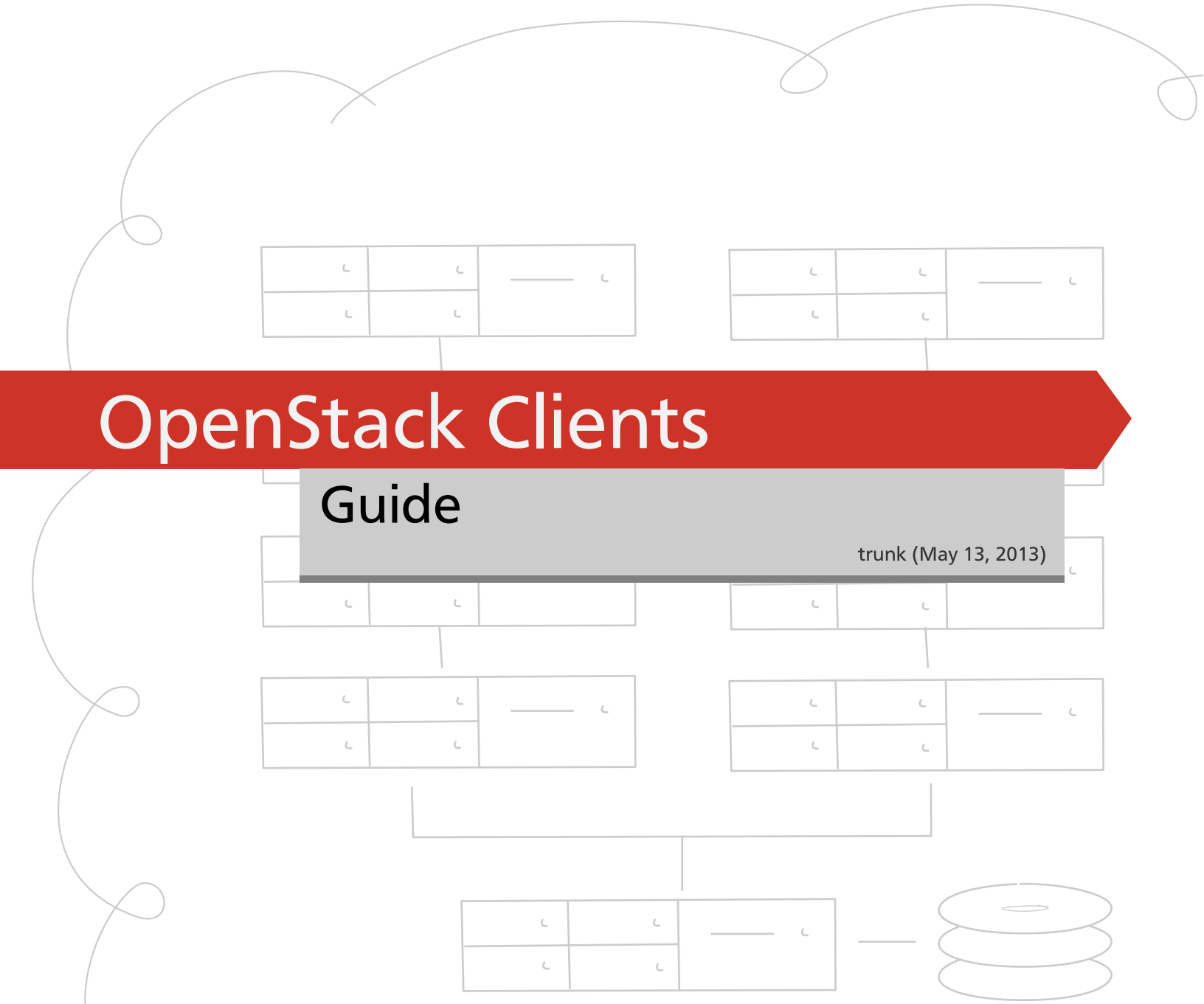
OpenStack Clients

Guide

trunk (May 13, 2013)



openstack™

docs.openstack.org

OpenStack Clients Guide

trunk (2013-05-13)

The OpenStack clients are command-line interfaces (CLIs) that let you run simple commands to make OpenStack API calls. These open-source Python clients are easy to learn and use. The OpenStack APIs are RESTful APIs that use all aspects of the HTTP protocol, including methods, URIs, media types, and response codes. To request OpenStack services, you must first issue an authentication request to the OpenStack Identity Service v2.0.

