --flavor m1.small \
                           --docker-volume-size 3 \
                           --coe kubernetes \
                           --network-driver flannel

+-----------------------+--------------------------------------+
| Property              | Value                                |
+-----------------------+--------------------------------------+
| insecure_registry     | None                                 |
| http_proxy            | None                                 |
| updated_at            | None                                 |
| master_flavor_id      | None                                 |
| uuid                  | 5519b24a-621c-413c-832f-c30424528b31 |
| no_proxy              | None                                 |
| https_proxy           | None                                 |
| tls_disabled          | False                                |
| keypair_id            | time4funkey                          |
| public                | False                                |
| labels                | {}                                   |
| docker_volume_size    | 5                                    |
| server_type           | vm                                   |
| external_network_id   | public                               |
| cluster_distro        | fedora-atomic                        |
| image_id              | fedora-atomic-latest                 |
| volume_driver         | None                                 |
| registry_enabled      | False                                |
| docker_storage_driver | devicemapper                         |
| apiserver_port        | None                                 |
| name                  | secure-kubernetes                    |
| created_at            | 2016-07-25T23:09:50+00:00            |
| network_driver        | flannel                              |
| fixed_network         | None                                 |
| coe                   | kubernetes                           |
| flavor_id             | m1.small                             |
| dns_nameserver        | 8.8.8.8                              |
```

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

Now create a cluster. Use the ClusterTemplate name as a template for cluster creation:

```
magnum cluster-create secure-k8s-cluster \
                      --cluster-template secure-kubernetes \
                      --node-count 1

+--------------------+--------------------------------------------------------------+
| Property           | Value                                                        |
+--------------------+--------------------------------------------------------------+
| status             | CREATE_IN_PROGRESS                                           |
| uuid               | 3968ffd5-678d-4555-9737-35f191340fda                         |
| stack_id           | c96b66dd-2109-4ae2-b510-b3428f1e8761                         |
| status_reason      | None                                                         |
| created_at         | 2016-07-25T23:14:06+00:00                                    |
| updated_at         | None                                                         |
| create_timeout     | 0                                                            |
| api_address        | None                                                         |
| coe_version        | -                                                            |
| cluster_template_id| 5519b24a-621c-413c-832f-c30424528b31                         |
| master_addresses   | None                                                         |
| node_count         | 1                                                            |
| node_addresses     | None                                                         |
| master_count       | 1                                                            |
| container_version  | -                                                            |
| discovery_url      | https://discovery.etcd.io/ba52a8178e7364d43a323ee4387cf28e |
| name               | secure-k8s-cluster                                            |
+--------------------+--------------------------------------------------------------+
```

Now run cluster-show command to get the details of the cluster and verify that the api_address is ‘https’:

```
magnum cluster-show secure-k8scluster
+--------------------+--------------------------------------------------------------+
| Property           | Value                                                        |
+--------------------+--------------------------------------------------------------+
| status             | CREATE_COMPLETE                                              |
| uuid               | 04952c60-a338-437f-a7e7-d016d1d00e65                         |
| stack_id           | b7bf72ce-b08e-4768-8201-e63a99346898                         |
| status_reason      | Stack CREATE completed successfully                          |
| created_at         | 2016-07-25T23:14:06+00:00                                    |
| updated_at         | 2016-07-25T23:14:10+00:00                                    |
| create_timeout     | 60                                                           |
| coe_version        | v1.2.0                                                       |
| api_address        | https://192.168.19.86:6443                                   |
| cluster_template_id| da2825a0-6d09-4208-b39e-b2db666f1118                         |
```

```
| master_addresses   | ['192.168.19.87']                                              |
| node_count         | 1                                                              |
| node_addresses     | ['192.168.19.88']                                              |
| master_count       | 1                                                              |
| container_version  | 1.9.1                                                          |
| discovery_url      | https://discovery.etcd.io/3b7fb09733429d16679484673ba3bfd5 |
| name               | secure-k8s-cluster                                              |
+--------------------+----------------------------------------------------------------+
```

You can see the api_address contains https in the URL, showing that the Kubernetes services are configured securely with SSL certificates and now any communication to kube-apiserver will be over https.

5.16.2 Interfacing with a secure cluster

To communicate with the API endpoint of a secure cluster, you will need so supply 3 SSL artifacts:

1. Your client key
2. A certificate for your client key that has been signed by a Certificate Authority (CA)
3. The certificate of the CA

There are two ways to obtain these 3 artifacts.

5.16.2.1 Automated

Magnum provides the command 'cluster-config' to help the user in setting up the environment and artifacts for TLS, for example:

```
magnum cluster-config swarm-cluster --dir myclusterconfig
```

This will display the necessary environment variables, which you can add to your environment:

```
export DOCKER_HOST=tcp://172.24.4.5:2376
export DOCKER_CERT_PATH=myclusterconfig
export DOCKER_TLS_VERIFY=True
```

And the artifacts are placed in the directory specified:

```
ca.pem
```

```
cert.pem
key.pem
```

You can now use the native client to interact with the COE. The variables and artifacts are unique to the cluster.

The parameters for 'bay-config' are as follows:

–dir <dirname>

Directory to save the certificate and config files.

`--force`

Overwrite existing files in the directory specified.

5.16.2.2 Manual

You can create the key and certificates manually using the following steps.

Client Key

Your personal private key is essentially a cryptographically generated string of bytes. It should be protected in the same manner as a password. To generate an RSA key, you can use the 'genrsa' command of the 'openssl' tool:

```
openssl genrsa -out key.pem 4096
```

This command generates a 4096 byte RSA key at key.pem.

Signed Certificate

To authenticate your key, you need to have it signed by a CA. First generate the Certificate Signing Request (CSR). The CSR will be used by Magnum to generate a signed certificate that you will use to communicate with the cluster. To generate a CSR, openssl requires a config file that specifies a few values. Using the example template below, you can fill in the 'CN' value with your name and save it as client.conf:

```
$ cat > client.conf << END
[req]
distinguished_name = req_distinguished_name
req_extensions     = req_ext
prompt = no
[req_distinguished_name]
CN = Your Name
```

```
[req_ext]
extendedKeyUsage = clientAuth
END
```

Once you have client.conf, you can run the openssl 'req' command to generate the CSR:

```
openssl req -new -days 365 \
    -config client.conf \
    -key key.pem \
    -out client.csr
```

Now that you have your client CSR, you can use the Magnum CLI to send it off to Magnum to get it signed:

```
magnum ca-sign --cluster secure-k8s-cluster --csr client.csr > cert.pem
```

Certificate Authority

The final artifact you need to retrieve is the CA certificate for the cluster. This is used by your native client to ensure you are only communicating with hosts that Magnum set up:

```
magnum ca-show --cluster secure-k8s-cluster > ca.pem
```

Rotate Certificate

To rotate the CA certificate for a cluster and invalidate all user certificates, you can use the following command:

```
magnum ca-rotate --cluster secure-k8s-cluster
```

5.16.3 User Examples

Here are some examples for using the CLI on a secure Kubernetes and Swarm cluster. You can perform all the TLS set up automatically by:

```
eval $(magnum cluster-config <cluster-name>)
```

Or you can perform the manual steps as described above and specify the TLS options on the CLI. The SSL artifacts are assumed to be saved in local files as follows:

```
- key.pem: your SSL key
- cert.pem: signed certificate
- ca.pem: certificate for cluster CA
```

For Kubernetes, you need to get ‘kubectl’, a kubernetes CLI tool, to communicate with the cluster:

```
curl -O https://storage.googleapis.com/kubernetes-release/release/v1.2.0/bin/linux/amd64/
kubectl
chmod +x kubectl
sudo mv kubectl /usr/local/bin/kubectl
```

Now let’s run some ‘kubectl’ commands to check the secure communication. If you used ‘cluster-config’, then you can simply run the ‘kubectl’ command without having to specify the TLS options since they have been defined in the environment:

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"0", GitVersion:"v1.2.0",
 GitCommit:"cffae0523cfa80ddf917aba69f08508b91f603d5", GitTreeState:"clean"}
Server Version: version.Info{Major:"1", Minor:"0", GitVersion:"v1.2.0",
 GitCommit:"cffae0523cfa80ddf917aba69f08508b91f603d5", GitTreeState:"clean"}
```

You can specify the TLS options manually as follows:

```
KUBERNETES_URL=$(magnum cluster-show secure-k8s-cluster |
                 awk '/ api_address /{print $4}')
kubectl version --certificate-authority=ca.pem \
                --client-key=key.pem \
                --client-certificate=cert.pem -s $KUBERNETES_URL

kubectl create -f redis-master.yaml --certificate-authority=ca.pem \
                                    --client-key=key.pem \
                                    --client-certificate=cert.pem -s $KUBERNETES_URL

pods/test2

kubectl get pods --certificate-authority=ca.pem \
                 --client-key=key.pem \
                 --client-certificate=cert.pem -s $KUBERNETES_URL
NAME           READY     STATUS    RESTARTS   AGE
redis-master   2/2       Running   0          1m
```

Beside using the environment variables, you can also configure ‘kubectl’ to remember the TLS options:

```
kubectl config set-cluster secure-k8s-cluster --server=${KUBERNETES_URL} \
    --certificate-authority=${PWD}/ca.pem
kubectl config set-credentials client --certificate-authority=${PWD}/ca.pem \
    --client-key=${PWD}/key.pem --client-certificate=${PWD}/cert.pem
kubectl config set-context secure-k8scluster --cluster=secure-k8scluster --user=client
kubectl config use-context secure-k8scluster
```

Then you can use 'kubectl' commands without the certificates:

```
kubectl get pods
NAME           READY     STATUS    RESTARTS   AGE
redis-master   2/2       Running   0          1m
```

Access to Kubernetes User Interface:

```
curl -L ${KUBERNETES_URL}/ui --cacert ca.pem --key key.pem \
    --cert cert.pem
```

You may also set up 'kubectl' proxy which will use your client certificates to allow you to browse to a local address to use the UI without installing a certificate in your browser:

```
kubectl proxy --api-prefix=/ --certificate-authority=ca.pem --client-key=key.pem \
              --client-certificate=cert.pem -s $KUBERNETES_URL
```

You can then open http://localhost:8001/ui in your browser.

The examples for Docker are similar. With 'cluster-config' set up, you can just run docker commands without TLS options. To specify the TLS options manually:

```
docker -H tcp://192.168.19.86:2376 --tlsverify \
       --tlscacert ca.pem \
       --tlskey key.pem \
       --tlscert cert.pem \
       info
```

5.16.4 Storing the certificates

Magnum generates and maintains a certificate for each cluster so that it can also communicate securely with the cluster. As a result, it is necessary to store the certificates in a secure manner. Magnum provides the following methods for storing the certificates and this is configured in /etc/magnum/magnum.conf in the section [certificates] with the parameter 'cert_manager_type'.

1. Barbican: Barbican is a service in OpenStack for storing secrets. It is used by Magnum to store the certificates when cert_manager_type is configured as:

   ```
   cert_manager_type = barbican
   ```

   This is the recommended configuration for a production environment. Magnum will interface with Barbican to store and retrieve certificates, delegating the task of securing the certificates to Barbican.

2. Magnum database: In some cases, a user may want an alternative to storing the certificates that does not require Barbican. This can be a development environment, or a private cloud that has been secured by other means. Magnum can store the certificates in its own database; this is done with the configuration:

   ```
   cert_manager_type = x509keypair
   ```

   This storage mode is only as secure as the controller server that hosts the database for the OpenStack services.

3. Local store: As another alternative that does not require Barbican, Magnum can simply store the certificates on the local host filesystem where the conductor is running, using the configuration:

   ```
   cert_manager_type = local
   ```

   Note that this mode is only supported when there is a single Magnum conductor running since the certificates are stored locally. The 'local' mode is not recommended for a production environment.

For the nodes, the certificates for communicating with the masters are stored locally and the nodes are assumed to be secured.

5.17 Networking

There are two components that make up the networking in a cluster.

1. The Neutron infrastructure for the cluster: this includes the private network, subnet, ports, routers, load balancers, etc.

2. The networking model presented to the containers: this is what the containers see in communicating with each other and to the external world. Typically this consists of a driver deployed on each node.

The two components are deployed and managed separately. The Neutron infrastructure is the integration with OpenStack; therefore, it is stable and more or less similar across different COE types. The networking model, on the other hand, is specific to the COE type and is still under active development in the various COE communities, for example, Docker libnetwork (https://github.com/docker/libnetwork) and Kubernetes Container Networking (https://github.com/ku-

bernetes/kubernetes/blob/release-1.1/docs/design/networking.md). As a result, the implementation for the networking models is evolving and new models are likely to be introduced in the future.

For the Neutron infrastructure, the following configuration can be set in the ClusterTemplate:

external-network

The external Neutron network ID to connect to this cluster. This is used to connect the cluster to the external internet, allowing the nodes in the cluster to access external URL for discovery, image download, etc. If not specified, the default value is "public" and this is valid for a typical devstack.

fixed-network

The Neutron network to use as the private network for the cluster nodes. If not specified, a new Neutron private network will be created.

dns-nameserver

The DNS nameserver to use for this cluster. This is an IP address for the server and it is used to configure the Neutron subnet of the cluster (dns_nameservers). If not specified, the default DNS is 8.8.8.8, the publicly available DNS.

http-proxy, https-proxy, no-proxy

The proxy for the nodes in the cluster, to be used when the cluster is behind a firewall and containers cannot access URL's on the external internet directly. For the parameter http-proxy and https-proxy, the value to provide is a URL and it will be set in the environment variable HTTP_PROXY and HTTPS_PROXY respectively in the nodes. For the parameter no-proxy, the value to provide is an IP or list of IP's separated by comma. Likewise, the value will be set in the environment variable NO_PROXY in the nodes.

For the networking model to the container, the following configuration can be set in the ClusterTemplate:

network-driver

The network driver name for instantiating container networks. Currently, the following network drivers are supported:

I v	K b n	S	Mesos
Flannel	supported	supported	supported
Docker	unsupported	supported	supported

If not specified, the default driver is Flannel for Kubernetes, and Docker for Swarm and Mesos.

Particular network driver may require its own set of parameters for configuration, and these parameters are specified through the labels in the ClusterTemplate. Labels are arbitrary key=value pairs.

When Flannel is specified as the network driver, the following optional labels can be added:

IPv4 network in CIDR format to use for the entire Flannel network. If not specified, the default is 10.100.0.0/16.

The size of the subnet allocated to each host. If not specified, the default is 24.

The type of backend for Flannel. Possible values are *udp, vxlan, host-gw*. If not specified, the default is *udp*. Selecting the best backend depends on your networking. Generally, *udp* is the most generally supported backend since there is little requirement on the network, but it typically offers the lowest performance. The *vxlan* backend performs better, but requires vxlan support in the kernel so the image used to provision the nodes needs to include this support. The *host-gw* backend offers the best performance since it does not actually encapsulate messages, but it requires all the nodes to be on the same L2 network. The

private Neutron network that Magnum creates does meet this requirement; therefore if the parameter *fixed_network* is not specified in the ClusterTemplate, *host-gw* is the best choice for the Flannel backend.

5.18 High Availability

To be filled in

5.19 Scaling

5.19.1 Performance tuning for periodic task

Magnum's periodic task performs a `stack-get` operation on the Heat stack underlying each of its clusters. If you have a large amount of clusters this can create considerable load on the Heat API. To reduce that load you can configure Magnum to perform one global `stack-list` per periodic task instead of one per cluster. This is disabled by default, both from the Heat and Magnum side since it causes a security issue, though: any user in any tenant holding the `admin` role can perform a global `stack-list` operation if Heat is configured to allow it for Magnum. If you want to enable it nonetheless, proceed as follows:

1. Set `periodic_global_stack_list` in magnum.conf to `True` (`False` by default).
2. Update heat policy to allow magnum list stacks. To this end, edit your heat policy file, usually etc/heat/policy.json``:

   ```
   ...
   stacks:global_index: "rule:context_is_admin",
   ```

   Now restart heat.

5.19.2 Containers and nodes

Scaling containers and nodes refers to increasing or decreasing allocated system resources. Scaling is a broad topic and involves many dimensions. In the context of Magnum in this guide, we consider the following issues:

- Scaling containers and scaling cluster nodes (infrastructure)
- Manual and automatic scaling

Since this is an active area of development, a complete solution covering all issues does not exist yet, but partial solutions are emerging.

Scaling containers involves managing the number of instances of the container by replicating or deleting instances. This can be used to respond to change in the workload being supported by the application; in this case, it is typically driven by certain metrics relevant to the application such as response time, etc. Other use cases include rolling upgrade, where a new version of a service can gradually be scaled up while the older version is gradually scaled down. Scaling containers is supported at the COE level and is specific to each COE as well as the version of the COE. You will need to refer to the documentation for the proper COE version for full details, but following are some pointers for reference.

For Kubernetes, pods are scaled manually by setting the count in the replication controller. Kubernetes version 1.3 and later also supports autoscaling (http://blog.kubernetes.io/2016/07/autoscaling-in-kubernetes.html). For Docker, the tool 'Docker Compose' provides the command docker-compose scale (https://docs.docker.com/compose/reference/scale/) which lets you manually set the number of instances of a container. For Swarm version 1.12 and later, services can also be scaled manually through the command docker service scale (https://docs.docker.com/engine/swarm/swarm-tutorial/scale-service/). Automatic scaling for Swarm is not yet available. Mesos manages the resources and does not support scaling directly; instead, this is provided by frameworks running within Mesos. With the Marathon framework currently supported in the Mesos cluster, you can use the scale operation (https://mesosphere.github.io/marathon/docs/application-basics.html) on the Marathon UI or through a REST API call to manually set the attribute 'instance' for a container.

Scaling the cluster nodes involves managing the number of nodes in the cluster by adding more nodes or removing nodes. There is no direct correlation between the number of nodes and the number of containers that can be hosted since the resources consumed (memory, CPU, etc) depend on the containers. However, if a certain resource is exhausted in the cluster, adding

more nodes would add more resources for hosting more containers. As part of the infrastructure management, Magnum supports manual scaling through the attribute 'node_count' in the cluster, so you can scale the cluster simply by changing this attribute:

```
magnum cluster-update mycluster replace node_count=2
```

Adding nodes to a cluster is straightforward: Magnum deploys additional VMs or baremetal servers through the heat templates and invokes the COE-specific mechanism for registering the new nodes to update the available resources in the cluster. Afterward, it is up to the COE or user to re-balance the workload by launching new container instances or re-launching dead instances on the new nodes.

Removing nodes from a cluster requires some more care to ensure continuous operation of the containers since the nodes being removed may be actively hosting some containers. Magnum performs a simple heuristic that is specific to the COE to find the best node candidates for removal, as follows:

Kubernetes

Magnum scans the pods in the namespace 'Default' to determine the nodes that are *not* hosting any (empty nodes). If the number of nodes to be removed is equal or less than the number of these empty nodes, these nodes will be removed from the cluster. If the number of nodes to be removed is larger than the number of empty nodes, a warning message will be sent to the Magnum log and the empty nodes along with additional nodes will be removed from the cluster. The additional nodes are selected randomly and the pods running on them will be deleted without warning. For this reason, a good practice is to manage the pods through the replication controller so that the deleted pods will be relaunched elsewhere in the cluster. Note also that even when only the empty nodes are removed, there is no guarantee that no pod will be deleted because there is no locking to ensure that Kubernetes will not launch new pods on these nodes after Magnum has scanned the pods.

Swarm

No node selection heuristic is currently supported. If you decrease the node_count, a node will be chosen by magnum without consideration of what containers are running on the selected node.

Mesos

Magnum scans the running tasks on Marathon server to determine the nodes on which there is *no* task running (empty nodes). If the number of nodes to be removed is equal or less than the number of these empty nodes, these nodes will be removed from the cluster.

If the number of nodes to be removed is larger than the number of empty nodes, a warning message will be sent to the Magnum log and the empty nodes along with additional nodes will be removed from the cluster. The additional nodes are selected randomly and the containers running on them will be deleted without warning. Note that even when only the empty nodes are removed, there is no guarantee that no container will be deleted because there is no locking to ensure that Mesos will not launch new containers on these nodes after Magnum has scanned the tasks.

Currently, scaling containers and scaling cluster nodes are handled separately, but in many use cases, there are interactions between the two operations. For instance, scaling up the containers may exhaust the available resources in the cluster, thereby requiring scaling up the cluster nodes as well. Many complex issues are involved in managing this interaction. A presentation at the OpenStack Tokyo Summit 2015 covered some of these issues along with some early proposals, Exploring Magnum and Senlin integration for autoscaling containers (https://www.openstack.org/summit/tokyo-2015/videos/presentation/exploring-magnum-and-senlin-integration-for-autoscaling-containers). This remains an active area of discussion and research.

5.20 Storage

Currently Cinder provides the block storage to the containers, and the storage is made available in two ways: as ephemeral storage and as persistent storage.

5.20.1 Ephemeral storage

The filesystem for the container consists of multiple layers from the image and a top layer that holds the modification made by the container. This top layer requires storage space and the storage is configured in the Docker daemon through a number of storage options. When the container is removed, the storage allocated to the particular container is also deleted.

Magnum can manage the containers' filesystem in two ways, storing them on the local disk of the compute instances or in a separate Cinder block volume for each node in the cluster, mounts it to the node and configures it to be used as ephemeral storage. Users can specify the size of the Cinder volume with the ClusterTemplate attribute 'docker-volume-size'. Currently the block size is fixed at cluster creation time, but future lifecycle operations may allow modifying the block size during the life of the cluster.

For drivers that support additional volumes for container storage, a label named 'docker_volume_type' is exposed so that users can select different cinder volume types for their volumes. The default volume *must* be set in 'default_docker_volume_type' in the 'cinder' section of magnum.conf, an obvious value is the default volume type set in cinder.conf of your cinder deployment . Please note, that docker_volume_type refers to a cinder volume type and it is unrelated to docker or kubernetes volumes.

Both local disk and the Cinder block storage can be used with a number of Docker storage drivers available.

- 'devicemapper': When used with a dedicated Cinder volume it is configured using direct-lvm and offers very good performance. If it's used with the compute instance's local disk uses a loopback device offering poor performance and it's not recommended for production environments. Using the 'devicemapper' driver does allow the use of SELinux.
- 'overlay' When used with a dedicated Cinder volume offers as good or better performance than devicemapper. If used on the local disk of the compute instance (especially with high IOPS drives) you can get significant performance gains. However, for kernel versions less than 4.9, SELinux must be disabled inside the containers resulting in worse container isolation, although it still runs in enforcing mode on the cluster compute instances.

5.20.2 Persistent storage

In some use cases, data read/written by a container needs to persist so that it can be accessed later. To persist the data, a Cinder volume with a filesystem on it can be mounted on a host and be made available to the container, then be unmounted when the container exits.

Docker provides the 'volume' feature for this purpose: the user invokes the 'volume create' command, specifying a particular volume driver to perform the actual work. Then this volume can be mounted when a container is created. A number of third-party volume drivers support OpenStack Cinder as the backend, for example Rexray and Flocker. Magnum currently supports Rexray as the volume driver for Swarm and Mesos. Other drivers are being considered.

Kubernetes allows a previously created Cinder block to be mounted to a pod and this is done by specifying the block ID in the pod YAML file. When the pod is scheduled on a node, Kubernetes will interface with Cinder to request the volume to be mounted on this node, then Kubernetes will launch the Docker container with the proper options to make the filesystem on the Cinder

volume accessible to the container in the pod. When the pod exits, Kubernetes will again send a request to Cinder to unmount the volume's filesystem, making it available to be mounted on other nodes.

Magnum supports these features to use Cinder as persistent storage using the ClusterTemplate attribute 'volume-driver' and the support matrix for the COE types is summarized as follows:

I v	K b n	S v	M s	os
ci de	su pr ed	su po ed	un- su pp ed	un- sup- port- ed
re su po ed	ex su pp ed	ma su pp ed	un- sup- port- ed	

Following are some examples for using Cinder as persistent storage.

5.20.2.1 Using Cinder in Kubernetes

NOTE: This feature requires Kubernetes version 1.5.0 or above. The public Fedora image from Atomic currently meets this requirement.

1. Create the ClusterTemplate.
   Specify 'cinder' as the volume-driver for Kubernetes:

```
magnum cluster-template-create k8s-cluster-template \
                           --image fedora-23-atomic-7 \
                           --keypair testkey \
                           --external-network public \
                           --dns-nameserver 8.8.8.8 \
                           --flavor m1.small \
                           --docker-volume-size 5 \
                           --network-driver flannel \
```

```
                            --coe kubernetes \
                            --volume-driver cinder
```

2. Create the cluster:

```
magnum cluster-create k8s-cluster \
                      --cluster-template k8s-cluster-template \
                      --node-count 1
```

Kubernetes is now ready to use Cinder for persistent storage. Following is an example illustrating how Cinder is used in a pod.

1. Create the cinder volume:

```
cinder create --display-name=test-repo 1

XML:ID=$(cinder create --display-name=test-repo 1 | awk -F'|' '$2~/^[[:space:]]*id/
 {print $3}')
```

   The command will generate the volume with a ID. The volume ID will be specified in Step 2.

2. Create a pod in this cluster and mount this cinder volume to the pod. Create a file (e.g nginx-cinder.yaml) describing the pod:

```
cat > nginx-cinder.yaml << END
apiVersion: v1
kind: Pod
metadata:
  name: aws-web
spec:
  containers:
    - name: web
      image: nginx
      ports:
        - name: web
          containerPort: 80
          hostPort: 8081
          protocol: TCP
      volumeMounts:
        - name: html-volume
          mountPath: "/usr/share/nginx/html"
  volumes:
    - name: html-volume
      cinder:
        # Enter the volume ID below
```

```
        volumeID: $ID
        fsType: ext4
END
```

NOTE: The Cinder volume ID needs to be configured in the YAML file so the existing Cinder volume can be mounted in a pod by specifying the volume ID in the pod manifest as follows:

```
volumes:
- name: html-volume
  cinder:
    volumeID: $ID
    fsType: ext4
```

- Create the pod by the normal Kubernetes interface:

```
kubectl create -f nginx-cinder.yaml
```

You can start a shell in the container to check that the mountPath exists, and on an OpenStack client you can run the command 'cinder list' to verify that the cinder volume status is 'in-use'.

5.20.2.2 Using Cinder in Swarm

To be filled in

5.20.2.3 Using Cinder in Mesos

1. Create the ClusterTemplate.
   Specify 'rexray' as the volume-driver for Mesos. As an option, you can specify in a label the attributes 'rexray_preempt' to enable any host to take control of a volume regardless if other hosts are using the volume. If this is set to false, the driver will ensure data safety by locking the volume:

```
magnum cluster-template-create mesos-cluster-template \
                           --image ubuntu-mesos \
                           --keypair testkey \
                           --external-network public \
                           --dns-nameserver 8.8.8.8 \
                           --master-flavor m1.magnum \
                           --docker-volume-size 4 \
                           --tls-disabled \
```

```
                            --flavor m1.magnum \
                            --coe mesos \
                            --volume-driver rexray \
                            --labels rexray-preempt=true
```

2. Create the Mesos cluster:

```
magnum cluster-create mesos-cluster \
                      --cluster-template mesos-cluster-template \
                      --node-count 1
```

3. Create the cinder volume and configure this cluster:

```
cinder create --display-name=redisdata 1
```

   Create the following file

```
cat > mesos.json << END
{
  "id": "redis",
  "container": {
    "docker": {
    "image": "redis",
    "network": "BRIDGE",
    "portMappings": [
      { "containerPort": 80, "hostPort": 0, "protocol": "tcp"}
    ],
    "parameters": [
       { "key": "volume-driver", "value": "rexray" },
       { "key": "volume", "value": "redisdata:/data" }
    ]
    }
 },
 "cpus": 0.2,
 "mem": 32.0,
 "instances": 1
}
END
```

NOTE: When the Mesos cluster is created using this ClusterTemplate, the Mesos cluster will be configured so that a filesystem on an existing cinder volume can be mounted in a container by configuring the parameters to mount the cinder volume in the JSON file

```
"parameters": [
   { "key": "volume-driver", "value": "rexray" },
   { "key": "volume", "value": "redisdata:/data" }
```

```
]
```

- Create the container using Marathon REST API

```
MASTER_IP=$(magnum cluster-show mesos-cluster | awk '/ api_address /{print $4}')
curl -X POST -H "Content-Type: application/json" \
http://${MASTER_IP}:8080/v2/apps -d@mesos.json
```

You can log into the container to check that the mountPath exists, and you can run the command 'cinder list' to verify that your cinder volume status is 'in-use'.

5.21 Image Management

When a COE is deployed, an image from Glance is used to boot the nodes in the cluster and then the software will be configured and started on the nodes to bring up the full cluster. An image is based on a particular distro such as Fedora, Ubuntu, etc, and is prebuilt with the software specific to the COE such as Kubernetes, Swarm, Mesos. The image is tightly coupled with the following in Magnum:

1. Heat templates to orchestrate the configuration.
2. Template definition to map ClusterTemplate parameters to Heat template parameters.
3. Set of scripts to configure software.

Collectively, they constitute the driver for a particular COE and a particular distro; therefore, developing a new image needs to be done in conjunction with developing these other components. Image can be built by various methods such as diskimagebuilder, or in some case, a distro image can be used directly. A number of drivers and the associated images is supported in Magnum as reference implementation. In this section, we focus mainly on the supported images.

All images must include support for cloud-init and the heat software configuration utility:

- os-collect-config
- os-refresh-config
- os-apply-config
- heat-config
- heat-config-script

Additional software are described as follows.

5.21.1 Kubernetes on Fedora Atomic

This image can be downloaded from the public Atomic site (https://alt.fedoraproject.org/pub/alt/atomic/stable/Cloud-Images/x86_64/Images/) or can be built locally using diskimagebuilder. Details can be found in the fedora-atomic element (https://github.com/openstack/magnum/tree/master/magnum/elements/fedora-atomic) The image currently has the following OS/software:

OS version	
Fedora	26
Docker	1.13.1
Kubernetes	1.7.4
etcd	3.1.3
Flannel	0.7.0

The following software are managed as systemd services:

- kube-apiserver
- kubelet
- etcd
- flannel (if specified as network driver)
- docker

The following software are managed as Docker containers:

- kube-controller-manager
- kube-scheduler
- kube-proxy

The login for this image is *fedora*.

5.21.2 Kubernetes on CoreOS

CoreOS publishes a stock image (http://beta.release.core-os.net/amd64-usr/current/coreos_production_openstack_image.img.bz2) that is being used to deploy Kubernetes. This image has the following OS/software:

Software	Version
CoreOS	4.3.6
Docker	1.9.1
Kubernetes	1.0.6
etcd	2.2.3
Flannel	0.5.5

The following software are managed as systemd services:

- kubelet
- flannel (if specified as network driver)

- docker
- etcd

The following software are managed as Docker containers:

- kube-apiserver
- kube-controller-manager
- kube-scheduler
- kube-proxy

The login for this image is *core*.

5.21.3 Kubernetes on Ironic

This image is built manually using diskimagebuilder. The scripts and instructions are included in Magnum code repo (https://github.com/openstack/magnum/tree/master/magnum/templates/kubernetes/elements). Currently Ironic is not fully supported yet, therefore more details will be provided when this driver has been fully tested.

5.21.4 Swarm on Fedora Atomic

This image is the same as the image for Kubernetes on Fedora Atomic described above. The login for this image is *fedora*.

5.21.5 Mesos on Ubuntu

This image is built manually using diskimagebuilder. The Fedora site hosts the current image ubuntu-mesos-latest.qcow2 (https://fedorapeople.org/groups/magnum/ubuntu-mesos-latest.qcow2).

OS	version
Ubuntu	14.04
Docker	1.8.1
Mesos	0.25.0
Marathon	0.11.1

5.22 Notification

Magnum provides notifications about usage data so that 3rd party applications can use the data for auditing, billing, monitoring, or quota purposes. This document describes the current inclusions and exclusions for Magnum notifications.

Magnum uses Cloud Auditing Data Federation (CADF (http://www.dmtf.org/standards/cadf)) Notification as its notification format for better support of auditing, details about CADF are documented below.

5.22.1 Auditing with CADF

Magnum uses the PyCADF (http://docs.openstack.org/developer/pycadf) library to emit CADF notifications, these events adhere to the DMTF CADF (http://www.dmtf.org/standards/cadf) specification. This standard provides auditing capabilities for compliance with security, operational, and business processes and supports normalized and categorized event data for federation and aggregation.

Below table describes the event model components and semantics for each component:

model component	CADF Definition
OBSERVER	The RESOURCE that generates the CADF Event Record based on its observation (directly or indirectly) of the Actual Event.
INITIATOR	The RESOURCE that initiated, originated, or instigated the event's ACTION, accord-

model component	CADF Definition
	ing to the OBSERVER.
ACTION	The operation or activity the INITIATOR has performed, has attempted to perform or has pending against the event's TARGET, according to the OBSERVER.
TARGET	The RESOURCE against which

model component	CADF Definition
	the ACTION of a CADF Event Record was performed, attempted, or is pending, according to the OBSERVER.
OUTCOME	The result or status of the ACTION against the TARGET, according to the OBSERVER.

The `payload` portion of a CADF Notification is a CADF `event`, which is represented as a JSON dictionary. For example:

```
{
```

```
    "typeURI": "http://schemas.dmtf.org/cloud/audit/1.0/event",
    "initiator": {
        "typeURI": "service/security/account/user",
        "host": {
            "agent": "curl/7.22.0(x86_64-pc-linux-gnu)",
            "address": "127.0.0.1"
        },
        "id": "<initiator_id>"
    },
    "target": {
        "typeURI": "<target_uri>",
        "id": "openstack:1c2fc591-facb-4479-a327-520dade1ea15"
    },
    "observer": {
        "typeURI": "service/security",
        "id": "openstack:3d4a50a9-2b59-438b-bf19-c231f9c7625a"
    },
    "eventType": "activity",
    "eventTime": "2014-02-14T01:20:47.932842+00:00",
    "action": "<action>",
    "outcome": "success",
    "id": "openstack:f5352d7b-bee6-4c22-8213-450e7b646e9f",
}
```

Where the following are defined:

- `<initiator_id>`: ID of the user that performed the operation
- `<target_uri>`: CADF specific target URI, (i.e.: data/security/project)
- `<action>`: The action being performed, typically: `<operation>`. `<resource_type>`

Additionally there may be extra keys present depending on the operation being performed, these will be discussed below.

Note, the `eventType` property of the CADF payload is different from the `event_type` property of a notifications. The former (`eventType`) is a CADF keyword which designates the type of event that is being measured, this can be: `activity`, `monitor` or `control`. Whereas the latter (`event_type`) is described in previous sections as: `magnum.<resource_type>.<operation>`

5.22.2 Supported Events

The following table displays the corresponding relationship between resource types and operations. The bay type is deprecated and will be removed in a future version. Cluster is the new equivalent term.

resource	supported operations	typeURI
bay	create, update, delete	service/magnum/bay
cluster	create, update, delete	service/magnum/cluster

5.22.3 Example Notification - Cluster Create

The following is an example of a notification that is sent when a cluster is created. This example can be applied for any `create`, `update` or `delete` event that is seen in the table above. The `<action>` and `typeURI` fields will be change.

```
{
    "event_type": "magnum.cluster.created",
    "message_id": "0156ee79-b35f-4cef-ac37-d4a85f231c69",
    "payload": {
        "typeURI": "http://schemas.dmtf.org/cloud/audit/1.0/event",
        "initiator": {
            "typeURI": "service/security/account/user",
            "id": "c9f76d3c31e142af9291de2935bde98a",
            "user_id": "0156ee79-b35f-4cef-ac37-d4a85f231c69",
            "project_id": "3d4a50a9-2b59-438b-bf19-c231f9c7625a"
        },
        "target": {
            "typeURI": "service/magnum/cluster",
```

```
            "id": "openstack:1c2fc591-facb-4479-a327-520dade1ea15"
        },
        "observer": {
            "typeURI": "service/magnum/cluster",
            "id": "openstack:3d4a50a9-2b59-438b-bf19-c231f9c7625a"
        },
        "eventType": "activity",
        "eventTime": "2015-05-20T01:20:47.932842+00:00",
        "action": "create",
        "outcome": "success",
        "id": "openstack:f5352d7b-bee6-4c22-8213-450e7b646e9f",
        "resource_info": "671da331c47d4e29bb6ea1d270154ec3"
    }
    "priority": "INFO",
    "publisher_id": "magnum.host1234",
    "timestamp": "2016-05-20 15:03:45.960280"
}
```

5.23 Container Monitoring

The offered monitoring stack relies on the following set of containers and services:

- cAdvisor
- Node Exporter
- Prometheus
- Grafana

To setup this monitoring stack, users are given two configurable labels in the Magnum cluster template's definition:

This label accepts a boolean value. If *True*, the monitoring stack will be setup. By default *prometheus_monitoring = False*.

This label lets users create their own *admin* user password for the Grafana interface. It expects a string value. By default it is set to *admin*.

5.23.1 Container Monitoring in Kubernetes

By default, all Kubernetes clusters already contain *cAdvisor* integrated with the *Kubelet* binary. Its container monitoring data can be accessed on a node level basis through *http://NODE_IP:4194*.

Node Exporter is part of the above mentioned monitoring stack as it can be used to export machine metrics. Such functionality also work on a node level which means that when `prometheus-monitoring` is *True*, the Kubernetes nodes will be populated with an additional manifest under */etc/kubernetes/manifests*. Node Exporter is then automatically picked up and launched as a regular Kubernetes POD.

To aggregate and complement all the existing monitoring metrics and add a built-in visualization layer, Prometheus is used. It is launched by the Kubernetes master node(s) as a *Service* within a *Deployment* with one replica and it relies on a *ConfigMap* where the Prometheus configuration (prometheus.yml) is defined. This configuration uses Prometheus native support for service discovery in Kubernetes clusters, *kubernetes_sd_configs*. The respective manifests can be found in */srv/kubernetes/monitoring/* on the master nodes and once the service is up and running, Prometheus UI can be accessed through port 9090.

Finally, for custom plotting and enhanced metric aggregation and visualization, Prometheus can be integrated with Grafana as it provides native compliance for Prometheus data sources. Also Grafana is deployed as a *Service* within a *Deployment* with one replica. The default user is *admin* and the password is setup according to `grafana-admin-passwd`. There is also a default Grafana dashboard provided with this installation, from the official Grafana dashboards' repository (https://grafana.net/dashboards). The Prometheus data source is automatically added to Grafana once it is up and running, pointing to *http://prometheus:9090* through *Proxy*. The respective manifests can also be found in */srv/kubernetes/monitoring/* on the master nodes and once the service is running, the Grafana dashboards can be accessed through port 3000.

For both Prometheus and Grafana, there is an assigned *systemd* service called *kube-enable-monitoring*.

5.24 Kubernetes External Load Balancer

In a Kubernetes cluster, all masters and minions are connected to a private Neutron subnet, which in turn is connected by a router to the public network. This allows the nodes to access each other and the external internet.

All Kubernetes pods and services created in the cluster are connected to a private container network which by default is Flannel, an overlay network that runs on top of the Neutron private subnet. The pods and services are assigned IP addresses from this container network and they can access each other and the external internet. However, these IP addresses are not accessible from an external network.

To publish a service endpoint externally so that the service can be accessed from the external network, Kubernetes provides the external load balancer feature. This is done by simply specifying the attribute “type: LoadBalancer” in the service manifest. When the service is created, Kubernetes will add an external load balancer in front of the service so that the service will have an external IP address in addition to the internal IP address on the container network. The service endpoint can then be accessed with this external IP address. Refer to the Kubernetes service document (https://kubernetes.io/docs/concepts/services-networking/service/#type-loadbalancer) for more details.

A Kubernetes cluster deployed by Magnum will have all the necessary configuration required for the external load balancer. This document describes how to use this feature.

5.24.1 Steps for the cluster administrator

Because the Kubernetes master needs to interface with OpenStack to create and manage the Neutron load balancer, we need to provide a credential for Kubernetes to use.