List of Tables

List of Examples

1. OpenStack Clients Overview

The OpenStack clients are command-line interfaces (CLIs) that let you run simple commands to make OpenStack API calls. These open-source Python clients are easy to learn and use.

The OpenStack APIs are RESTful APIs that use all aspects of the HTTP protocol, including methods, URIs, media types, and response codes. Internally, each client command runs cURL commands that embed API requests.

To install an OpenStack client, see Chapter 2, "Install the OpenStack Clients" [3].

To request OpenStack services either through the clients or through the APIs directly, you must first issue an authentication request to the OpenStack Identity Service v2.0. To do so, you can run the **credentials** command, which is a nova client command.

For example, to use the OpenStack Compute API from the command line, complete these steps:

1. Install the nova client.
2. Issue the nova **credentials** command.
3. Issue other nova client commands, such as **boot**, **list**, and so on.

Note

An OpenStack common client is in development.

To install the OpenStack clients on a Mac OS X or Linux system, use **pip** because it is easy and ensures that you get the latest version of the client from the Python Package Index. Also, **pip** lets you update or remove a package later on.

Use the following OpenStack clients to access the CLIs:

Table 1.1. OpenStack Clients

Client	Associated API	Description	See
cinder	Block storage	Create and delete volumes, attach volumes to and detach volumes from servers, create and delete snapshots, create volumes from snapshots, and get volume statistics.	Chapter 10, "cinder Client" [53]
glance	Image service	Manage images. For example, add and set permissions on images.	Chapter 7, "glance Client" [12]
keystone	Identity service	Create and manage users, tenants, roles, endpoints, and credentials.	Chapter 6, "keystone Client" [10]
nova	Compute, Compute extensions	Authenticate, launch servers, set security groups, control IP addresses on servers, control volumes, and create images.	Chapter 9, "nova Client" [28]
quantum	Networking	Configure networks for guest servers.	Chapter 8, "quantum Client" [18]
swift	Object storage	Gather statistics, list items, update metadata, upload, download and delete files stored by the object storage service. Provides access to a swift installation for ad hoc processing.	Chapter 11, "swift Client" [55]

Client	Associated API	Description	See
heat	Orchestration	Launch stacks from templates, view details of running stacks (including events and resources), update and delete stacks.	Chapter 12, "heat Client" [57]

2. Install the OpenStack Clients

Table of Contents

To manage your servers, images, volumes, isolated networks, and other cloud resources from the command line, install and use the open-source clients.

To install the clients, first install some prerequisite software. Then, install the Python packages. Each Python package is an OpenStack client.

Before You Begin

Before you begin, install the following prerequisite software:

Table 2.1. OpenStack Clients Prerequisite Software

Prerequisite	Description
Python 2.6 or later	Currently, the clients do not support Python 3.
setuptools package	Installed by default on Mac OS X. Many Linux distributions provide packages to make **setuptools** easy to install. Search your package manager for **setuptools** to find an installation package. If you cannot find one, download the **setuptools** package directly from http://pypi.python.org/pypi/setuptools.

pip package	To install the clients on a Mac OS X or Linux system, use **pip**. It is easy to use and ensures that you get the latest version of the clients from the Python Package Index. Also, it lets you update or remove the packages later on.
	
	Install **pip** through the package manager for your system:
	
	• Mac OS X
	`$ sudo easy_install pip`
	• Ubuntu 12.04
	A packaged version enables you to use **dpkg** or **aptitude** to install the python-novaclient.
	`$ aptitude install python-novaclient`
	• Ubuntu
	`$ aptitude install python-pip`
	• RHEL, CentOS, or Fedora
	`$ yum install python-pip`
	• openSUSE 12.2 and earlier
	A packaged version available in the Open Build Service enables you to use **rpm** or **zypper** to install the the python-novaclient.

Prerequisite	Description

	`$ zypper install python-pip`
	Alternatively, you can still use **pip**.
	• openSUSE 12.3
	A packaged version enables you to use **rpm** or **zypper** to install the python-novaclient.
	For example:
	`$ zypper install python-novaclient`

Install the OpenStack Clients

Procedure 2.1. To install the OpenStack clients:

1. **Install or update the client packages**

   You must install each client separately.