In the current implementation, the cluster administrator needs to manually perform this step. We are looking into several ways to let Magnum automate this step in a secure manner. This means that after the Kubernetes cluster is initially deployed, the load balancer support is disabled. If the administrator does not want to enable this feature, no further action is required. All the services will be created normally; services that specify the load balancer will also be created successfully, but a load balancer will not be created.

Note that different versions of Kubernetes require different versions of Neutron LBaaS plugin running on the OpenStack instance:

```
============================  ==============================
Kubernetes Version on Master  Neutron LBaaS Version Required
============================  ==============================
1.2                           LBaaS v1
1.3 or later                  LBaaS v2
============================  ==============================
```

Before enabling the Kubernetes load balancer feature, confirm that the OpenStack instance is running the required version of Neutron LBaaS plugin. To determine if your OpenStack instance is running LBaaS v1, try running the following command from your OpenStack control node:

```
neutron lb-pool-list
```

Or look for the following configuration in neutron.conf or neutron_lbaas.conf:

```
service_provider =
 LOADBALANCER:Haproxy:neutron_lbaas.services.loadbalancer.drivers.haproxy.plugin_driver.HaproxyOnHostPl
```

To determine if your OpenStack instance is running LBaaS v2, try running the following command from your OpenStack control node:

```
neutron lbaas-pool-list
```

Or look for the following configuration in neutron.conf or neutron_lbaas.conf:

```
service_plugins = neutron.plugins.services.agent_loadbalancer.plugin.LoadBalancerPluginv2
```

To configure LBaaS v1 or v2, refer to the Neutron documentation.

Before deleting the Kubernetes cluster, make sure to delete all the services that created load balancers. Because the Neutron objects created by Kubernetes are not managed by Heat, they will not be deleted by Heat and this will cause the cluster-delete operation to fail. If this occurs, delete the neutron objects manually (lb-pool, lb-vip, lb-member, lb-healthmonitor) and then run cluster-delete again.

5.24.2 Steps for the users

This feature requires the OpenStack cloud provider to be enabled. To do so, enable the cinder support (–volume-driver cinder).

For the user, publishing the service endpoint externally involves the following 2 steps:

1. Specify “type: LoadBalancer” in the service manifest
2. After the service is created, associate a floating IP with the VIP of the load balancer pool.

The following example illustrates how to create an external endpoint for a pod running nginx.

Create a file (e.g nginx.yaml) describing a pod running nginx:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
   app: nginx
spec:
```

```
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

Create a file (e.g nginx-service.yaml) describing a service for the nginx pod:

```
apiVersion: v1
kind: Service
metadata:
  name: nginxservice
  labels:
    app: nginx
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
  selector:
    app: nginx
  type: LoadBalancer
```

Please refer to the quickstart (https://docs.openstack.org/developer/magnum/userguide.html) guide on how to connect to Kubernetes running on the launched cluster. Assuming a Kubernetes cluster named k8sclusterv1 has been created, deploy the pod and service using following commands:

```
kubectl create -f nginx.yaml

kubectl create -f nginx-service.yaml
```

For more details on verifying the load balancer in OpenStack, refer to the following section on how it works.

Next, associate a floating IP to the load balancer. This can be done easily on Horizon by navigating to:

```
Compute -> Access & Security -> Floating IPs
```

Click on “Allocate IP To Project” and then on “Associate” for the new floating IP.

Alternatively, associating a floating IP can be done on the command line by allocating a floating IP, finding the port of the VIP, and associating the floating IP to the port. The commands shown below are for illustration purpose and assume that there is only one service with load balancer running in the cluster and no other load balancers exist except for those created for the cluster.

First create a floating IP on the public network:

```
neutron floatingip-create public

Created a new floatingip:

+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    |                                      |
| floating_ip_address | 172.24.4.78                          |
| floating_network_id | 4808eacb-e1a0-40aa-97b6-ecb745af2a4d |
| id                  | b170eb7a-41d0-4c00-9207-18ad1c30fecf |
| port_id             |                                      |
| router_id           |                                      |
| status              | DOWN                                 |
| tenant_id           | 012722667dc64de6bf161556f49b8a62     |
+---------------------+--------------------------------------+
```

Note the floating IP 172.24.4.78 that has been allocated. The ID for this floating IP is shown above, but it can also be queried by:

```
FLOATING_XML:ID=$(neutron floatingip-list | grep "172.24.4.78" | awk '{print $2}')
```

Next find the VIP for the load balancer:

```
VIP_XML:ID=$(neutron lb-vip-list | grep TCP | grep -v pool | awk '{print $2}')
```

Find the port for this VIP:

```
PORT_XML:ID=$(neutron lb-vip-show $VIP_ID | grep port_id | awk '{print $4}')
```

Finally associate the floating IP with the port of the VIP:

```
neutron floatingip-associate $FLOATING_ID $PORT_ID
```

The endpoint for nginx can now be accessed on a browser at this floating IP:

```
http://172.24.4.78:80
```

Alternatively, you can check for the nginx ‘welcome’ message by:

```
curl http://172.24.4.78:80
```

NOTE: it is not necessary to indicate port :80 here but it is shown to correlate with the port that was specified in the service manifest.

5.24.3 How it works

Kubernetes is designed to work with different Clouds such as Google Compute Engine (GCE), Amazon Web Services (AWS), and OpenStack; therefore, different load balancers need to be created on the particular Cloud for the services. This is done through a plugin for each Cloud and the OpenStack plugin was developed by Angus Lees:

```
https://github.com/kubernetes/kubernetes/blob/release-1.0/pkg/cloudprovider/openstack/
openstack.go
```

When the Kubernetes components kube-apiserver and kube-controller-manager start up, they will use the credential provided to authenticate a client to interface with OpenStack.

When a service with load balancer is created, the plugin code will interface with Neutron in this sequence:

1. Create lb-pool for the Kubernetes service
2. Create lb-member for the minions
3. Create lb-healthmonitor
4. Create lb-vip on the private network of the Kubernetes cluster

These Neutron objects can be verified as follows. For the load balancer pool:

```
neutron lb-pool-list
+--------------------------------------
+--------------------------------------------------+----------+-------------+----------
+----------------+--------+
| id                                   | name
 | provider | lb_method   | protocol | admin_state_up | status |
+--------------------------------------
+--------------------------------------------------+----------+-------------+----------
+----------------+--------+
| 241357b3-2a8f-442e-b534-bde7cd6ba7e4 | a1f03e40f634011e59c9efa163eae8ab
 | haproxy  | ROUND_ROBIN | TCP      | True           | ACTIVE |
| 82b39251-1455-4eb6-a81e-802b54c2df29 | k8sclusterv1-iypacicrskib-api_pool-fydshw7uvr7h
 | haproxy  | ROUND_ROBIN | HTTP     | True           | ACTIVE |
| e59ea983-c6e8-4cec-975d-89ade6b59e50 | k8sclusterv1-iypacicrskib-etcd_pool-qbpo43ew2m3x
 | haproxy  | ROUND_ROBIN | HTTP     | True           | ACTIVE |
+--------------------------------------
+--------------------------------------------------+----------+-------------+----------
+----------------+--------+
```

Note that 2 load balancers already exist to implement high availability for the cluster (api and ectd). The new load balancer for the Kubernetes service uses the TCP protocol and has a name assigned by Kubernetes.

For the members of the pool:

```
neutron lb-member-list
+--------------------------------------+----------+---------------+--------
+---------------+--------+
| id                                   | address  | protocol_port | weight |
 admin_state_up | status |
+--------------------------------------+----------+---------------+--------
+---------------+--------+
| 9ab7dcd7-6e10-4d9f-ba66-861f4d4d627c | 10.0.0.5 |          8080 |      1 | True
   | ACTIVE |
| b179c1ad-456d-44b2-bf83-9cdc127c2b27 | 10.0.0.5 |          2379 |      1 | True
   | ACTIVE |
| f222b60e-e4a9-4767-bc44-ffa66ec22afe | 10.0.0.6 |         31157 |      1 | True
   | ACTIVE |
+--------------------------------------+----------+---------------+--------
+---------------+--------+
```

Again, 2 members already exist for high availability and they serve the master node at 10.0.0.5. The new member serves the minion at 10.0.0.6, which hosts the Kubernetes service.

For the monitor of the pool:

```
neutron lb-healthmonitor-list
+--------------------------------------+------+----------------+
| id                                   | type | admin_state_up |
+--------------------------------------+------+----------------+
| 381d3d35-7912-40da-9dc9-b2322d5dda47 | TCP  | True           |
| 67f2ae8f-ffc6-4f86-ba5f-1a135f4af85c | TCP  | True           |
| d55ff0f3-9149-44e7-9b52-2e055c27d1d3 | TCP  | True           |
+--------------------------------------+------+----------------+
```

For the VIP of the pool:

```
neutron lb-vip-list
+--------------------------------------+----------------------------------+----------
+----------+---------------+--------+
| id                                   | name                             | address  |
 protocol | admin_state_up | status |
+--------------------------------------+----------------------------------+----------
+----------+---------------+--------+
| 9ae2ebfb-b409-4167-9583-4a3588d2ff42 | api_pool.vip                     | 10.0.0.3 |
 HTTP     | True           | ACTIVE |
```

```
| c318aec6-8b7b-485c-a419-1285a7561152 | a1f03e40f634011e59c9efa163eae8ab | 10.0.0.7 |
 TCP      | True           | ACTIVE |
| fc62cf40-46ad-47bd-aa1e-48339b95b011 | etcd_pool.vip                    | 10.0.0.4 |
 HTTP     | True           | ACTIVE |
+--------------------------------------+----------------------------------+----------
+----------+---------------+--------+
```

Note that the VIP is created on the private network of the cluster; therefore it has an internal IP address of 10.0.0.7. This address is also associated as the "external address" of the Kubernetes service. You can verify this in Kubernetes by running following command:

```
kubectl get services
NAME           LABELS                                    SELECTOR    IP(S)
 PORT(S)
kubernetes     component=apiserver,provider=kubernetes   <none>      10.254.0.1
 443/TCP
nginxservice   app=nginx                                 app=nginx   10.254.122.191   80/
TCP
                                                                     10.0.0.7
```

On GCE, the networking implementation gives the load balancer an external address automatically. On OpenStack, we need to take the additional step of associating a floating IP to the load balancer.

5.25 Terminology

Cluster (previously Bay)

A cluster is the construct in which Magnum launches container orchestration engines. After a cluster has been created the user is able to add containers to it either directly, or in the case of the Kubernetes container orchestration engine within pods - a logical construct specific to that implementation. A cluster is created based on a ClusterTemplate.

ClusterTemplate (previously BayModel)

A ClusterTemplate in Magnum is roughly equivalent to a flavor in Nova. It acts as a template that defines options such as the container orchestration engine, keypair and image for use when Magnum is creating clusters using the given ClusterTemplate.

Container Orchestration Engine (COE)

A container orchestration engine manages the lifecycle of one or more containers, logically represented in Magnum as a cluster. Magnum supports a number of container orchestration engines, each with their own pros and cons, including Docker Swarm, Kubernetes, and Mesos.

5.26 Overview

Magnum is an OpenStack API service developed by the OpenStack Containers Team making container orchestration engines (COE) such as Docker Swarm, Kubernetes and Apache Mesos available as the first class resources in OpenStack.

Magnum uses Heat to orchestrate an OS image which contains Docker and COE and runs that image in either virtual machines or bare metal in a cluster configuration.

Magnum offers complete life-cycle management of COEs in an OpenStack environment, integrated with other OpenStack services for a seamless experience for OpenStack users who wish to run containers in an OpenStack environment.

Following are few salient features of Magnum:

- Standard API based complete life-cycle management for Container Clusters
- Multi-tenancy for container clusters
- Choice of COE: Kubernetes, Swarm, Mesos, DC/OS
- Choice of container cluster deployment model: VM or Bare-metal
- Keystone-based multi-tenant security and auth management
- Neutron based multi-tenant network control and isolation
- Cinder based volume service for containers
- Integrated with OpenStack: SSO experience for cloud users
- Secure container cluster access (TLS enabled)

More details: Magnum Project Wiki (https://wiki.openstack.org/wiki/Magnum)

5.27 ClusterTemplate

A ClusterTemplate (previously known as BayModel) is a collection of parameters to describe how a cluster can be constructed. Some parameters are relevant to the infrastructure of the cluster, while others are for the particular COE. In a typical workflow, a user would create a ClusterTemplate, then create one or more clusters using the ClusterTemplate. A cloud provider can also define a number of ClusterTemplates and provide them to the users. A ClusterTemplate cannot be updated or deleted if a cluster using this ClusterTemplate still exists.

The definition and usage of the parameters of a ClusterTemplate are as follows. They are loosely grouped as: mandatory, infrastructure, COE specific.

<name>

Name of the ClusterTemplate to create. The name does not have to be unique. If multiple ClusterTemplates have the same name, you will need to use the UUID to select the ClusterTemplate when creating a cluster or updating, deleting a ClusterTemplate. If a name is not specified, a random name will be generated using a string and a number, for example “pi-13-model”.

–coe <coe>

Specify the Container Orchestration Engine to use. Supported COE’s include ‘kubernetes’, ‘swarm’, ‘mesos’. If your environment has additional cluster drivers installed, refer to the cluster driver documentation for the new COE names. This is a mandatory parameter and there is no default value.

–image <image>

The name or UUID of the base image in Glance to boot the servers for the cluster. The image must have the attribute ‘os_distro’ defined as appropriate for the cluster driver. For the currently supported images, the os_distro names are:

C	os-distro
Kubernetes	Fedora-atomic, CoreOS
Swarm	Fedora-atomic

C	os-distro
Mesos	Ubuntu

This is a mandatory parameter and there is no default value.

–keypair <keypair>

The name of the SSH keypair to configure in the cluster servers for ssh access. You will need the key to be able to ssh to the servers in the cluster. The login name is specific to the cluster driver. If keypair is not provided in template it will be required at Cluster create. This value will be overridden by any keypair value that is provided during Cluster create.

–external-network <external-network>

The name or network ID of a Neutron network to provide connectivity to the external internet for the cluster. This network must be an external network, i.e. its attribute 'router:external' must be 'True'. The servers in the cluster will be connected to a private network and Magnum will create a router between this private network and the external network. This will allow the servers to download images, access discovery service, etc, and the containers to install packages, etc. In the opposite direction, floating IP's will be allocated from the external network to provide access from the external internet to servers and the container services hosted in the cluster. This is a mandatory parameter and there is no default value.

`--public`

Access to a ClusterTemplate is normally limited to the admin, owner or users within the same tenant as the owners. Setting this flag makes the ClusterTemplate public and accessible by other users. The default is not public.

–server-type <server-type>

The servers in the cluster can be VM or baremetal. This parameter selects the type of server to create for the cluster. The default is 'vm'. Possible values are 'vm', 'bm'.

–network-driver <network-driver>

The name of a network driver for providing the networks for the containers. Note that this is different and separate from the Neutron network for the cluster. The operation and networking model are specific to the particular driver. Supported network drivers and the default driver are:

COE	Network Driver	Default Driver
Kubernetes	Flannel	Flannel
Swarm	Docker, Flannel	Flannel
Mesos	Docker	Docker

–volume-driver <volume-driver>

The name of a volume driver for managing the persistent storage for the containers. The functionality supported are specific to the driver. Supported volume drivers and the default driver are:

COE	Volume Driver	Default Driver
Kubernetes	Cinder	No Driver
Swarm	Rexray	No Driver

C	V u ve	De- fau	Dri-
Me	Ro	Na	ay Dri- ver

–dns-nameserver <dns-nameserver>

The DNS nameserver for the servers and containers in the cluster to use. This is configured in the private Neutron network for the cluster. The default is ‘8.8.8.8’.

–flavor <flavor>

The nova flavor id for booting the node servers. The default is ‘m1.small’.

–master-flavor <master-flavor>

The nova flavor id for booting the master or manager servers. The default is ‘m1.small’.

–http-proxy <http-proxy>

The IP address for a proxy to use when direct http access from the servers to sites on the external internet is blocked. This may happen in certain countries or enterprises, and the proxy allows the servers and containers to access these sites. The format is a URL including a port number. The default is ‘None’.

–https-proxy <https-proxy>

The IP address for a proxy to use when direct https access from the servers to sites on the external internet is blocked. This may happen in certain countries or enterprises, and the proxy allows the servers and containers to access these sites. The format is a URL including a port number. The default is ‘None’.

–no-proxy <no-proxy>

When a proxy server is used, some sites should not go through the proxy and should be accessed normally. In this case, you can specify these sites as a comma separated list of IP’s. The default is ‘None’.

–docker-volume-size <docker-volume-size>

If specified, container images will be stored in a cinder volume of the specified size in GB. Each cluster node will have a volume attached of the above size. If not specified, images will be stored in the compute instance's local disk. For the 'devicemapper' storage driver, the minimum value is 3GB. For the 'overlay' storage driver, the minimum value is 1GB. This value can be overridden at cluster creation.

–docker-storage-driver <docker-storage-driver>

The name of a driver to manage the storage for the images and the container's writable layer. The supported drivers are 'devicemapper' and 'overlay'. The default is 'devicemapper'.

–labels <KEY1=VALUE1,KEY2=VALUE2;KEY3=VALUE3...>

Arbitrary labels in the form of key=value pairs. The accepted keys and valid values are defined in the cluster drivers. They are used as a way to pass additional parameters that are specific to a cluster driver. Refer to the subsection on labels for a list of the supported key/value pairs and their usage.

`--tls-disabled`

Transport Layer Security (TLS) is normally enabled to secure the cluster. In some cases, users may want to disable TLS in the cluster, for instance during development or to troubleshoot certain problems. Specifying this parameter will disable TLS so that users can access the COE endpoints without a certificate. The default is TLS enabled.

`--registry-enabled`

Docker images by default are pulled from the public Docker registry, but in some cases, users may want to use a private registry. This option provides an alternative registry based on the Registry V2: Magnum will create a local registry in the cluster backed by swift to host the images. Refer to Docker Registry 2.0 (https://github.com/docker/distribution) for more details. The default is to use the public registry.

`--master-lb-enabled`

Since multiple masters may exist in a cluster, a load balancer is created to provide the API endpoint for the cluster and to direct requests to the masters. In some cases, such as when the LBaaS service is not available, this option can be set to 'false' to create a cluster without the load balancer. In this case, one of the masters will serve as the API endpoint. The default is 'true', i.e. to create the load balancer for the cluster.

5.27.1 Labels

Labels is a general method to specify supplemental parameters that are specific to certain COE or associated with certain options. Their format is key/value pair and their meaning is interpreted by the drivers that uses them. The drivers do validate the key/value pairs. Their usage is explained in details in the appropriate sections, however, since there are many possible labels, the following table provides a summary to help give a clearer picture. The label keys in the table are linked to more details elsewhere in the user guide.

label key	label value	default
`flannel-network-cidr`	IPv4 CIDR	10.100.0.0/16
`flannel-backend`		• udp • vxlan • host-gw
`flannel-network-subnetlen`	size of subnet to assign to node	24
`rexray-preempt`		• true • false

label key	label value	default
`mesos-slave-isolation`		• filesystem/posix • filesystem/linux • filesystem/shared • posix/cpu • posix/mem • posix/disk • cgroups/cpu • cgroups/mem • docker/runtime • namespaces/pid

label key	label value	default
`mesos-slave-image-providers`		• "" (empty string) • `appc` • `docker` • `appc,docker`
`mesos-slave-work-dir`	(directory name)	""
`mesos-slave-executor-env-variables`	(file name)	""
`swarm-strategy`		• `spread` • `binpack` • `random`

label key	label value	default
`admission-control-list`	see below	see below
`prometheus-monitoring`		• true • false
`grafana-admin-passwd`	(any string)	“admin”
`kube-tag`	see below	see below
`kube-dashboard-enabled`		• true • false
`docker-volume-type`	see below	see below
`etcd-volume-size`	etcd storage	0

label key	label value	default
	volume size	

5.28 Cluster

A cluster (previously known as bay) is an instance of the ClusterTemplate of a COE. Magnum deploys a cluster by referring to the attributes defined in the particular ClusterTemplate as well as a few additional parameters for the cluster. Magnum deploys the orchestration templates provided by the cluster driver to create and configure all the necessary infrastructure. When ready, the cluster is a fully operational COE that can host containers.