   Run the following command to install or update a client package:

   ```
   $ sudo pip install [--update] python-<project>client
   ```

   Where <project> is the project name and is one of the following values:

   - `nova`. Compute API.
   - `quantum`. Networking API.
   - `keystone`. Identity service API.
   - `glance`. Image service API.
   - `swift`. Object storage API.
   - `cinder`. Block storage API.
   - `heat`. Orchestration API.

   For example, to install the nova client, run the following command:

   ```
   $ sudo pip install python-novaclient
   ```

   To update the nova client, run the following command:

   ```
   $ sudo pip install --upgrade python-novaclient
   ```

   To remove the nova client, run the following command:

   ```
   $ sudo pip uninstall python-novaclient
   ```

2. **Set environment variables and authenticate**

   Before you can issue client commands, you must set environment variables and authenticate. See Chapter 3, “Authenticate” [5].

3. Authenticate

To authenticate a tenant to run commands for the OpenStack clients, you issue an authentication request to the OpenStack Identity Service v2.0. To issue this request, you must first set and source environment variables and install the nova client. Then, you issue an authentication request through the nova **credentials** command.

Procedure 3.1. To authenticate:

1. **Set environment variables**

   Before you can issue client commands, you must set environment variables to authenticate.

   You can either edit your bash profile to add and set environment variables or use an `openrc` file downloaded from an OpenStack Dashboard.

   Either edit your .bash_profile file:

   ```
   $ nano ~/.bash_profile
   ```

   Add the following lines to the bash profile. Edit the values for the **OS_USERNAME**, **OS_PASSWORD**, and **OS_TENANT_NAME** variables:

   ```
   export OS_USERNAME=username
   export OS_PASSWORD=password
   export OS_TENANT_NAME=tenant
   export OS_AUTH_URL=https://identity.api.rackspacecloud.com/v2.0/ #an example, insert
    your endpoint here
   export NOVACLIENT_DEBUG=1
   export NOVA_VERSION=2
   ```

   Or download an `openrc` file from the OpenStack Dashboard:

   ```
   #!/bin/bash

   # With the addition of Keystone, to use an openstack cloud you should
   # authenticate against keystone, which returns a **Token** and **Service
   # Catalog**.  The catalog contains the endpoint for all services the
   # user/tenant has access to - including nova, glance, keystone, swift.
   #
   # *NOTE*: Using the 2.0 *auth api* does not mean that compute api is 2.0.  We
   # will use the 1.1 *compute api*
   export OS_AUTH_URL=http://10.0.100.102:5000/v2.0

   # With the addition of Keystone we have standardized on the term **tenant**
   # as the entity that owns the resources.
   export OS_TENANT_ID=feacce5a1fc347f88cfc0dee838429d6
   export OS_TENANT_NAME=tenant

   # In addition to the owning entity (tenant), openstack stores the entity
   # performing the action as the **user**.
   export OS_USERNAME=username

   # With Keystone you pass the keystone password.
   echo "Please enter your OpenStack Password: "
   read -s OS_PASSWORD_INPUT
   export OS_PASSWORD=$OS_PASSWORD_INPUT
   ```

   Source the file:

```
$ source openrc.sh
```

Enter your OpenStack password when prompted.

The following table describes the environment variables:

Table 3.1. OpenStack Clients Environment Variables

Environment Variable	Description
OS_USERNAME	Your OpenStack username.
OS_PASSWORD	Your OpenStack user password.
OS_TENANT_ID	Your tenant ID, usually provided with your username.
OS_TENANT_NAME	Your tenant name, usually provided with your username.
OS_AUTH_URL	The endpoint for the Identity Service (keystone), which the nova client uses for authentication. Include the trailing forward slash (/) in the URL. Otherwise, you receive a 404 error.
NOVACLIENT_DEBUG	Set to 1 to show the underlying cURL commands with embedded API requests in the command responses. Otherwise, omit this variable.
NOVA_VERSION	The version of the API. Set to 2.

After you set the variables, save the file.

2. **Set permissions on and source the bash profile**

   Because the bash profile contains a password, set permissions on it so other people cannot read it:

   ```
   $ chmod 600 ~/.bash_profile
   ```

   To source the variables to make them available in your current shell, run the following command:

   ```
   $ source ~/.bash_profile
   ```

3. **Authenticate**

   To authenticate, you must install the nova client. To install the nova client, see Chapter 2, "Install the OpenStack Clients" [3].