5.28.1 Infrastructure

The infrastructure of the cluster consists of the resources provided by the various OpenStack services. Existing infrastructure, including infrastructure external to OpenStack, can also be used by the cluster, such as DNS, public network, public discovery service, Docker registry. The actual resources created depends on the COE type and the options specified; therefore you need to refer to the cluster driver documentation of the COE for specific details. For instance, the option '–master-lb-enabled' in the ClusterTemplate will cause a load balancer pool along with the health monitor and floating IP to be created. It is important to distinguish resources in the IaaS level from resources in the PaaS level. For instance, the infrastructure networking in OpenStack IaaS is different and separate from the container networking in Kubernetes or Swarm PaaS.

Typical infrastructure includes the following.

Servers

The servers host the containers in the cluster and these servers can be VM or bare metal. VM's are provided by Nova. Since multiple VM's are hosted on a physical server, the VM's provide the isolation needed for containers between different tenants running on the same physical server. Bare metal servers are provided by Ironic and are used when peak performance with virtually no overhead is needed for the containers.

Identity

Keystone provides the authentication and authorization for managing the cluster infrastructure.

Network

Networking among the servers is provided by Neutron. Since COE currently are not multi-tenant, isolation for multi-tenancy on the networking level is done by using a private network for each cluster. As a result, containers belonging to one tenant will not be accessible to containers or servers of another tenant. Other networking resources may also be used, such as load balancer and routers. Networking among containers can be provided by Kuryr if needed.

Storage

Cinder provides the block storage that can be used to host the containers and as persistent storage for the containers.

Security

Barbican provides the storage of secrets such as certificates used for Transport Layer Security (TLS) within the cluster.

5.28.2 Life cycle

The set of life cycle operations on the cluster is one of the key value that Magnum provides, enabling clusters to be managed painlessly on OpenStack. The current operations are the basic CRUD operations, but more advanced operations are under discussion in the community and will be implemented as needed.

NOTE The OpenStack resources created for a cluster are fully accessible to the cluster owner. Care should be taken when modifying or reusing these resources to avoid impacting Magnum operations in unexpected manners. For instance, if you launch your own Nova instance on the cluster private network, Magnum would not be aware of this instance. Therefore, the cluster-delete operation will fail because Magnum would not delete the extra Nova instance and the private Neutron network cannot be removed while a Nova instance is still attached.

NOTE Currently Heat nested templates are used to create the resources; therefore if an error occurs, you can troubleshoot through Heat. For more help on Heat stack troubleshooting, refer to the Troubleshooting Guide (https://github.com/openstack/magnum/blob/master/doc/source/troubleshooting-guide.rst#heat-stacks).

5.28.2.1 Create

NOTE bay-<command> are the deprecated versions of these commands and are still support in current release. They will be removed in a future version. Any references to the term bay will be replaced in the parameters when using the 'bay' versions of the commands. For example, in 'bay-create' –baymodel is used as the baymodel parameter for this command instead of –cluster-template.

The 'cluster-create' command deploys a cluster, for example:

```
magnum cluster-create mycluster \
                  --cluster-template mytemplate \
                  --node-count 8 \
                  --master-count 3
```

The 'cluster-create' operation is asynchronous; therefore you can initiate another 'cluster-create' operation while the current cluster is being created. If the cluster fails to be created, the infrastructure created so far may be retained or deleted depending on the particular orchestration engine. As a common practice, a failed cluster is retained during development for troubleshooting, but they are automatically deleted in production. The current cluster drivers use Heat templates and the resources of a failed 'cluster-create' are retained.

The definition and usage of the parameters for 'cluster-create' are as follows:

<name>

Name of the cluster to create. If a name is not specified, a random name will be generated using a string and a number, for example "gamma-7-cluster".

–cluster-template <cluster-template>

The ID or name of the ClusterTemplate to use. This is a mandatory parameter. Once a ClusterTemplate is used to create a cluster, it cannot be deleted or modified until all clusters that use the ClusterTemplate have been deleted.

–keypair <keypair>

The name of the SSH keypair to configure in the cluster servers for ssh access. You will need the key to be able to ssh to the servers in the cluster. The login name is specific to the cluster driver. If keypair is not provided it will attempt to use the value in the ClusterTemplate. If the ClusterTemplate is also missing a keypair value then an error will be returned. The keypair value provided here will override the keypair value from the ClusterTemplate.

–node-count <node-count>

The number of servers that will serve as node in the cluster. The default is 1.

–master-count <master-count>

The number of servers that will serve as master for the cluster. The default is 1. Set to more than 1 master to enable High Availability. If the option '–master-lb-enabled' is specified in the ClusterTemplate, the master servers will be placed in a load balancer pool.

–discovery-url <discovery-url>

The custom discovery url for node discovery. This is used by the COE to discover the servers that have been created to host the containers. The actual discovery mechanism varies with the COE. In some cases, Magnum fills in the server info in the discovery service. In other cases, if the discovery-url is not specified, Magnum will use the public discovery service at:

```
https://discovery.etcd.io
```

In this case, Magnum will generate a unique url here for each cluster and store the info for the servers.

–timeout <timeout>

The timeout for cluster creation in minutes. The value expected is a positive integer and the default is 60 minutes. If the timeout is reached during cluster-create, the operation will be aborted and the cluster status will be set to 'CREATE_FAILED'.

5.28.2.2 List

The 'cluster-list' command lists all the clusters that belong to the tenant, for example:

```
magnum cluster-list
```

5.28.2.3 Show

The ‘cluster-show’ command prints all the details of a cluster, for example:

```
magnum cluster-show mycluster
```

The properties include those not specified by users that have been assigned default values and properties from new resources that have been created for the cluster.

5.28.2.4 Update

A cluster can be modified using the ‘cluster-update’ command, for example:

```
magnum cluster-update mycluster replace node_count=8
```

The parameters are positional and their definition and usage are as follows.

<cluster>
: This is the first parameter, specifying the UUID or name of the cluster to update.

<op>
: This is the second parameter, specifying the desired change to be made to the cluster attributes. The allowed changes are ‘add’, ‘replace’ and ‘remove’.

<attribute=value>
: This is the third parameter, specifying the targeted attributes in the cluster as a list separated by blank space. To add or replace an attribute, you need to specify the value for the attribute. To remove an attribute, you only need to specify the name of the attribute. Currently the only attribute that can be replaced or removed is ‘node_count’. The attributes ‘name’, ‘master_count’ and ‘discovery_url’ cannot be replaced or delete. The table below summarizes the possible change to a cluster.

Attribute	replace	remove
node_count	update node_count to the desired number	not supported

Attribute	replace	remove
		fault of 1
master_count	no	no
name	no	no
discovery_url	no	no

The ‘cluster-update’ operation cannot be initiated when another operation is in progress.

NOTE: The attribute names in cluster-update are slightly different from the corresponding names in the cluster-create command: the dash ‘-‘ is replaced by an underscore ‘_’. For instance, ‘node-count’ in cluster-create is ‘node_count’ in cluster-update.

5.28.2.5 Scale

Scaling a cluster means adding servers to or removing servers from the cluster. Currently, this is done through the ‘cluster-update’ operation by modifying the node-count attribute, for example:

```
magnum cluster-update mycluster replace node_count=2
```

When some nodes are removed, Magnum will attempt to find nodes with no containers to remove. If some nodes with containers must be removed, Magnum will log a warning message.

5.28.2.6 Delete

The ‘cluster-delete’ operation removes the cluster by deleting all resources such as servers, network, storage; for example:

```
magnum cluster-delete mycluster
```

The only parameter for the cluster-delete command is the ID or name of the cluster to delete. Multiple clusters can be specified, separated by a blank space.

If the operation fails, there may be some remaining resources that have not been deleted yet. In this case, you can troubleshoot through Heat. If the templates are deleted manually in Heat, you can delete the cluster in Magnum to clean up the cluster from Magnum database.

The 'cluster-delete' operation can be initiated when another operation is still in progress.

5.29 Python Client

5.29.1 Installation

Follow the instructions in the OpenStack Installation Guide to enable the repositories for your distribution:

- RHEL/CentOS/Fedora (http://docs.openstack.org/liberty/install-guide-rdo/)
- Ubuntu/Debian (http://docs.openstack.org/liberty/install-guide-ubuntu/)
- openSUSE/SUSE Linux Enterprise (http://docs.openstack.org/liberty/install-guide-obs/)

Install using distribution packages for RHEL/CentOS/Fedora:

```
$ sudo yum install python-magnumclient
```

Install using distribution packages for Ubuntu/Debian:

```
$ sudo apt-get install python-magnumclient
```

Install using distribution packages for OpenSuSE and SuSE Enterprise Linux:

```
$ sudo zypper install python-magnumclient
```

5.29.2 Verifying installation

Execute the `magnum` command with the `–version` argument to confirm that the client is installed and in the system path:

```
$ magnum --version
1.1.0
```

Note that the version returned may differ from the above, 1.1.0 was the latest available version at the time of writing.

5.29.3 Using the command-line client

Refer to the OpenStack Command-Line Interface Reference (http://docs.openstack.org/cli-reference/magnum.html) for a full list of the commands supported by the `magnum` command-line client.

5.30 Horizon Interface

Magnum provides a Horizon plugin so that users can access the Container Infrastructure Management service through the OpenStack browser-based graphical UI. The plugin is available from magnum-ui (https://github.com/openstack/magnum-ui). It is not installed by default in the standard Horizon service, but you can follow the instruction for installing a Horizon plugin (http://docs.openstack.org/developer/horizon/tutorials/plugin.html#installing-your-plugin).

In Horizon, the container infrastructure panel is part of the 'Project' view and it currently supports the following operations:

- View list of cluster templates
- View details of a cluster template
- Create a cluster template
- Delete a cluster template
- View list of clusters
- View details of a cluster
- Create a cluster
- Delete a cluster
- Get the Certificate Authority for a cluster
- Sign a user key and obtain a signed certificate for accessing the secured COE API endpoint in a cluster.

Other operations are not yet supported and the CLI should be used for these.

5.31 Cluster Drivers

A cluster driver is a collection of python code, heat templates, scripts, images, and documents for a particular COE on a particular distro. Magnum presents the concept of ClusterTemplates and clusters. The implementation for a particular cluster type is provided by the cluster driver. In other words, the cluster driver provisions and manages the infrastructure for the COE. Magnum includes default drivers for the following COE and distro pairs:

COE	distro
Kubernetes	Fedora Atomic
Kubernetes	CoreOS
Swarm	Fedora Atomic
Mesos	Ubuntu

Magnum is designed to accommodate new cluster drivers to support custom COE's and this section describes how a new cluster driver can be constructed and enabled in Magnum.

5.31.1 Directory structure

Magnum expects the components to be organized in the following directory structure under the directory ‘drivers’:

```
COE_Distro/
   image/
   templates/
   api.py
   driver.py
   monitor.py
   scale.py
   template_def.py
   version.py
```

The minimum required components are:

driver.py

Python code that implements the controller operations for the particular COE. The driver must implement: Currently supported: `cluster_create`, `cluster_update`, `cluster_delete`.

templates

A directory of orchestration templates for managing the lifecycle of clusters, including creation, configuration, update, and deletion. Currently only Heat templates are supported, but in the future other orchestration mechanism such as Ansible may be supported.

template_def.py

Python code that maps the parameters from the ClusterTemplate to the input parameters for the orchestration and invokes the orchestration in the templates directory.

version.py

Tracks the latest version of the driver in this directory. This is defined by a `version` attribute and is represented in the form of `1.0.0`. It should also include a `Driver` attribute with descriptive name such as `fedora_swarm_atomic`.

The remaining components are optional:

image

Instructions for obtaining or building an image suitable for the COE.

api.py

Python code to interface with the COE.

monitor.py

Python code to monitor the resource utilization of the cluster.

scale.py

Python code to scale the cluster by adding or removing nodes.

5.31.2 Sample cluster driver

To help developers in creating new COE drivers, a minimal cluster driver is provided as an example. The 'docker' cluster driver will simply deploy a single VM running Ubuntu with the latest Docker version installed. It is not a true cluster, but the simplicity will help to illustrate the key concepts.

To be filled in

5.31.3 Installing a cluster driver

To be filled in

5.32 Cluster Type Definition

There are three key pieces to a Cluster Type Definition:

1. Heat Stack template - The HOT file that Magnum will use to generate a cluster using a Heat Stack.
2. Template definition - Magnum's interface for interacting with the Heat template.
3. Definition Entry Point - Used to advertise the available Cluster Types.

5.32.1 The Heat Stack Template

The Heat Stack Template is where most of the real work happens. The result of the Heat Stack Template should be a full Container Orchestration Environment.

5.32.2 The Template Definition

Template definitions are a mapping of Magnum object attributes and Heat template parameters, along with Magnum consumable template outputs. A Cluster Type Definition indicates which Cluster Types it can provide. Cluster Types are how Magnum determines which of the enabled Cluster Type Definitions it will use for a given cluster.

5.32.3 The Definition Entry Point

Entry points are a standard discovery and import mechanism for Python objects. Each Template Definition should have an Entry Point in the `magnum.template_definitions` group. This example exposes it's Template Definition as `example_template = example_template:ExampleTemplate` in the `magnum.template_definitions` group.

5.32.4 Installing Cluster Templates

Because Cluster Type Definitions are basically Python projects, they can be worked with like any other Python project. They can be cloned from version control and installed or uploaded to a package index and installed via utilities such as pip.

Enabling a Cluster Type is as simple as adding it's Entry Point to the `enabled_definitions` config option in magnum.conf.:

```
# Setup python environment and install Magnum

$ virtualenv .venv
$ source .venv/bin/active
(.venv)$ git clone https://github.com/openstack/magnum.git
(.venv)$ cd magnum
(.venv)$ python setup.py install

# List installed templates, notice default templates are enabled

(.venv)$ magnum-template-manage list-templates
Enabled Templates
  magnum_vm_atomic_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster.yaml
  magnum_vm_coreos_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster-coreos.yaml
Disabled Templates

# Install example template
```

```
(.venv)$ cd contrib/templates/example
(.venv)$ python setup.py install

# List installed templates, notice example template is disabled

(.venv)$ magnum-template-manage list-templates
Enabled Templates
  magnum_vm_atomic_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster.yaml
  magnum_vm_coreos_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster-coreos.yaml
Disabled Templates
  example_template: /home/example/.venv/local/lib/python2.7/site-packages/
ExampleTemplate-0.1-py2.7.egg/example_template/example.yaml

# Enable example template by setting enabled_definitions in magnum.conf

(.venv)$ sudo mkdir /etc/magnum
(.venv)$ sudo bash -c "cat > /etc/magnum/magnum.conf << END_CONF
[bay]
enabled_definitions=magnum_vm_atomic_k8s,magnum_vm_coreos_k8s,example_template
END_CONF"

# List installed templates, notice example template is now enabled

(.venv)$ magnum-template-manage list-templates
Enabled Templates
  example_template: /home/example/.venv/local/lib/python2.7/site-packages/
ExampleTemplate-0.1-py2.7.egg/example_template/example.yaml
  magnum_vm_atomic_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster.yaml
  magnum_vm_coreos_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster-coreos.yaml
Disabled Templates

# Use --details argument to get more details about each template

(.venv)$ magnum-template-manage list-templates --details
Enabled Templates
  example_template: /home/example/.venv/local/lib/python2.7/site-packages/
ExampleTemplate-0.1-py2.7.egg/example_template/example.yaml
     Server_Type  OS       CoE
     vm         example  example_coe
  magnum_vm_atomic_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster.yaml
     Server_Type   OS             CoE
```

```
     vm        fedora-atomic  kubernetes
  magnum_vm_coreos_k8s: /home/example/.venv/local/lib/python2.7/site-packages/magnum/
templates/kubernetes/kubecluster-coreos.yaml
     Server_Type  OS      CoE
     vm         coreos  kubernetes
Disabled Templates
```

5.33 Heat Stack Templates

Heat Stack Templates are what Magnum passes to Heat to generate a cluster. For each ClusterTemplate resource in Magnum, a Heat stack is created to arrange all of the cloud resources needed to support the container orchestration environment. These Heat stack templates provide a mapping of Magnum object attributes to Heat template parameters, along with Magnum consumable stack outputs. Magnum passes the Heat Stack Template to the Heat service to create a Heat stack. The result is a full Container Orchestration Environment.

5.34 Choosing a COE

Magnum supports a variety of COE options, and allows more to be added over time as they gain popularity. As an operator, you may choose to support the full variety of options, or you may want to offer a subset of the available choices. Given multiple choices, your users can run one or more clusters, and each may use a different COE. For example, I might have multiple clusters that use Kubernetes, and just one cluster that uses Swarm. All of these clusters can run concurrently, even though they use different COE software.

Choosing which COE to use depends on what tools you want to use to manage your containers once you start your app. If you want to use the Docker tools, you may want to use the Swarm cluster type. Swarm will spread your containers across the various nodes in your cluster automatically. It does not monitor the health of your containers, so it can't restart them for you if they stop. It will not automatically scale your app for you (as of Swarm version 1.2.2). You may view this as a plus. If you prefer to manage your application yourself, you might prefer swarm over the other COE options.

Kubernetes (as of v1.2) is more sophisticated than Swarm (as of v1.2.2). It offers an attractive YAML file description of a pod, which is a grouping of containers that run together as part of a distributed application. This file format allows you to model your application deployment using

a declarative style. It has support for auto scaling and fault recovery, as well as features that allow for sophisticated software deployments, including canary deploys and blue/green deploys. Kubernetes is very popular, especially for web applications.

Apache Mesos is a COE that has been around longer than Kubernetes or Swarm. It allows for a variety of different frameworks to be used along with it, including Marathon, Aurora, Chronos, Hadoop, and a number of others. (http://mesos.apache.org/documentation/latest/frameworks/)

The Apache Mesos framework design can be used to run alternate COE software directly on Mesos. Although this approach is not widely used yet, it may soon be possible to run Mesos with Kubernetes and Swarm as frameworks, allowing you to share the resources of a cluster between multiple different COEs. Until this option matures, we encourage Magnum users to create multiple clusters, and use the COE in each cluster that best fits the anticipated workload.

Finding the right COE for your workload is up to you, but Magnum offers you a choice to select among the prevailing leading options. Once you decide, see the next sections for examples of how to create a cluster with your desired COE.

5.35 Native Clients

Magnum preserves the native user experience with a COE and does not provide a separate API or client. This means you will need to use the native client for the particular cluster type to interface with the clusters. In the typical case, there are two clients to consider:

COE level

This is the orchestration or management level such as Kubernetes, Swarm, Mesos and its frameworks.

Container level

This is the low level container operation. Currently it is Docker for all clusters.

The clients can be CLI and/or browser-based. You will need to refer to the documentation for the specific native client and appropriate version for details, but following are some pointers for reference.

Kubernetes CLI is the tool ‘kubectl’, which can be simply copied from a node in the cluster or downloaded from the Kubernetes release. For instance, if the cluster is running Kubernetes release 1.2.0, the binary for ‘kubectl’ can be downloaded as and set up locally as follows:

```
curl -O https://storage.googleapis.com/kubernetes-release/release/v1.2.0/bin/linux/amd64/
kubectl
```

```
chmod +x kubectl
sudo mv kubectl /usr/local/bin/kubectl
```

Kubernetes also provides a browser UI. If the cluster has the Kubernetes Dashboard running; it can be accessed using:

```
eval $(magnum cluster-config <cluster-name>)
kubectl proxy

The browser can be accessed at http://localhost:8001/ui
```

For Swarm, the main CLI is 'docker', along with associated tools such as 'docker-compose', etc. Specific version of the binaries can be obtained from the Docker Engine installation (https://docs.docker.com/engine/installation/binaries/).

Mesos cluster uses the Marathon framework.

Depending on the client requirement, you may need to use a version of the client that matches the version in the cluster. To determine the version of the COE and container, use the command 'cluster-show' and look for the attribute *coe_version* and *container_version*:

```
magnum cluster-show k8s-cluster
+--------------------+------------------------------------------------------------+
| Property           | Value                                                      |
+--------------------+------------------------------------------------------------+
| status             | CREATE_COMPLETE                                            |
| uuid               | 04952c60-a338-437f-a7e7-d016d1d00e65                       |
| stack_id           | b7bf72ce-b08e-4768-8201-e63a99346898                       |
| status_reason      | Stack CREATE completed successfully                        |
| created_at         | 2016-07-25T23:14:06+00:00                                  |
| updated_at         | 2016-07-25T23:14:10+00:00                                  |
| create_timeout     | 60                                                         |
| coe_version        | v1.2.0                                                     |
| api_address        | https://192.168.19.86:6443                                 |
| cluster_template_id| da2825a0-6d09-4208-b39e-b2db666f1118                       |
| master_addresses   | ['192.168.19.87']                                          |
| node_count         | 1                                                          |
| node_addresses     | ['192.168.19.88']                                          |
| master_count       | 1                                                          |
| container_version  | 1.9.1                                                      |
| discovery_url      | https://discovery.etcd.io/3b7fb09733429d16679484673ba3bfd5 |
| name               | k8s-cluster                                                |
+--------------------+------------------------------------------------------------+
```

5.36 Kubernetes

Kubernetes uses a range of terminology that we refer to in this guide. We define these common terms for your reference:

Pod

When using the Kubernetes container orchestration engine, a pod is the smallest deployable unit that can be created and managed. A pod is a co-located group of application containers that run with a shared context. When using Magnum, pods are created and managed within clusters. Refer to the pods section (http://kubernetes.io/v1.0/docs/user-guide/pods.html) in the Kubernetes User Guide (http://kubernetes.io/v1.0/docs/user-guide/) for more information.

Replication controller

A replication controller is used to ensure that at any given time a certain number of replicas of a pod are running. Pods are automatically created and deleted by the replication controller as necessary based on a template to ensure that the defined number of replicas exist. Refer to the replication controller section (http://kubernetes.io/v1.0/docs/user-guide/replication-controller.html) in the Kubernetes User Guide (http://kubernetes.io/v1.0/docs/user-guide/) for more information.

Service

A service is an additional layer of abstraction provided by the Kubernetes container orchestration engine which defines a logical set of pods and a policy for accessing them. This is useful because pods are created and deleted by a replication controller, for example, other pods needing to discover them can do so via the service abstraction. Refer to the services section (http://kubernetes.io/v1.0/docs/user-guide/services.html) in the Kubernetes User Guide (http://kubernetes.io/v1.0/docs/user-guide/) for more information.

When Magnum deploys a Kubernetes cluster, it uses parameters defined in the ClusterTemplate and specified on the cluster-create command, for example:

```
magnum cluster-template-create k8s-cluster-template \
                           --image fedora-atomic-latest \
                           --keypair testkey \
                           --external-network public \
                           --dns-nameserver 8.8.8.8 \
                           --flavor m1.small \
                           --docker-volume-size 5 \
                           --network-driver flannel \
                           --coe kubernetes
```

```
magnum cluster-create k8s-cluster \
                      --cluster-template k8s-cluster-template \
                      --master-count 3 \
                      --node-count 8
```

Following are further details relevant to a Kubernetes cluster:

Number of masters (master-count)

Specified in the cluster-create command to indicate how many servers will run as master in the cluster. Having more than one will provide high availability. The masters will be in a load balancer pool and the virtual IP address (VIP) of the load balancer will serve as the Kubernetes API endpoint. For external access, a floating IP associated with this VIP is available and this is the endpoint shown for Kubernetes in the 'cluster-show' command.

Number of nodes (node-count)

Specified in the cluster-create command to indicate how many servers will run as node in the cluster to host the users' pods. The nodes are registered in Kubernetes using the Nova instance name.

Network driver (network-driver)

Specified in the ClusterTemplate to select the network driver. The supported and default network driver is 'flannel', an overlay network providing a flat network for all pods.

Volume driver (volume-driver)

Specified in the ClusterTemplate to select the volume driver. The supported volume driver is 'cinder', allowing Cinder volumes to be mounted in containers for use as persistent storage. Data written to these volumes will persist after the container exits and can be accessed again from other containers, while data written to the union file system hosting the container will be deleted.

Storage driver (docker-storage-driver)

Specified in the ClusterTemplate to select the Docker storage driver. The supported storage drivers are 'devicemapper' and 'overlay', with 'devicemapper' being the default.

Image (image)

Specified in the ClusterTemplate to indicate the image to boot the servers. The image binary is loaded in Glance with the attribute 'os_distro = fedora-atomic'. Current supported images are Fedora Atomic (download from Fedora (https://alt.fedoraproject.org/pub/alt/atomic/stable/Cloud-Images/x86_64/Images)) and CoreOS (download from CoreOS (http://beta.release.core-os.net/amd64-usr/current/coreos_production_openstack_image.img.bz2))

TLS (tls-disabled)

Transport Layer Security is enabled by default, so you need a key and signed certificate to access the Kubernetes API and CLI. Magnum handles its own key and certificate when interfacing with the Kubernetes cluster. In development mode, TLS can be disabled. Refer to the 'Transport Layer Security'_ section for more details.

What runs on the servers

The servers for Kubernetes master host containers in the 'kube-system' name space to run the Kubernetes proxy, scheduler and controller manager. The masters will not host users' pods. Kubernetes API server, docker daemon, etcd and flannel run as systemd services. The servers for Kubernetes node also host a container in the 'kube-system' name space to run the Kubernetes proxy, while Kubernetes kubelet, docker daemon and flannel run as systemd services.

Log into the servers

You can log into the master servers using the login 'fedora' and the keypair specified in the ClusterTemplate.

In addition to the common attributes in the ClusterTemplate, you can specify the following attributes that are specific to Kubernetes by using the labels attribute.

This label corresponds to Kubernetes parameter for the API server '–admission-control'. For more details, refer to the Admission Controllers (https://kubernetes.io/docs/admin/admission-controllers//). The default value corresponds to the one recommended in this doc for our current Kubernetes version.

This label sets the size of a volume holding the etcd storage data. The default value is 0, meaning the etcd data is not persisted (no volume).

This label allows users to select a specific Kubernetes release, based on its container tag (https://hub.docker.com/r/openstackmagnum/kubernetes-apiserver/tags/). If unset, the current Magnum version's default Kubernetes release is installed.

This label triggers the deployment of the kubernetes dashboard. The default value is 1, meaning it will be enabled.

5.36.1 External load balancer for services

All Kubernetes pods and services created in the cluster are assigned IP addresses on a private container network so they can access each other and the external internet. However, these IP addresses are not accessible from an external network.

To publish a service endpoint externally so that the service can be accessed from the external network, Kubernetes provides the external load balancer feature. This is done by simply specifying in the service manifest the attribute “type: LoadBalancer”. Magnum enables and configures the Kubernetes plugin for OpenStack so that it can interface with Neutron and manage the necessary networking resources.

When the service is created, Kubernetes will add an external load balancer in front of the service so that the service will have an external IP address in addition to the internal IP address on the container network. The service endpoint can then be accessed with this external IP address. Kubernetes handles all the life cycle operations when pods are modified behind the service and when the service is deleted.

Refer to the document Kubernetes external load balancer (https://github.com/openstack/magnum/blob/master/doc/source/dev/kubernetes-load-balancer.rst) for more details.

5.37 Swarm

A Swarm cluster is a pool of servers running Docker daemon that is managed as a single Docker host. One or more Swarm managers accepts the standard Docker API and manage this pool of servers. Magnum deploys a Swarm cluster using parameters defined in the ClusterTemplate and specified on the ‘cluster-create’ command, for example:

```
magnum cluster-template-create swarm-cluster-template \
                           --image fedora-atomic-latest \
                           --keypair testkey \
                           --external-network public \
                           --dns-nameserver 8.8.8.8 \
                           --flavor m1.small \
                           --docker-volume-size 5 \
                           --coe swarm

magnum cluster-create swarm-cluster \
                   --cluster-template swarm-cluster-template \
                   --master-count 3 \
                   --node-count 8
```

Following are further details relevant to Swarm:

What runs on the servers

There are two types of servers in the Swarm cluster: managers and nodes. The Docker daemon runs on all servers. On the servers for manager, the Swarm manager is run as a Docker container on port 2376 and this is initiated by the systemd service swarm-manager. Etcd is also run on the manager servers for discovery of the node servers in the cluster. On the servers for node, the Swarm agent is run as a Docker container on port 2375 and this is initiated by the systemd service swarm-agent. On start up, the agents will register themselves in etcd and the managers will discover the new node to manage.

Number of managers (master-count)

Specified in the cluster-create command to indicate how many servers will run as managers in the cluster. Having more than one will provide high availability. The managers will be in a load balancer pool and the load balancer virtual IP address (VIP) will serve as the Swarm API endpoint. A floating IP associated with the load balancer VIP will serve as the external Swarm API endpoint. The managers accept the standard Docker API and perform the corresponding operation on the servers in the pool. For instance, when a new container is created, the managers will select one of the servers based on some strategy and schedule the containers there.

Number of nodes (node-count)

Specified in the cluster-create command to indicate how many servers will run as nodes in the cluster to host your Docker containers. These servers will register themselves in etcd for discovery by the managers, and interact with the managers. Docker daemon is run locally to host containers from users.

Network driver (network-driver)

Specified in the ClusterTemplate to select the network driver. The supported drivers are ‘docker’ and ‘flannel’, with ‘docker’ as the default. With the ‘docker’ driver, containers are connected to the ‘docker0’ bridge on each node and are assigned local IP address. With the ‘flannel’ driver, containers are connected to a flat overlay network and are assigned IP address by Flannel.

Volume driver (volume-driver)

Specified in the ClusterTemplate to select the volume driver to provide persistent storage for containers. The supported volume driver is 'rexray'. The default is no volume driver. When 'rexray' or other volume driver is deployed, you can use the Docker 'volume' command to create, mount, unmount, delete volumes in containers. Cinder block storage is used as the backend to support this feature.

Storage driver (docker-storage-driver)

Specified in the ClusterTemplate to select the Docker storage driver. The supported storage driver are 'devicemapper' and 'overlay', with 'devicemapper' being the default.

Image (image)

Specified in the ClusterTemplate to indicate the image to boot the servers for the Swarm manager and node. The image binary is loaded in Glance with the attribute 'os_distro = fedora-atomic'. Current supported image is Fedora Atomic (download from Fedora (https://alt.fedoraproject.org/pub/alt/atomic/stable/Cloud-Images/x86_64/Images))

TLS (tls-disabled)

Transport Layer Security is enabled by default to secure the Swarm API for access by both the users and Magnum. You will need a key and a signed certificate to access the Swarm API and CLI. Magnum handles its own key and certificate when interfacing with the Swarm cluster. In development mode, TLS can be disabled. Refer to the 'Transport Layer Security'_ section for details on how to create your key and have Magnum sign your certificate.

Log into the servers

You can log into the manager and node servers with the account 'fedora' and the keypair specified in the ClusterTemplate.

In addition to the common attributes in the ClusterTemplate, you can specify the following attributes that are specific to Swarm by using the labels attribute.

This label corresponds to Swarm parameter for master '–strategy'. For more details, refer to the Swarm Strategy (https://docs.docker.com/swarm/scheduler/strategy/). Valid values for this label are:

- spread
- binpack
- random

5.38 Mesos

A Mesos cluster consists of a pool of servers running as Mesos slaves, managed by a set of servers running as Mesos masters. Mesos manages the resources from the slaves but does not itself deploy containers. Instead, one of more Mesos frameworks running on the Mesos cluster would accept user requests on their own endpoint, using their particular API. These frameworks would then negotiate the resources with Mesos and the containers are deployed on the servers where the resources are offered.

Magnum deploys a Mesos cluster using parameters defined in the ClusterTemplate and specified on the 'cluster-create' command, for example:

```
magnum cluster-template-create mesos-cluster-template \
                        --image ubuntu-mesos \
                        --keypair testkey \
                        --external-network public \
                        --dns-nameserver 8.8.8.8 \
                        --flavor m1.small \
                        --coe mesos

magnum cluster-create mesos-cluster \
                    --cluster-template mesos-cluster-template \
                    --master-count 3 \
                    --node-count 8
```

Following are further details relevant to Mesos:

What runs on the servers

There are two types of servers in the Mesos cluster: masters and slaves. The Docker daemon runs on all servers. On the servers for master, the Mesos master is run as a process on port 5050 and this is initiated by the upstart service 'mesos-master'. Zookeeper is also run on the master servers, initiated by the upstart service 'zookeeper'. Zookeeper is used by the master servers for electing the leader among the masters, and by the slave servers and frameworks to determine the current leader. The framework Marathon is run as a process on port 8080 on the master servers, initiated by the upstart service 'marathon'. On the servers for slave, the Mesos slave is run as a process initiated by the upstart service 'mesos-slave'.

Number of master (master-count)

Specified in the cluster-create command to indicate how many servers will run as masters in the cluster. Having more than one will provide high availability. If the load balancer option is specified, the masters will be in a load balancer pool and the load balancer virtual IP address (VIP) will serve as the Mesos API endpoint. A floating IP associated with the load balancer VIP will serve as the external Mesos API endpoint.

Number of agents (node-count)

Specified in the cluster-create command to indicate how many servers will run as Mesos slave in the cluster. Docker daemon is run locally to host containers from users. The slaves report their available resources to the master and accept request from the master to deploy tasks from the frameworks. In this case, the tasks will be to run Docker containers.

Network driver (network-driver)

Specified in the ClusterTemplate to select the network driver. Currently 'docker' is the only supported driver: containers are connected to the 'docker0' bridge on each node and are assigned local IP address.

Volume driver (volume-driver)

Specified in the ClusterTemplate to select the volume driver to provide persistent storage for containers. The supported volume driver is 'rexray'. The default is no volume driver. When 'rexray' or other volume driver is deployed, you can use the Docker 'volume' command to create, mount, unmount, delete volumes in containers. Cinder block storage is used as the backend to support this feature.

Storage driver (docker-storage-driver)

This is currently not supported for Mesos.

Image (image)

Specified in the ClusterTemplate to indicate the image to boot the servers for the Mesos master and slave. The image binary is loaded in Glance with the attribute 'os_distro = ubuntu'. You can download the ready-built image (https://fedorapeople.org/groups/magnum/ubuntu-mesos-latest.qcow2).

TLS (tls-disabled)

Transport Layer Security is currently not implemented yet for Mesos.

Log into the servers

You can log into the manager and node servers with the account 'ubuntu' and the keypair specified in the ClusterTemplate.

In addition to the common attributes in the baymodel, you can specify the following attributes that are specific to Mesos by using the labels attribute.

When the volume driver 'rexray' is used, you can mount a data volume backed by Cinder to a host to be accessed by a container. In this case, the label 'rexray_preempt' can optionally be set to True or False to enable any host to take control of the volume regardless of whether other hosts are using the volume. This will in effect unmount the volume from the current host and remount it on the new host. If this label is set to false, then rexray will ensure data safety for locking the volume before remounting. The default value is False.

This label corresponds to the Mesos parameter for slave '–isolation'. The isolators are needed to provide proper isolation according to the runtime configurations specified in the container image. For more details, refer to the Mesos configuration (http://mesos.apache.org/documentation/latest/configuration/) and the Mesos container image support (http://mesos.apache.org/documentation/latest/container-image/). Valid values for this label are:

- filesystem/posix
- filesystem/linux
- filesystem/shared
- posix/cpu
- posix/mem
- posix/disk
- cgroups/cpu
- cgroups/mem
- docker/runtime
- namespaces/pid

This label corresponds to the Mesos parameter for agent '–image_providers', which tells Mesos containerizer what types of container images are allowed. For more details, refer to the Mesos configuration (http://mesos.apache.org/documentation/latest/configuration/) and the Mesos container image support (http://mesos.apache.org/documentation/latest/container-image/). Valid values are:

- appc
- docker
- appc,docker

This label corresponds to the Mesos parameter '–work_dir' for slave. For more details, refer to the Mesos configuration (http://mesos.apache.org/documentation/latest/configuration/). Valid value is a directory path to use as the work directory for the framework, for example:

```
mesos_slave_work_dir=/tmp/mesos
```

This label corresponds to the Mesos parameter for slave '–executor_environment_variables', which passes additional environment variables to the executor and subsequent tasks. For more details, refer to the Mesos configuration (http://mesos.apache.org/documentation/latest/configuration/). Valid value is the name of a JSON file, for example:

```
mesos_slave_executor_env_variables=/home/ubuntu/test.json
```

The JSON file should contain environment variables, for example:

```
{
   "PATH": "/bin:/usr/bin",
   "LD_LIBRARY_PATH": "/usr/local/lib"
}
```

By default the executor will inherit the slave's environment variables.

5.38.1 Building Mesos image

The boot image for Mesos cluster is an Ubuntu 14.04 base image with the following middleware pre-installed:

- `docker`
- `zookeeper`

- `mesos`
- `marathon`

The cluster driver provides two ways to create this image, as follows.

5.38.1.1 Diskimage-builder

To run the diskimage-builder (http://docs.openstack.org/developer/diskimage-builder) tool manually, use the provided elements (http://git.openstack.org/cgit/openstack/magnum/tree/magnum/drivers/mesos_ubuntu_v1/image/mesos/). Following are the typical steps to use the diskimage-builder tool on an Ubuntu server:

```
$ sudo apt-get update
$ sudo apt-get install git qemu-utils python-pip
$ sudo pip install diskimage-builder

$ git clone https://git.openstack.org/openstack/magnum
$ git clone https://git.openstack.org/openstack/dib-utils.git
$ git clone https://git.openstack.org/openstack/tripleo-image-elements.git
$ git clone https://git.openstack.org/openstack/heat-templates.git
$ export PATH="${PWD}/dib-utils/bin:$PATH"
$ export ELEMENTS_PATH=tripleo-image-elements/elements:heat-templates/hot/software-
config/elements:magnum/magnum/drivers/mesos_ubuntu_v1/image/mesos
$ export DIB_RELEASE=trusty

$ disk-image-create ubuntu vm docker mesos \
    os-collect-config os-refresh-config os-apply-config \
    heat-config heat-config-script \
    -o ubuntu-mesos.qcow2
```

5.38.1.2 Dockerfile

To build the image as above but within a Docker container, use the provided Dockerfile (http://git.openstack.org/cgit/openstack/magnum/tree/magnum/drivers/mesos_ubuntu_v1/image/Dockerfile). The output image will be saved as '/tmp/ubuntu-mesos.qcow2'. Following are the typical steps to run a Docker container to build the image:

```
$ git clone https://git.openstack.org/openstack/magnum
$ cd magnum/magnum/drivers/mesos_ubuntu_v1/image
$ sudo docker build -t magnum/mesos-builder .
```

```
$ sudo docker run -v /tmp:/output --rm -ti --privileged magnum/mesos-builder
...
Image file /output/ubuntu-mesos.qcow2 created...
```

5.38.2 Using Marathon

Marathon is a Mesos framework for long running applications. Docker containers can be deployed via Marathon's REST API. To get the endpoint for Marathon, run the cluster-show command and look for the property 'api_address'. Marathon's endpoint is port 8080 on this IP address, so the web console can be accessed at:

```
http://<api_address>:8080/
```

Refer to Marathon documentation for details on running applications. For example, you can 'post' a JSON app description to `http://<api_address>:8080/apps` to deploy a Docker container:

```
$ cat > app.json << END
{
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "libmesos/ubuntu"
    }
  },
  "id": "ubuntu",
  "instances": 1,
  "cpus": 0.5,
  "mem": 512,
  "uris": [],
  "cmd": "while sleep 10; do date -u +%T; done"
}
END
$ API_ADDRESS=$(magnum cluster-show mesos-cluster | awk '/ api_address /{print $4}')
$ curl -X POST -H "Content-Type: application/json" \
    http://${API_ADDRESS}:8080/v2/apps -d@app.json
```

5.39 Transport Layer Security

Magnum uses TLS to secure communication between a cluster's services and the outside world. TLS is a complex subject, and many guides on it exist already. This guide will not attempt to fully describe TLS, but instead will only cover the necessary steps to get a client set up to talk to a cluster with TLS. A more in-depth guide on TLS can be found in the OpenSSL Cookbook (https://www.feistyduck.com/books/openssl-cookbook/) by Ivan Ristić.