   To verify that you can talk to the API server, authenticate and list images:

   ```
   $ nova credentials
   $ nova image-list
   ```

   Then, list networks:

   ```
   $ nova network-list
   ```

4. **Troubleshooting**

   - If you cannot run commands successfully, make sure that you entered your user name, password, and account number correctly in the bash profile file.
   - If you change any environment variables, either log out and back in or source your bash profile again.

- To override some environment variable settings, you can use the options that are listed at the end of the **nova help** output. For example, you can override the OS_PASSWORD setting in the bash profile by specifying a password on a nova command, as follows:

  ```
  $ nova --password <password> image-list
  ```

  Where *password* is your password.

4. Get the Version for a Client

Search for the version number.

```
$pip freeze | grep python-
```

```
python-glanceclient==0.4.0
python-keystoneclient==0.1.2
-e git+https://github.com/openstack/python-novaclient.
git@077cc0bf22e378c4c4b970f2331a695e440a939f#egg=python_novaclient-dev
python-quantumclient==0.1.1
python-swiftclient==1.1.1
```

You can also use the yolk -l command to see what version of the CLI you have installed.

```
$yolk -l | grep python-novaclient
```

```
python-novaclient - 2.6.10.27    - active development (/Users/your.name/src/
cloud-servers/src/src/python-novaclient)
python-novaclient - 2012.1       - non-active
```

5. Get Help for Client Commands

Use the **help** command to get help for commands, parameters, and subcommands for any OpenStack client.

The syntax is:

```
$ <client-name> help
```

For example, to get help for glance client commands, run the following command:

```
$ glance help
```

The **help** command lists the available commands for the specified client.

Note

Depending on your credentials, you might not have permission to use every command.

To get help for a specific command, enter the command name after the `help` command, as follows:

```
$ <client-name> help <command-name>
```

For example, to get help for the glance **image-show** command, enter the following command:

```
$ glance help image-show
```

The **help** command shows the command usage, a description of the command, and descriptions of any positional and optional arguments, as follows:

```
usage: glance image-show [--human-readable] <IMAGE>

Describe a specific image.

Positional arguments:
  <IMAGE>           Name or ID of image to describe.

Optional arguments:
  --human-readable  Print image size in a human-friendly format.
```

6. keystone Client

Table of Contents

This chapter describes how to use the keystone client.

To install the client, see Chapter 2, "Install the OpenStack Clients" [3].