TLS is employed at 3 points in a cluster:

1. By Magnum to communicate with the cluster API endpoint
2. By the cluster worker nodes to communicate with the master nodes
3. By the end-user when they use the native client libraries to interact with the cluster. This applies to both a CLI or a program that uses a client for the particular cluster. Each client needs a valid certificate to authenticate and communicate with a cluster.

The first two cases are implemented internally by Magnum and are not exposed to the users, while the last case involves the users and is described in more details below.

5.39.1 Deploying a secure cluster

Current TLS support is summarized below:

C	TLS support
Kubernetes	yes
Swarm	yes
Mesos	no

For cluster type with TLS support, e.g. Kubernetes and Swarm, TLS is enabled by default. To disable TLS in Magnum, you can specify the parameter '–tls-disabled' in the ClusterTemplate. Please note it is not recommended to disable TLS due to security reasons.

In the following example, Kubernetes is used to illustrate a secure cluster, but the steps are similar for other cluster types that have TLS support.

First, create a ClusterTemplate; by default TLS is enabled in Magnum, therefore it does not need to be specified via a parameter:

```
magnum cluster-template-create secure-kubernetes \
                           --keypair default \
                           --external-network public \
                           --image fedora-atomic-latest \
                           --dns-nameserver 8.8.8.8 \
                           --flavor m1.small \
                           --docker-volume-size 3 \
                           --coe kubernetes \
                           --network-driver flannel

+-----------------------+--------------------------------------+
| Property              | Value                                |
+-----------------------+--------------------------------------+
| insecure_registry     | None                                 |
| http_proxy            | None                                 |
| updated_at            | None                                 |
| master_flavor_id      | None                                 |
| uuid                  | 5519b24a-621c-413c-832f-c30424528b31 |
| no_proxy              | None                                 |
| https_proxy           | None                                 |
| tls_disabled          | False                                |
| keypair_id            | time4funkey                          |
| public                | False                                |
| labels                | {}                                   |
| docker_volume_size    | 5                                    |
| server_type           | vm                                   |
| external_network_id   | public                               |
| cluster_distro        | fedora-atomic                        |
| image_id              | fedora-atomic-latest                 |
| volume_driver         | None                                 |
| registry_enabled      | False                                |
| docker_storage_driver | devicemapper                         |
| apiserver_port        | None                                 |
| name                  | secure-kubernetes                    |
| created_at            | 2016-07-25T23:09:50+00:00            |
| network_driver        | flannel                              |
| fixed_network         | None                                 |
| coe                   | kubernetes                           |
| flavor_id             | m1.small                             |
| dns_nameserver        | 8.8.8.8                              |
```

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

Now create a cluster. Use the ClusterTemplate name as a template for cluster creation:

```
magnum cluster-create secure-k8s-cluster \
                      --cluster-template secure-kubernetes \
                      --node-count 1

+--------------------+--------------------------------------------------------------+
| Property           | Value                                                        |
+--------------------+--------------------------------------------------------------+
| status             | CREATE_IN_PROGRESS                                           |
| uuid               | 3968ffd5-678d-4555-9737-35f191340fda                         |
| stack_id           | c96b66dd-2109-4ae2-b510-b3428f1e8761                         |
| status_reason      | None                                                         |
| created_at         | 2016-07-25T23:14:06+00:00                                    |
| updated_at         | None                                                         |
| create_timeout     | 0                                                            |
| api_address        | None                                                         |
| coe_version        | -                                                            |
| cluster_template_id| 5519b24a-621c-413c-832f-c30424528b31                         |
| master_addresses   | None                                                         |
| node_count         | 1                                                            |
| node_addresses     | None                                                         |
| master_count       | 1                                                            |
| container_version  | -                                                            |
| discovery_url      | https://discovery.etcd.io/ba52a8178e7364d43a323ee4387cf28e |
| name               | secure-k8s-cluster                                            |
+--------------------+--------------------------------------------------------------+
```

Now run cluster-show command to get the details of the cluster and verify that the api_address is ‘https’:

```
magnum cluster-show secure-k8scluster
+--------------------+--------------------------------------------------------------+
| Property           | Value                                                        |
+--------------------+--------------------------------------------------------------+
| status             | CREATE_COMPLETE                                              |
| uuid               | 04952c60-a338-437f-a7e7-d016d1d00e65                         |
| stack_id           | b7bf72ce-b08e-4768-8201-e63a99346898                         |
| status_reason      | Stack CREATE completed successfully                          |
| created_at         | 2016-07-25T23:14:06+00:00                                    |
| updated_at         | 2016-07-25T23:14:10+00:00                                    |
| create_timeout     | 60                                                           |
| coe_version        | v1.2.0                                                       |
| api_address        | https://192.168.19.86:6443                                   |
| cluster_template_id| da2825a0-6d09-4208-b39e-b2db666f1118                         |
```

```
| master_addresses   | ['192.168.19.87']                                              |
| node_count         | 1                                                              |
| node_addresses     | ['192.168.19.88']                                              |
| master_count       | 1                                                              |
| container_version  | 1.9.1                                                          |
| discovery_url      | https://discovery.etcd.io/3b7fb09733429d16679484673ba3bfd5 |
| name               | secure-k8s-cluster                                              |
+--------------------+----------------------------------------------------------------+
```

You can see the api_address contains https in the URL, showing that the Kubernetes services are configured securely with SSL certificates and now any communication to kube-apiserver will be over https.

5.39.2 Interfacing with a secure cluster

To communicate with the API endpoint of a secure cluster, you will need so supply 3 SSL artifacts:

1. Your client key
2. A certificate for your client key that has been signed by a Certificate Authority (CA)
3. The certificate of the CA

There are two ways to obtain these 3 artifacts.

5.39.2.1 Automated

Magnum provides the command 'cluster-config' to help the user in setting up the environment and artifacts for TLS, for example:

```
magnum cluster-config swarm-cluster --dir myclusterconfig
```

This will display the necessary environment variables, which you can add to your environment:

```
export DOCKER_HOST=tcp://172.24.4.5:2376
export DOCKER_CERT_PATH=myclusterconfig
export DOCKER_TLS_VERIFY=True
```

And the artifacts are placed in the directory specified:

```
ca.pem
```

```
cert.pem
key.pem
```

You can now use the native client to interact with the COE. The variables and artifacts are unique to the cluster.

The parameters for 'bay-config' are as follows:

–dir <dirname>
: Directory to save the certificate and config files.

`--force`
: Overwrite existing files in the directory specified.

5.39.2.2 Manual

You can create the key and certificates manually using the following steps.

Client Key
: Your personal private key is essentially a cryptographically generated string of bytes. It should be protected in the same manner as a password. To generate an RSA key, you can use the 'genrsa' command of the 'openssl' tool:

```
openssl genrsa -out key.pem 4096
```

This command generates a 4096 byte RSA key at key.pem.

Signed Certificate
: To authenticate your key, you need to have it signed by a CA. First generate the Certificate Signing Request (CSR). The CSR will be used by Magnum to generate a signed certificate that you will use to communicate with the cluster. To generate a CSR, openssl requires a config file that specifies a few values. Using the example template below, you can fill in the 'CN' value with your name and save it as client.conf:

```
$ cat > client.conf << END
[req]
distinguished_name = req_distinguished_name
req_extensions     = req_ext
prompt = no
[req_distinguished_name]
CN = Your Name
```

```
[req_ext]
extendedKeyUsage = clientAuth
END
```

Once you have client.conf, you can run the openssl 'req' command to generate the CSR:

```
openssl req -new -days 365 \
    -config client.conf \
    -key key.pem \
    -out client.csr
```

Now that you have your client CSR, you can use the Magnum CLI to send it off to Magnum to get it signed:

```
magnum ca-sign --cluster secure-k8s-cluster --csr client.csr > cert.pem
```

Certificate Authority

The final artifact you need to retrieve is the CA certificate for the cluster. This is used by your native client to ensure you are only communicating with hosts that Magnum set up:

```
magnum ca-show --cluster secure-k8s-cluster > ca.pem
```

Rotate Certificate

To rotate the CA certificate for a cluster and invalidate all user certificates, you can use the following command:

```
magnum ca-rotate --cluster secure-k8s-cluster
```

5.39.3 User Examples

Here are some examples for using the CLI on a secure Kubernetes and Swarm cluster. You can perform all the TLS set up automatically by:

```
eval $(magnum cluster-config <cluster-name>)
```

Or you can perform the manual steps as described above and specify the TLS options on the CLI. The SSL artifacts are assumed to be saved in local files as follows:

```
- key.pem: your SSL key
- cert.pem: signed certificate
- ca.pem: certificate for cluster CA
```

For Kubernetes, you need to get 'kubectl', a kubernetes CLI tool, to communicate with the cluster:

```
curl -O https://storage.googleapis.com/kubernetes-release/release/v1.2.0/bin/linux/amd64/
kubectl
chmod +x kubectl
sudo mv kubectl /usr/local/bin/kubectl
```

Now let's run some 'kubectl' commands to check the secure communication. If you used 'cluster-config', then you can simply run the 'kubectl' command without having to specify the TLS options since they have been defined in the environment:

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"0", GitVersion:"v1.2.0",
 GitCommit:"cffae0523cfa80ddf917aba69f08508b91f603d5", GitTreeState:"clean"}
Server Version: version.Info{Major:"1", Minor:"0", GitVersion:"v1.2.0",
 GitCommit:"cffae0523cfa80ddf917aba69f08508b91f603d5", GitTreeState:"clean"}
```

You can specify the TLS options manually as follows:

```
KUBERNETES_URL=$(magnum cluster-show secure-k8s-cluster |
                 awk '/ api_address /{print $4}')
kubectl version --certificate-authority=ca.pem \
                --client-key=key.pem \
                --client-certificate=cert.pem -s $KUBERNETES_URL

kubectl create -f redis-master.yaml --certificate-authority=ca.pem \
                                    --client-key=key.pem \
                                    --client-certificate=cert.pem -s $KUBERNETES_URL

pods/test2

kubectl get pods --certificate-authority=ca.pem \
                 --client-key=key.pem \
                 --client-certificate=cert.pem -s $KUBERNETES_URL
NAME           READY     STATUS    RESTARTS   AGE
redis-master   2/2       Running   0          1m
```

Beside using the environment variables, you can also configure 'kubectl' to remember the TLS options:

```
kubectl config set-cluster secure-k8s-cluster --server=${KUBERNETES_URL} \
    --certificate-authority=${PWD}/ca.pem
kubectl config set-credentials client --certificate-authority=${PWD}/ca.pem \
    --client-key=${PWD}/key.pem --client-certificate=${PWD}/cert.pem
kubectl config set-context secure-k8scluster --cluster=secure-k8scluster --user=client
kubectl config use-context secure-k8scluster
```

Then you can use 'kubectl' commands without the certificates:

```
kubectl get pods
NAME           READY     STATUS    RESTARTS   AGE
redis-master   2/2       Running   0          1m
```

Access to Kubernetes User Interface:

```
curl -L ${KUBERNETES_URL}/ui --cacert ca.pem --key key.pem \
    --cert cert.pem
```

You may also set up 'kubectl' proxy which will use your client certificates to allow you to browse to a local address to use the UI without installing a certificate in your browser:

```
kubectl proxy --api-prefix=/ --certificate-authority=ca.pem --client-key=key.pem \
              --client-certificate=cert.pem -s $KUBERNETES_URL
```

You can then open http://localhost:8001/ui in your browser.

The examples for Docker are similar. With 'cluster-config' set up, you can just run docker commands without TLS options. To specify the TLS options manually:

```
docker -H tcp://192.168.19.86:2376 --tlsverify \
       --tlscacert ca.pem \
       --tlskey key.pem \
       --tlscert cert.pem \
       info
```

5.39.4 Storing the certificates

Magnum generates and maintains a certificate for each cluster so that it can also communicate securely with the cluster. As a result, it is necessary to store the certificates in a secure manner. Magnum provides the following methods for storing the certificates and this is configured in /etc/magnum/magnum.conf in the section [certificates] with the parameter 'cert_manager_type'.

1. Barbican: Barbican is a service in OpenStack for storing secrets. It is used by Magnum to store the certificates when cert_manager_type is configured as:

   ```
   cert_manager_type = barbican
   ```

   This is the recommended configuration for a production environment. Magnum will interface with Barbican to store and retrieve certificates, delegating the task of securing the certificates to Barbican.

2. Magnum database: In some cases, a user may want an alternative to storing the certificates that does not require Barbican. This can be a development environment, or a private cloud that has been secured by other means. Magnum can store the certificates in its own database; this is done with the configuration:

   ```
   cert_manager_type = x509keypair
   ```

   This storage mode is only as secure as the controller server that hosts the database for the OpenStack services.

3. Local store: As another alternative that does not require Barbican, Magnum can simply store the certificates on the local host filesystem where the conductor is running, using the configuration:

   ```
   cert_manager_type = local
   ```

   Note that this mode is only supported when there is a single Magnum conductor running since the certificates are stored locally. The 'local' mode is not recommended for a production environment.

For the nodes, the certificates for communicating with the masters are stored locally and the nodes are assumed to be secured.

5.40 Networking

There are two components that make up the networking in a cluster.

1. The Neutron infrastructure for the cluster: this includes the private network, subnet, ports, routers, load balancers, etc.

2. The networking model presented to the containers: this is what the containers see in communicating with each other and to the external world. Typically this consists of a driver deployed on each node.

The two components are deployed and managed separately. The Neutron infrastructure is the integration with OpenStack; therefore, it is stable and more or less similar across different COE types. The networking model, on the other hand, is specific to the COE type and is still under active development in the various COE communities, for example, Docker libnetwork (https://github.com/docker/libnetwork) and Kubernetes Container Networking (https://github.com/ku-

bernetes/kubernetes/blob/release-1.1/docs/design/networking.md). As a result, the implementation for the networking models is evolving and new models are likely to be introduced in the future.

For the Neutron infrastructure, the following configuration can be set in the ClusterTemplate:

external-network

The external Neutron network ID to connect to this cluster. This is used to connect the cluster to the external internet, allowing the nodes in the cluster to access external URL for discovery, image download, etc. If not specified, the default value is "public" and this is valid for a typical devstack.

fixed-network

The Neutron network to use as the private network for the cluster nodes. If not specified, a new Neutron private network will be created.

dns-nameserver

The DNS nameserver to use for this cluster. This is an IP address for the server and it is used to configure the Neutron subnet of the cluster (dns_nameservers). If not specified, the default DNS is 8.8.8.8, the publicly available DNS.

http-proxy, https-proxy, no-proxy

The proxy for the nodes in the cluster, to be used when the cluster is behind a firewall and containers cannot access URL's on the external internet directly. For the parameter http-proxy and https-proxy, the value to provide is a URL and it will be set in the environment variable HTTP_PROXY and HTTPS_PROXY respectively in the nodes. For the parameter no-proxy, the value to provide is an IP or list of IP's separated by comma. Likewise, the value will be set in the environment variable NO_PROXY in the nodes.

For the networking model to the container, the following configuration can be set in the ClusterTemplate:

network-driver

The network driver name for instantiating container networks. Currently, the following network drivers are supported:

I v	K b n	S	Mesos
Flannel	supported	supported	supported
Docker	unsupported	supported	supported

If not specified, the default driver is Flannel for Kubernetes, and Docker for Swarm and Mesos.

Particular network driver may require its own set of parameters for configuration, and these parameters are specified through the labels in the ClusterTemplate. Labels are arbitrary key=value pairs.

When Flannel is specified as the network driver, the following optional labels can be added:

IPv4 network in CIDR format to use for the entire Flannel network. If not specified, the default is 10.100.0.0/16.

The size of the subnet allocated to each host. If not specified, the default is 24.

The type of backend for Flannel. Possible values are *udp, vxlan, host-gw*. If not specified, the default is *udp*. Selecting the best backend depends on your networking. Generally, *udp* is the most generally supported backend since there is little requirement on the network, but it typically offers the lowest performance. The *vxlan* backend performs better, but requires vxlan support in the kernel so the image used to provision the nodes needs to include this support. The *host-gw* backend offers the best performance since it does not actually encapsulate messages, but it requires all the nodes to be on the same L2 network. The

private Neutron network that Magnum creates does meet this requirement; therefore if the parameter *fixed_network* is not specified in the ClusterTemplate, *host-gw* is the best choice for the Flannel backend.

5.41 High Availability

To be filled in

5.42 Scaling

5.42.1 Performance tuning for periodic task

Magnum's periodic task performs a `stack-get` operation on the Heat stack underlying each of its clusters. If you have a large amount of clusters this can create considerable load on the Heat API. To reduce that load you can configure Magnum to perform one global `stack-list` per periodic task instead of one per cluster. This is disabled by default, both from the Heat and Magnum side since it causes a security issue, though: any user in any tenant holding the `admin` role can perform a global `stack-list` operation if Heat is configured to allow it for Magnum. If you want to enable it nonetheless, proceed as follows:

1. Set `periodic_global_stack_list` in magnum.conf to `True` (`False` by default).
2. Update heat policy to allow magnum list stacks. To this end, edit your heat policy file, usually etc/heat/policy.json``:

   ```
   ...
   stacks:global_index: "rule:context_is_admin",
   ```

   Now restart heat.

5.42.2 Containers and nodes

Scaling containers and nodes refers to increasing or decreasing allocated system resources. Scaling is a broad topic and involves many dimensions. In the context of Magnum in this guide, we consider the following issues:

- Scaling containers and scaling cluster nodes (infrastructure)
- Manual and automatic scaling

Since this is an active area of development, a complete solution covering all issues does not exist yet, but partial solutions are emerging.

Scaling containers involves managing the number of instances of the container by replicating or deleting instances. This can be used to respond to change in the workload being supported by the application; in this case, it is typically driven by certain metrics relevant to the application such as response time, etc. Other use cases include rolling upgrade, where a new version of a service can gradually be scaled up while the older version is gradually scaled down. Scaling containers is supported at the COE level and is specific to each COE as well as the version of the COE. You will need to refer to the documentation for the proper COE version for full details, but following are some pointers for reference.

For Kubernetes, pods are scaled manually by setting the count in the replication controller. Kubernetes version 1.3 and later also supports autoscaling (http://blog.kubernetes.io/2016/07/autoscaling-in-kubernetes.html). For Docker, the tool 'Docker Compose' provides the command docker-compose scale (https://docs.docker.com/compose/reference/scale/) which lets you manually set the number of instances of a container. For Swarm version 1.12 and later, services can also be scaled manually through the command docker service scale (https://docs.docker.com/engine/swarm/swarm-tutorial/scale-service/). Automatic scaling for Swarm is not yet available. Mesos manages the resources and does not support scaling directly; instead, this is provided by frameworks running within Mesos. With the Marathon framework currently supported in the Mesos cluster, you can use the scale operation (https://mesosphere.github.io/marathon/docs/application-basics.html) on the Marathon UI or through a REST API call to manually set the attribute 'instance' for a container.

Scaling the cluster nodes involves managing the number of nodes in the cluster by adding more nodes or removing nodes. There is no direct correlation between the number of nodes and the number of containers that can be hosted since the resources consumed (memory, CPU, etc) depend on the containers. However, if a certain resource is exhausted in the cluster, adding

more nodes would add more resources for hosting more containers. As part of the infrastructure management, Magnum supports manual scaling through the attribute 'node_count' in the cluster, so you can scale the cluster simply by changing this attribute:

```
magnum cluster-update mycluster replace node_count=2
```

Adding nodes to a cluster is straightforward: Magnum deploys additional VMs or baremetal servers through the heat templates and invokes the COE-specific mechanism for registering the new nodes to update the available resources in the cluster. Afterward, it is up to the COE or user to re-balance the workload by launching new container instances or re-launching dead instances on the new nodes.

Removing nodes from a cluster requires some more care to ensure continuous operation of the containers since the nodes being removed may be actively hosting some containers. Magnum performs a simple heuristic that is specific to the COE to find the best node candidates for removal, as follows:

Kubernetes

Magnum scans the pods in the namespace 'Default' to determine the nodes that are *not* hosting any (empty nodes). If the number of nodes to be removed is equal or less than the number of these empty nodes, these nodes will be removed from the cluster. If the number of nodes to be removed is larger than the number of empty nodes, a warning message will be sent to the Magnum log and the empty nodes along with additional nodes will be removed from the cluster. The additional nodes are selected randomly and the pods running on them will be deleted without warning. For this reason, a good practice is to manage the pods through the replication controller so that the deleted pods will be relaunched elsewhere in the cluster. Note also that even when only the empty nodes are removed, there is no guarantee that no pod will be deleted because there is no locking to ensure that Kubernetes will not launch new pods on these nodes after Magnum has scanned the pods.

Swarm

No node selection heuristic is currently supported. If you decrease the node_count, a node will be chosen by magnum without consideration of what containers are running on the selected node.

Mesos

Magnum scans the running tasks on Marathon server to determine the nodes on which there is *no* task running (empty nodes). If the number of nodes to be removed is equal or less than the number of these empty nodes, these nodes will be removed from the cluster.

If the number of nodes to be removed is larger than the number of empty nodes, a warning message will be sent to the Magnum log and the empty nodes along with additional nodes will be removed from the cluster. The additional nodes are selected randomly and the containers running on them will be deleted without warning. Note that even when only the empty nodes are removed, there is no guarantee that no container will be deleted because there is no locking to ensure that Mesos will not launch new containers on these nodes after Magnum has scanned the tasks.

Currently, scaling containers and scaling cluster nodes are handled separately, but in many use cases, there are interactions between the two operations. For instance, scaling up the containers may exhaust the available resources in the cluster, thereby requiring scaling up the cluster nodes as well. Many complex issues are involved in managing this interaction. A presentation at the OpenStack Tokyo Summit 2015 covered some of these issues along with some early proposals, Exploring Magnum and Senlin integration for autoscaling containers (https://www.openstack.org/summit/tokyo-2015/videos/presentation/exploring-magnum-and-senlin-integration-for-autoscaling-containers). This remains an active area of discussion and research.

5.43 Storage

Currently Cinder provides the block storage to the containers, and the storage is made available in two ways: as ephemeral storage and as persistent storage.

5.43.1 Ephemeral storage

The filesystem for the container consists of multiple layers from the image and a top layer that holds the modification made by the container. This top layer requires storage space and the storage is configured in the Docker daemon through a number of storage options. When the container is removed, the storage allocated to the particular container is also deleted.

Magnum can manage the containers' filesystem in two ways, storing them on the local disk of the compute instances or in a separate Cinder block volume for each node in the cluster, mounts it to the node and configures it to be used as ephemeral storage. Users can specify the size of the Cinder volume with the ClusterTemplate attribute 'docker-volume-size'. Currently the block size is fixed at cluster creation time, but future lifecycle operations may allow modifying the block size during the life of the cluster.

For drivers that support additional volumes for container storage, a label named 'docker_volume_type' is exposed so that users can select different cinder volume types for their volumes. The default volume *must* be set in 'default_docker_volume_type' in the 'cinder' section of magnum.conf, an obvious value is the default volume type set in cinder.conf of your cinder deployment . Please note, that docker_volume_type refers to a cinder volume type and it is unrelated to docker or kubernetes volumes.

Both local disk and the Cinder block storage can be used with a number of Docker storage drivers available.

- 'devicemapper': When used with a dedicated Cinder volume it is configured using direct-lvm and offers very good performance. If it's used with the compute instance's local disk uses a loopback device offering poor performance and it's not recommended for production environments. Using the 'devicemapper' driver does allow the use of SELinux.
- 'overlay' When used with a dedicated Cinder volume offers as good or better performance than devicemapper. If used on the local disk of the compute instance (especially with high IOPS drives) you can get significant performance gains. However, for kernel versions less than 4.9, SELinux must be disabled inside the containers resulting in worse container isolation, although it still runs in enforcing mode on the cluster compute instances.

5.43.2 Persistent storage

In some use cases, data read/written by a container needs to persist so that it can be accessed later. To persist the data, a Cinder volume with a filesystem on it can be mounted on a host and be made available to the container, then be unmounted when the container exits.

Docker provides the 'volume' feature for this purpose: the user invokes the 'volume create' command, specifying a particular volume driver to perform the actual work. Then this volume can be mounted when a container is created. A number of third-party volume drivers support OpenStack Cinder as the backend, for example Rexray and Flocker. Magnum currently supports Rexray as the volume driver for Swarm and Mesos. Other drivers are being considered.

Kubernetes allows a previously created Cinder block to be mounted to a pod and this is done by specifying the block ID in the pod YAML file. When the pod is scheduled on a node, Kubernetes will interface with Cinder to request the volume to be mounted on this node, then Kubernetes will launch the Docker container with the proper options to make the filesystem on the Cinder

volume accessible to the container in the pod. When the pod exits, Kubernetes will again send a request to Cinder to unmount the volume's filesystem, making it available to be mounted on other nodes.

Magnum supports these features to use Cinder as persistent storage using the ClusterTemplate attribute 'volume-driver' and the support matrix for the COE types is summarized as follows:

I v	K b n	S v	M e s o s
cinder	supported	supported	supported
rexray	supported	supported	supported

Following are some examples for using Cinder as persistent storage.

5.43.2.1 Using Cinder in Kubernetes

NOTE: This feature requires Kubernetes version 1.5.0 or above. The public Fedora image from Atomic currently meets this requirement.

1. Create the ClusterTemplate.
   Specify 'cinder' as the volume-driver for Kubernetes:

```
magnum cluster-template-create k8s-cluster-template \
                           --image fedora-23-atomic-7 \
                           --keypair testkey \
                           --external-network public \
                           --dns-nameserver 8.8.8.8 \
                           --flavor m1.small \
                           --docker-volume-size 5 \
                           --network-driver flannel \
```

```
                            --coe kubernetes \
                            --volume-driver cinder
```

2. Create the cluster:

```
magnum cluster-create k8s-cluster \
                      --cluster-template k8s-cluster-template \
                      --node-count 1
```

Kubernetes is now ready to use Cinder for persistent storage. Following is an example illustrating how Cinder is used in a pod.

1. Create the cinder volume:

```
cinder create --display-name=test-repo 1

XML:ID=$(cinder create --display-name=test-repo 1 | awk -F'|' '$2~/^[[:space:]]*id/
 {print $3}')
```

   The command will generate the volume with a ID. The volume ID will be specified in Step 2.

2. Create a pod in this cluster and mount this cinder volume to the pod. Create a file (e.g nginx-cinder.yaml) describing the pod:

```
cat > nginx-cinder.yaml << END
apiVersion: v1
kind: Pod
metadata:
  name: aws-web
spec:
  containers:
    - name: web
      image: nginx
      ports:
        - name: web
          containerPort: 80
          hostPort: 8081
          protocol: TCP
      volumeMounts:
        - name: html-volume
          mountPath: "/usr/share/nginx/html"
  volumes:
    - name: html-volume
      cinder:
        # Enter the volume ID below
```

```
        volumeID: $ID
        fsType: ext4
END
```

NOTE: The Cinder volume ID needs to be configured in the YAML file so the existing Cinder volume can be mounted in a pod by specifying the volume ID in the pod manifest as follows:

```
volumes:
- name: html-volume
  cinder:
    volumeID: $ID
    fsType: ext4
```

- Create the pod by the normal Kubernetes interface:

```
kubectl create -f nginx-cinder.yaml
```

You can start a shell in the container to check that the mountPath exists, and on an OpenStack client you can run the command 'cinder list' to verify that the cinder volume status is 'in-use'.

5.43.2.2 Using Cinder in Swarm

To be filled in

5.43.2.3 Using Cinder in Mesos

1. Create the ClusterTemplate.
   Specify 'rexray' as the volume-driver for Mesos. As an option, you can specify in a label the attributes 'rexray_preempt' to enable any host to take control of a volume regardless if other hosts are using the volume. If this is set to false, the driver will ensure data safety by locking the volume:

```
magnum cluster-template-create mesos-cluster-template \
                           --image ubuntu-mesos \
                           --keypair testkey \
                           --external-network public \
                           --dns-nameserver 8.8.8.8 \
                           --master-flavor m1.magnum \
                           --docker-volume-size 4 \
                           --tls-disabled \
```

```
                            --flavor m1.magnum \
                            --coe mesos \
                            --volume-driver rexray \
                            --labels rexray-preempt=true
```

2. Create the Mesos cluster:

```
magnum cluster-create mesos-cluster \
                      --cluster-template mesos-cluster-template \
                      --node-count 1
```

3. Create the cinder volume and configure this cluster:

```
cinder create --display-name=redisdata 1
```

   Create the following file

```
cat > mesos.json << END
{
  "id": "redis",
  "container": {
    "docker": {
    "image": "redis",
    "network": "BRIDGE",
    "portMappings": [
      { "containerPort": 80, "hostPort": 0, "protocol": "tcp"}
    ],
    "parameters": [
       { "key": "volume-driver", "value": "rexray" },
       { "key": "volume", "value": "redisdata:/data" }
    ]
    }
 },
 "cpus": 0.2,
 "mem": 32.0,
 "instances": 1
}
END
```

NOTE: When the Mesos cluster is created using this ClusterTemplate, the Mesos cluster will be configured so that a filesystem on an existing cinder volume can be mounted in a container by configuring the parameters to mount the cinder volume in the JSON file

```
"parameters": [
   { "key": "volume-driver", "value": "rexray" },
   { "key": "volume", "value": "redisdata:/data" }
```

```
]
```

- Create the container using Marathon REST API

```
MASTER_IP=$(magnum cluster-show mesos-cluster | awk '/ api_address /{print $4}')
curl -X POST -H "Content-Type: application/json" \
http://${MASTER_IP}:8080/v2/apps -d@mesos.json
```

You can log into the container to check that the mountPath exists, and you can run the command 'cinder list' to verify that your cinder volume status is 'in-use'.

5.44 Image Management

When a COE is deployed, an image from Glance is used to boot the nodes in the cluster and then the software will be configured and started on the nodes to bring up the full cluster. An image is based on a particular distro such as Fedora, Ubuntu, etc, and is prebuilt with the software specific to the COE such as Kubernetes, Swarm, Mesos. The image is tightly coupled with the following in Magnum:

1. Heat templates to orchestrate the configuration.
2. Template definition to map ClusterTemplate parameters to Heat template parameters.
3. Set of scripts to configure software.

Collectively, they constitute the driver for a particular COE and a particular distro; therefore, developing a new image needs to be done in conjunction with developing these other components. Image can be built by various methods such as diskimagebuilder, or in some case, a distro image can be used directly. A number of drivers and the associated images is supported in Magnum as reference implementation. In this section, we focus mainly on the supported images.

All images must include support for cloud-init and the heat software configuration utility:

- os-collect-config
- os-refresh-config
- os-apply-config
- heat-config
- heat-config-script

Additional software are described as follows.

5.44.1 Kubernetes on Fedora Atomic

This image can be downloaded from the public Atomic site (https://alt.fedoraproject.org/pub/alt/atomic/stable/Cloud-Images/x86_64/Images/) or can be built locally using diskimagebuilder. Details can be found in the fedora-atomic element (https://github.com/openstack/magnum/tree/master/magnum/elements/fedora-atomic) The image currently has the following OS/software:

Software	version
Fedora	26
Docker	1.13.1
Kubernetes	1.7.4
etcd	3.1.3
Flannel	0.7.0

The following software are managed as systemd services:

- kube-apiserver
- kubelet
- etcd
- flannel (if specified as network driver)
- docker

The following software are managed as Docker containers:

- kube-controller-manager
- kube-scheduler
- kube-proxy

The login for this image is *fedora*.

5.44.2 Kubernetes on CoreOS

CoreOS publishes a stock image (http://beta.release.core-os.net/amd64-usr/current/coreos_production_openstack_image.img.bz2) that is being used to deploy Kubernetes. This image has the following OS/software:

Software	Version
CoreOS	4.3.6
Docker	1.9.1
Kubernetes	1.0.6
etcd	2.2.3
Flannel	0.5.5

The following software are managed as systemd services:

- kubelet
- flannel (if specified as network driver)

- docker
- etcd

The following software are managed as Docker containers:

- kube-apiserver
- kube-controller-manager
- kube-scheduler
- kube-proxy

The login for this image is *core*.

5.44.3 Kubernetes on Ironic

This image is built manually using diskimagebuilder. The scripts and instructions are included in Magnum code repo (https://github.com/openstack/magnum/tree/master/magnum/templates/kubernetes/elements). Currently Ironic is not fully supported yet, therefore more details will be provided when this driver has been fully tested.

5.44.4 Swarm on Fedora Atomic

This image is the same as the image for Kubernetes on Fedora Atomic described above. The login for this image is *fedora*.

5.44.5 Mesos on Ubuntu

This image is built manually using diskimagebuilder. The Fedora site hosts the current image ubuntu-mesos-latest.qcow2 (https://fedorapeople.org/groups/magnum/ubuntu-mesos-latest.qcow2).

Os version	
Ubuntu	14.04
Docker	1.8.1
Mesos	0.25.0
Marathon	0.11.1

5.45 Notification

Magnum provides notifications about usage data so that 3rd party applications can use the data for auditing, billing, monitoring, or quota purposes. This document describes the current inclusions and exclusions for Magnum notifications.

Magnum uses Cloud Auditing Data Federation (CADF (http://www.dmtf.org/standards/cadf)) Notification as its notification format for better support of auditing, details about CADF are documented below.

5.45.1 Auditing with CADF

Magnum uses the PyCADF (http://docs.openstack.org/developer/pycadf) library to emit CADF notifications, these events adhere to the DMTF CADF (http://www.dmtf.org/standards/cadf) specification. This standard provides auditing capabilities for compliance with security, operational, and business processes and supports normalized and categorized event data for federation and aggregation.

Below table describes the event model components and semantics for each component:

model component	CADF Definition
OBSERVER	The RESOURCE that generates the CADF Event Record based on its observation (directly or indirectly) of the Actual Event.
INITIATOR	The RESOURCE that initiated, originated, or instigated the event's ACTION, accord-

model component	CADF Definition
	ing to the OBSERVER.
ACTION	The operation or activity the INITIATOR has performed, has attempted to perform or has pending against the event's TARGET, according to the OBSERVER.
TARGET	The RESOURCE against which

model component	CADF Definition
	the ACTION of a CADF Event Record was performed, attempted, or is pending, according to the OBSERVER.
OUTCOME	The result or status of the ACTION against the TARGET, according to the OBSERVER.

The `payload` portion of a CADF Notification is a CADF `event`, which is represented as a JSON dictionary. For example:

```
{
```

```
    "typeURI": "http://schemas.dmtf.org/cloud/audit/1.0/event",
    "initiator": {
        "typeURI": "service/security/account/user",
        "host": {
            "agent": "curl/7.22.0(x86_64-pc-linux-gnu)",
            "address": "127.0.0.1"
        },
        "id": "<initiator_id>"
    },
    "target": {
        "typeURI": "<target_uri>",
        "id": "openstack:1c2fc591-facb-4479-a327-520dade1ea15"
    },
    "observer": {
        "typeURI": "service/security",
        "id": "openstack:3d4a50a9-2b59-438b-bf19-c231f9c7625a"
    },
    "eventType": "activity",
    "eventTime": "2014-02-14T01:20:47.932842+00:00",
    "action": "<action>",
    "outcome": "success",
    "id": "openstack:f5352d7b-bee6-4c22-8213-450e7b646e9f",
}
```

Where the following are defined:

- `<initiator_id>`: ID of the user that performed the operation
- `<target_uri>`: CADF specific target URI, (i.e.: data/security/project)
- `<action>`: The action being performed, typically: `<operation>`. `<resource_type>`

Additionally there may be extra keys present depending on the operation being performed, these will be discussed below.

Note, the `eventType` property of the CADF payload is different from the `event_type` property of a notifications. The former (`eventType`) is a CADF keyword which designates the type of event that is being measured, this can be: `activity`, `monitor` or `control`. Whereas the latter (`event_type`) is described in previous sections as: `magnum.<resource_type>.<operation>`

5.45.2 Supported Events

The following table displays the corresponding relationship between resource types and operations. The bay type is deprecated and will be removed in a future version. Cluster is the new equivalent term.

resource	supported operations	typeURI
bay	create, update, delete	service/magnum/bay
cluster	create, update, delete	service/magnum/cluster

5.45.3 Example Notification - Cluster Create

The following is an example of a notification that is sent when a cluster is created. This example can be applied for any `create`, `update` or `delete` event that is seen in the table above. The `<action>` and `typeURI` fields will be change.

```
{
    "event_type": "magnum.cluster.created",
    "message_id": "0156ee79-b35f-4cef-ac37-d4a85f231c69",
    "payload": {
        "typeURI": "http://schemas.dmtf.org/cloud/audit/1.0/event",
        "initiator": {
            "typeURI": "service/security/account/user",
            "id": "c9f76d3c31e142af9291de2935bde98a",
            "user_id": "0156ee79-b35f-4cef-ac37-d4a85f231c69",
            "project_id": "3d4a50a9-2b59-438b-bf19-c231f9c7625a"
        },
        "target": {
            "typeURI": "service/magnum/cluster",
```

```
            "id": "openstack:1c2fc591-facb-4479-a327-520dade1ea15"
        },
        "observer": {
            "typeURI": "service/magnum/cluster",
            "id": "openstack:3d4a50a9-2b59-438b-bf19-c231f9c7625a"
        },
        "eventType": "activity",
        "eventTime": "2015-05-20T01:20:47.932842+00:00",
        "action": "create",
        "outcome": "success",
        "id": "openstack:f5352d7b-bee6-4c22-8213-450e7b646e9f",
        "resource_info": "671da331c47d4e29bb6ea1d270154ec3"
    }
    "priority": "INFO",
    "publisher_id": "magnum.host1234",
    "timestamp": "2016-05-20 15:03:45.960280"
}
```

5.46 Container Monitoring

The offered monitoring stack relies on the following set of containers and services:

- cAdvisor
- Node Exporter
- Prometheus
- Grafana

To setup this monitoring stack, users are given two configurable labels in the Magnum cluster template's definition:

This label accepts a boolean value. If *True*, the monitoring stack will be setup. By default *prometheus_monitoring = False*.

This label lets users create their own *admin* user password for the Grafana interface. It expects a string value. By default it is set to *admin*.

5.46.1 Container Monitoring in Kubernetes

By default, all Kubernetes clusters already contain *cAdvisor* integrated with the *Kubelet* binary. Its container monitoring data can be accessed on a node level basis through *http://NODE_IP:4194*.

Node Exporter is part of the above mentioned monitoring stack as it can be used to export machine metrics. Such functionality also work on a node level which means that when `prometheus-monitoring` is *True*, the Kubernetes nodes will be populated with an additional manifest under */etc/kubernetes/manifests*. Node Exporter is then automatically picked up and launched as a regular Kubernetes POD.

To aggregate and complement all the existing monitoring metrics and add a built-in visualization layer, Prometheus is used. It is launched by the Kubernetes master node(s) as a *Service* within a *Deployment* with one replica and it relies on a *ConfigMap* where the Prometheus configuration (prometheus.yml) is defined. This configuration uses Prometheus native support for service discovery in Kubernetes clusters, *kubernetes_sd_configs*. The respective manifests can be found in */srv/kubernetes/monitoring/* on the master nodes and once the service is up and running, Prometheus UI can be accessed through port 9090.

Finally, for custom plotting and enhanced metric aggregation and visualization, Prometheus can be integrated with Grafana as it provides native compliance for Prometheus data sources. Also Grafana is deployed as a *Service* within a *Deployment* with one replica. The default user is *admin* and the password is setup according to `grafana-admin-passwd`. There is also a default Grafana dashboard provided with this installation, from the official Grafana dashboards' repository (https://grafana.net/dashboards). The Prometheus data source is automatically added to Grafana once it is up and running, pointing to *http://prometheus:9090* through *Proxy*. The respective manifests can also be found in */srv/kubernetes/monitoring/* on the master nodes and once the service is running, the Grafana dashboards can be accessed through port 3000.

For both Prometheus and Grafana, there is an assigned *systemd* service called *kube-enable-monitoring*.

5.47 Kubernetes External Load Balancer

In a Kubernetes cluster, all masters and minions are connected to a private Neutron subnet, which in turn is connected by a router to the public network. This allows the nodes to access each other and the external internet.

All Kubernetes pods and services created in the cluster are connected to a private container network which by default is Flannel, an overlay network that runs on top of the Neutron private subnet. The pods and services are assigned IP addresses from this container network and they can access each other and the external internet. However, these IP addresses are not accessible from an external network.

To publish a service endpoint externally so that the service can be accessed from the external network, Kubernetes provides the external load balancer feature. This is done by simply specifying the attribute “type: LoadBalancer” in the service manifest. When the service is created, Kubernetes will add an external load balancer in front of the service so that the service will have an external IP address in addition to the internal IP address on the container network. The service endpoint can then be accessed with this external IP address. Refer to the Kubernetes service document (https://kubernetes.io/docs/concepts/services-networking/service/#type-loadbalancer) for more details.

A Kubernetes cluster deployed by Magnum will have all the necessary configuration required for the external load balancer. This document describes how to use this feature.

5.47.1 Steps for the cluster administrator

Because the Kubernetes master needs to interface with OpenStack to create and manage the Neutron load balancer, we need to provide a credential for Kubernetes to use.

In the current implementation, the cluster administrator needs to manually perform this step. We are looking into several ways to let Magnum automate this step in a secure manner. This means that after the Kubernetes cluster is initially deployed, the load balancer support is disabled. If the administrator does not want to enable this feature, no further action is required. All the services will be created normally; services that specify the load balancer will also be created successfully, but a load balancer will not be created.

Note that different versions of Kubernetes require different versions of Neutron LBaaS plugin running on the OpenStack instance:

```
==========================  ============================
Kubernetes Version on Master  Neutron LBaaS Version Required
==========================  ============================
1.2                         LBaaS v1
1.3 or later                LBaaS v2
==========================  ============================
```

Before enabling the Kubernetes load balancer feature, confirm that the OpenStack instance is running the required version of Neutron LBaaS plugin. To determine if your OpenStack instance is running LBaaS v1, try running the following command from your OpenStack control node:

```
neutron lb-pool-list
```

Or look for the following configuration in neutron.conf or neutron_lbaas.conf:

```
service_provider =
 LOADBALANCER:Haproxy:neutron_lbaas.services.loadbalancer.drivers.haproxy.plugin_driver.HaproxyOnHostPl
```

To determine if your OpenStack instance is running LBaaS v2, try running the following command from your OpenStack control node:

```
neutron lbaas-pool-list
```

Or look for the following configuration in neutron.conf or neutron_lbaas.conf:

```
service_plugins = neutron.plugins.services.agent_loadbalancer.plugin.LoadBalancerPluginv2
```

To configure LBaaS v1 or v2, refer to the Neutron documentation.

Before deleting the Kubernetes cluster, make sure to delete all the services that created load balancers. Because the Neutron objects created by Kubernetes are not managed by Heat, they will not be deleted by Heat and this will cause the cluster-delete operation to fail. If this occurs, delete the neutron objects manually (lb-pool, lb-vip, lb-member, lb-healthmonitor) and then run cluster-delete again.

5.47.2 Steps for the users

This feature requires the OpenStack cloud provider to be enabled. To do so, enable the cinder support (–volume-driver cinder).

For the user, publishing the service endpoint externally involves the following 2 steps:

1. Specify “type: LoadBalancer” in the service manifest
2. After the service is created, associate a floating IP with the VIP of the load balancer pool.

The following example illustrates how to create an external endpoint for a pod running nginx.

Create a file (e.g nginx.yaml) describing a pod running nginx:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
   app: nginx
spec:
```

```
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

Create a file (e.g nginx-service.yaml) describing a service for the nginx pod:

```
apiVersion: v1
kind: Service
metadata:
  name: nginxservice
  labels:
    app: nginx
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
  selector:
    app: nginx
  type: LoadBalancer
```

Please refer to the quickstart (https://docs.openstack.org/developer/magnum/userguide.html) guide on how to connect to Kubernetes running on the launched cluster. Assuming a Kubernetes cluster named k8sclusterv1 has been created, deploy the pod and service using following commands:

```
kubectl create -f nginx.yaml

kubectl create -f nginx-service.yaml
```

For more details on verifying the load balancer in OpenStack, refer to the following section on how it works.

Next, associate a floating IP to the load balancer. This can be done easily on Horizon by navigating to:

```
Compute -> Access & Security -> Floating IPs
```

Click on “Allocate IP To Project” and then on “Associate” for the new floating IP.

Alternatively, associating a floating IP can be done on the command line by allocating a floating IP, finding the port of the VIP, and associating the floating IP to the port. The commands shown below are for illustration purpose and assume that there is only one service with load balancer running in the cluster and no other load balancers exist except for those created for the cluster.

First create a floating IP on the public network:

```
neutron floatingip-create public

Created a new floatingip:

+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    |                                      |
| floating_ip_address | 172.24.4.78                          |
| floating_network_id | 4808eacb-e1a0-40aa-97b6-ecb745af2a4d |
| id                  | b170eb7a-41d0-4c00-9207-18ad1c30fecf |
| port_id             |                                      |
| router_id           |                                      |
| status              | DOWN                                 |
| tenant_id           | 012722667dc64de6bf161556f49b8a62     |
+---------------------+--------------------------------------+
```

Note the floating IP 172.24.4.78 that has been allocated. The ID for this floating IP is shown above, but it can also be queried by:

```
FLOATING_XML:ID=$(neutron floatingip-list | grep "172.24.4.78" | awk '{print $2}')
```

Next find the VIP for the load balancer:

```
VIP_XML:ID=$(neutron lb-vip-list | grep TCP | grep -v pool | awk '{print $2}')
```

Find the port for this VIP:

```
PORT_XML:ID=$(neutron lb-vip-show $VIP_ID | grep port_id | awk '{print $4}')
```

Finally associate the floating IP with the port of the VIP:

```
neutron floatingip-associate $FLOATING_ID $PORT_ID
```

The endpoint for nginx can now be accessed on a browser at this floating IP:

```
http://172.24.4.78:80
```

Alternatively, you can check for the nginx ‘welcome’ message by:

```
curl http://172.24.4.78:80
```

NOTE: it is not necessary to indicate port :80 here but it is shown to correlate with the port that was specified in the service manifest.

5.47.3 How it works

Kubernetes is designed to work with different Clouds such as Google Compute Engine (GCE), Amazon Web Services (AWS), and OpenStack; therefore, different load balancers need to be created on the particular Cloud for the services. This is done through a plugin for each Cloud and the OpenStack plugin was developed by Angus Lees:

```
https://github.com/kubernetes/kubernetes/blob/release-1.0/pkg/cloudprovider/openstack/
openstack.go
```

When the Kubernetes components kube-apiserver and kube-controller-manager start up, they will use the credential provided to authenticate a client to interface with OpenStack.

When a service with load balancer is created, the plugin code will interface with Neutron in this sequence:

1. Create lb-pool for the Kubernetes service
2. Create lb-member for the minions
3. Create lb-healthmonitor
4. Create lb-vip on the private network of the Kubernetes cluster

These Neutron objects can be verified as follows. For the load balancer pool:

```
neutron lb-pool-list
+--------------------------------------
+--------------------------------------------------+----------+-------------+----------
+----------------+--------+
| id                                   | name
 | provider | lb_method   | protocol | admin_state_up | status |
+--------------------------------------
+--------------------------------------------------+----------+-------------+----------
+----------------+--------+
| 241357b3-2a8f-442e-b534-bde7cd6ba7e4 | a1f03e40f634011e59c9efa163eae8ab
 | haproxy  | ROUND_ROBIN | TCP      | True           | ACTIVE |
| 82b39251-1455-4eb6-a81e-802b54c2df29 | k8sclusterv1-iypacicrskib-api_pool-fydshw7uvr7h
 | haproxy  | ROUND_ROBIN | HTTP     | True           | ACTIVE |
| e59ea983-c6e8-4cec-975d-89ade6b59e50 | k8sclusterv1-iypacicrskib-etcd_pool-qbpo43ew2m3x
 | haproxy  | ROUND_ROBIN | HTTP     | True           | ACTIVE |
+--------------------------------------
+--------------------------------------------------+----------+-------------+----------
+----------------+--------+
```

Note that 2 load balancers already exist to implement high availability for the cluster (api and ectd). The new load balancer for the Kubernetes service uses the TCP protocol and has a name assigned by Kubernetes.

For the members of the pool:

```
neutron lb-member-list
+--------------------------------------+----------+---------------+--------
+---------------+--------+
| id                                   | address  | protocol_port | weight |
 admin_state_up | status |
+--------------------------------------+----------+---------------+--------
+---------------+--------+
| 9ab7dcd7-6e10-4d9f-ba66-861f4d4d627c | 10.0.0.5 |          8080 |      1 | True
   | ACTIVE |
| b179c1ad-456d-44b2-bf83-9cdc127c2b27 | 10.0.0.5 |          2379 |      1 | True
   | ACTIVE |
| f222b60e-e4a9-4767-bc44-ffa66ec22afe | 10.0.0.6 |         31157 |      1 | True
   | ACTIVE |
+--------------------------------------+----------+---------------+--------
+---------------+--------+
```

Again, 2 members already exist for high availability and they serve the master node at 10.0.0.5. The new member serves the minion at 10.0.0.6, which hosts the Kubernetes service.

For the monitor of the pool:

```
neutron lb-healthmonitor-list
+--------------------------------------+------+----------------+
| id                                   | type | admin_state_up |
+--------------------------------------+------+----------------+
| 381d3d35-7912-40da-9dc9-b2322d5dda47 | TCP  | True           |
| 67f2ae8f-ffc6-4f86-ba5f-1a135f4af85c | TCP  | True           |
| d55ff0f3-9149-44e7-9b52-2e055c27d1d3 | TCP  | True           |
+--------------------------------------+------+----------------+
```

For the VIP of the pool:

```
neutron lb-vip-list
+--------------------------------------+----------------------------------+----------
+----------+---------------+--------+
| id                                   | name                             | address  |
 protocol | admin_state_up | status |
+--------------------------------------+----------------------------------+----------
+----------+---------------+--------+
| 9ae2ebfb-b409-4167-9583-4a3588d2ff42 | api_pool.vip                     | 10.0.0.3 |
 HTTP     | True           | ACTIVE |
```

```
| c318aec6-8b7b-485c-a419-1285a7561152 | a1f03e40f634011e59c9efa163eae8ab | 10.0.0.7 |
 TCP      | True           | ACTIVE |
| fc62cf40-46ad-47bd-aa1e-48339b95b011 | etcd_pool.vip                    | 10.0.0.4 |
 HTTP     | True           | ACTIVE |
+--------------------------------------+----------------------------------+----------
+----------+---------------+--------+
```

Note that the VIP is created on the private network of the cluster; therefore it has an internal IP address of 10.0.0.7. This address is also associated as the “external address” of the Kubernetes service. You can verify this in Kubernetes by running following command:

```
kubectl get services
NAME           LABELS                                    SELECTOR    IP(S)
 PORT(S)
kubernetes     component=apiserver,provider=kubernetes   <none>      10.254.0.1
 443/TCP
nginxservice   app=nginx                                 app=nginx   10.254.122.191   80/
TCP
                                                                     10.0.0.7
```

On GCE, the networking implementation gives the load balancer an external address automatically. On OpenStack, we need to take the additional step of associating a floating IP to the load balancer.

6 Nova User Guide

As an end user of nova, you'll use nova to create and manage servers with either tools or the API directly.

6.1 Tools for using Nova

- Horizon (https://docs.openstack.org/horizon/pike/user/launch-instances.html): The official web UI for the OpenStack Project.
- OpenStack Client (https://docs.openstack.org/python-openstackclient/pike/): The official CLI for OpenStack Projects. You should use this as your CLI for most OpenStack operations, as it includes commands for most of the projects in OpenStack.
- Nova Client (https://docs.openstack.org/python-novaclient/pike/user/shell.html): For some very advanced features (or administrative commands) of nova you may need to use nova client. It is still supported, but we recommend the `openstack` CLI.

6.2 Writing to the API

All end user (and some administrative) features of nova are exposed via a REST API, which can be used to build more complicated logic or automation with nova. This can be consumed directly, or via various SDKs. The following resources will help you get started with consuming the API directly.

- Compute API Guide: (https://developer.openstack.org/api-guide/compute/): The concept guide for the API. This helps lay out the concepts behind the API to make consuming the API reference easier.
- Compute API Reference (http://developer.openstack.org/api-ref/compute/): The complete reference for the API, including all methods, request, and response parameters and their meaning.
- The compute API evolves over time through Microversions (https://developer.openstack.org/api-guide/compute/microversions.html). This provides the history of all those changes. Consider it a "what's new" in the compute API.

- Block Device Mapping (https://docs.openstack.org/nova/pike/user/block-device-mapping.html): One of the trickier parts to understand is the Block Device Mapping parameters used to connect specific block devices to computes. This deserves its own deep dive.
- Store metadata on a configuration drive (https://docs.openstack.org/nova/pike/user/config-drive.html): Provide information to the guest instance when it is created.

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 (https://git.openstack.org/cgit/openstack/openstack-manuals) 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 (telemetry)*; 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 (barbican)*.

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 (telemetry)*; 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 (cinder)*.

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 (congress)*.

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.

Cross-Origin Resource Sharing (CORS)

A mechanism that allows many resources (for example, fonts, JavaScript) on a web page to be requested from another domain outside the domain from which the resource originated. In particular, JavaScript's AJAX calls can use the XMLHttpRequest mechanism.

Crowbar

An open source community project by Dell that aims to provide all necessary services to quickly deploy clouds.

current workload

An element of the Compute capacity cache that is calculated based on the number of build, snapshot, migrate, and resize operations currently in progress on a given host.

customer

Alternative term for 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 (designate)*.

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 (freezer)*.

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)*.

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 (telemetry)*; 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 high availability load balancer and proxy server 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)*.

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)*.

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.

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-tenant 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.

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 snap-

shots, 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-tenant 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 (ironic)*.

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.

Jenkins
: Tool used to run jobs automatically for OpenStack development.

jumbo frame
: Feature in modern Ethernet networks that supports frames up to approximately 9000 bytes.

Juno
: The code name for the tenth release of OpenStack. The design summit took place in Atlanta, Georgia, US and Juno is an unincorporated community in Georgia.

K

Kerberos
: A network authentication protocol which works on the basis of tickets. Kerberos allows nodes communication over a non-secure network, and allows nodes to prove their identity to one another in a secure manner.

kernel-based VM (KVM)
: An OpenStack-supported hypervisor. KVM is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V), ARM, IBM Power, and IBM zSeries. It consists of a loadable kernel module, that provides the core virtualization infrastructure and a processor specific module.

Key Manager service (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 (keystone)*.

Kickstart

A tool to automate system configuration and installation on Red Hat, Fedora, and CentOS-based Linux distributions.

Kilo

The code name for the eleventh release of OpenStack. The design summit took place in Paris, France. Due to delays in the name selection, the release was known only as K. Because `k` is the unit symbol for kilo and the reference artifact is stored near Paris in the Pavillon de Breteuil in Sèvres, the community chose Kilo as the release name.

L

large object

An object within Object Storage that is larger than 5 GB.

Launchpad

The collaboration site for OpenStack.

Layer-2 (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 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 rovide 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 *Container Infrastructure Management service (magnum)*.

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)*.

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 (mistral)*.

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 (monasca)*.

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-tenant, 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 tenants 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 (murano)*.

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)*.

neutron API

An alternative name for *Networking API (Neutron 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)*.

Nova API

Alternative term for the *Compute API (Nova 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 service (swift)*.

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 service (swift)* 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 (octavia)*.

Oldie

Term for an *Object Storage service (swift)* 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, and Queens. 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 (oslo)*.

P

panko

Part of the OpenStack *Telemetry service (telemetry)*; 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 eastermost stretch of Interstate 90.

Platform-as-a-Service (PaaS)

Provides to the consumer the ability to deploy applications through a programming language or tools supported by the cloud platform provider. An example of Platform-as-a-Service is an Eclipse/Java programming platform provided with no downloads required.

plug-in

Software component providing the actual implementation for Networking APIs, or for Compute APIs, depending on the context.

policy service

Component of Identity that provides a rule-management interface and a rule-based authorization engine.

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 (rally)*.

RAM filter

The Compute setting that enables or disables RAM overcommitment.

RAM overcommit

The ability to start new VM instances based on the actual memory usage of a host, as opposed to basing the decision on the amount of RAM each running instance thinks it has available. Also known as memory overcommit.

rate limit

Configurable option within Object Storage to limit database writes on a per-account and/or per-container basis.

raw

One of the VM image disk formats supported by Image service; an unstructured disk image.

rebalance

The process of distributing Object Storage partitions across all drives in the ring; used during initial ring creation and after ring reconfiguration.

reboot

Either a soft or hard reboot of a server. With a soft reboot, the operating system is signaled to restart, which enables a graceful shutdown of all processes. A hard reboot is the equivalent of power cycling the server. The virtualization platform should ensure that the reboot action has completed successfully, even in cases in which the underlying domain/VM is paused or halted/stopped.

rebuild

Removes all data on the server and replaces it with the specified image. Server ID and IP addresses remain the same.

Recon

An Object Storage component that collects meters.

record

Belongs to a particular domain and is used to specify information about the domain. There are several types of DNS records. Each record type contains particular information used to describe the purpose of that record. Examples include mail exchange (MX) records, which specify the mail server for a particular domain; and name server (NS) records, which specify the authoritative name servers for a domain.

record ID

A number within a database that is incremented each time a change is made. Used by Object Storage when replicating.

Red Hat Enterprise Linux (RHEL)

A Linux distribution that is compatible with OpenStack.

reference architecture

A recommended architecture for an OpenStack cloud.

region

A discrete OpenStack environment with dedicated API endpoints that typically shares only the Identity (keystone) with other regions.

registry

Alternative term for the Image service registry.

registry server

An Image service that provides VM image metadata information to clients.

Reliable, Autonomic Distributed Object Store

(RADOS)

A collection of components that provides object storage within Ceph. Similar to OpenStack Object Storage.

Remote Procedure Call (RPC)

The method used by the Compute RabbitMQ for intra-service communications.

replica

Provides data redundancy and fault tolerance by creating copies of Object Storage objects, accounts, and containers so that they are not lost when the underlying storage fails.

replica count

The number of replicas of the data in an Object Storage ring.

replication

The process of copying data to a separate physical device for fault tolerance and performance.

replicator

The Object Storage back-end process that creates and manages object replicas.

request ID

Unique ID assigned to each request sent to Compute.

rescue image

A special type of VM image that is booted when an instance is placed into rescue mode. Allows an administrator to mount the file systems for an instance to correct the problem.

resize

Converts an existing server to a different flavor, which scales the server up or down. The original server is saved to enable rollback if a problem occurs. All resizes must be tested and explicitly confirmed, at which time the original server is removed.

RESTful

A kind of web service API that uses REST, or Representational State Transfer. REST is the style of architecture for hypermedia systems that is used for the World Wide Web.

ring

An entity that maps Object Storage data to partitions. A separate ring exists for each service, such as account, object, and container.

ring builder

Builds and manages rings within Object Storage, assigns partitions to devices, and pushes the configuration to other storage nodes.

role

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 (sahara)*.

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 (senlin)*.

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 (manila)*. 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 (manila)* 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-tenant cloud environment, similar to how OpenStack provides block-based storage management through the OpenStack *Block Storage service (cinder)* 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 (solum)*.

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

Alternative term for a paused VM instance.

swap

Disk-based virtual memory used by operating systems to provide more memory than is actually available on the system.

swauth

An authentication and authorization service for Object Storage, implemented through WSGI middleware; uses Object Storage itself as the persistent backing store.

swift

Codename for OpenStack *Object Storage service (swift)*.

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 (tacker)*

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 (trove)*.

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 (RCA) service (Vitrage)*.

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 (watcher)*.

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

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 (zaqar)*.

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.