keystone Command Reference

```
usage: keystone [--os-username <auth-user-name>]
                [--os-password <auth-password>]
                [--os-tenant-name <auth-tenant-name>]
                [--os-tenant-id <tenant-id>] [--os-auth-url <auth-url>]
                [--os-region-name <region-name>]
                [--os-identity-api-version <identity-api-version>]
                [--token <service-token>] [--endpoint <service-endpoint>]
                [--os-cacert <ca-certificate>] [--os-cert <certificate>]
                [--os-key <key>] [--insecure] [--username <auth-user-name>]
                [--password <auth-password>] [--tenant_name <tenant-name>]
                [--auth_url <auth-url>] [--region_name <region-name>]
```

```
catalog             List service catalog, possibly filtered by service.
    ec2-credentials-create
                        Create EC2-compatible credentials for user per tenant
    ec2-credentials-delete
                        Delete EC2-compatible credentials
    ec2-credentials-get
                        Display EC2-compatible credentials
    ec2-credentials-list
                        List EC2-compatible credentials for a user
    endpoint-create     Create a new endpoint associated with a service
    endpoint-delete     Delete a service endpoint
    endpoint-get        Find endpoint filtered by a specific attribute or
                        service type
    endpoint-list       List configured service endpoints
    role-create         Create new role
    role-delete         Delete role
    role-get            Display role details
    role-list           List all roles
    service-create      Add service to Service Catalog
    service-delete      Delete service from Service Catalog
    service-get         Display service from Service Catalog
    service-list        List all services in Service Catalog
    tenant-create       Create new tenant
    tenant-delete       Delete tenant
    tenant-get          Display tenant details
    tenant-list         List all tenants
    tenant-update       Update tenant name, description, enabled status
    token-get           Display the current user token
    user-create         Create new user
```

```
  user-delete         Delete user
  user-get            Display user details.
  user-list           List users
  user-password-update
                      Update user password
  user-role-add       Add role to user
  user-role-list      List roles granted to a user
  user-role-remove    Remove role from user
  user-update         Update user's name, email, and enabled status
  discover            Discover Keystone servers and show authentication
                      protocols and
  bash-completion     Prints all of the commands and options to stdout.
  help                Display help about this program or one of its
  subcommands.
```

```
     Optional arguments:
--os-username <auth-user-name>
                      Defaults to env[OS_USERNAME]
--os-password <auth-password>
                      Defaults to env[OS_PASSWORD]
--os-tenant-name <auth-tenant-name>
                      Defaults to env[OS_TENANT_NAME]
--os-tenant-id <tenant-id>
                      Defaults to env[OS_TENANT_ID]
--os-auth-url <auth-url>
                      Defaults to env[OS_AUTH_URL]
--os-region-name <region-name>
                      Defaults to env[OS_REGION_NAME]
--os-identity-api-version <identity-api-version>
                      Defaults to env[OS_IDENTITY_API_VERSION] or 2.0
--token <service-token>
                      Defaults to env[SERVICE_TOKEN]
--endpoint <service-endpoint>
                      Defaults to env[SERVICE_ENDPOINT]
--os-cacert <ca-certificate>
                      Defaults to env[OS_CA_CERT]
--os-cert <certificate>
                      Defaults to env[OS_CERT]
--os-key <key>        Defaults to env[OS_KEY]
--insecure            Explicitly allow keystoneclient to perform "insecure"
                      SSL (https) requests. The server's certificate will
                      not be verified against any certificate authorities.
                      This option should be used with caution.
--username <auth-user-name>
                      Deprecated
--password <auth-password>
                      Deprecated
--tenant_name <tenant-name>
                      Deprecated
--auth_url <auth-url>
                      Deprecated
--region_name <region-name>
                      Deprecated
```

7. glance Client

Table of Contents

This chapter describes how to use the glance client.

To install the client, see Chapter 2, “Install the OpenStack Clients” [3].

List Images

To see what images are available to you, use this command:

```
$ glance image-list
+--------------------------------------+----------------+--------+--------+
|                  ID                  |      Name      | Status | Server |
+--------------------------------------+----------------+--------+--------+
| 53b205cc-7abc-46eb-aa60-eabc449b4217 |  natty-image   | ACTIVE |        |
| 588d93af-645d-4312-a5b0-81347715a91b |   tty-image    | ACTIVE |        |
| ac6f83b7-078c-47bd-b4c2-4053282da49e | oneiric-image  | ACTIVE |        |
| e110fb7d-2a9e-4da5-923f-5565867ce87a | maverick-image | ACTIVE |        |
+--------------------------------------+----------------+--------+--------+
```

You can also narrow down the list by using grep to find only the CentOS images with a command like this:

```
$ glance image-list | grep 'natty'

| 12 | natty-server-cloudimg-amd64-kernel   | ACTIVE |        |
| 13 | natty-server-cloudimg-amd64          | ACTIVE |        |
```

Add a New Image

Adding a new image to your OpenStack cloud.

This process uses the following commands:

- **glance image-create**
- **glance member-create**
- **glance member-list**
- **glance image-show**

Before You Add an Image

Ensure you have created an image that is OpenStack compatible. For details, see the Image Management chapter in the *OpenStack Compute Administration Manual*.

Upload an image to glance

Assuming you had a CentOS 6.3 image in qcow2 format called centos63.qcow2, the following example will upload it to glance and configure it for public access.

```
$ glance image-create --name centos63-image --disk-format=qcow2 --container-
format=raw --is-public=True < ./centos63.qcow2
```

Update an image

To update an image, use the following command:

```
$ glance image-update <image>
```

Where `image` is the name or ID of the image that you want to update. You can use the following optional arguments to modify the following image properties:

```
Optional arguments:
 –name <NAME>        Name of image.
 –disk-format <DISK_FORMAT>
                     Disk format of image. Acceptable formats: ami, ari,
                     aki, vhd, vmdk, raw, qcow2, vdi, and iso.
 –container-format <CONTAINER_FORMAT>
                     Container format of image. Acceptable formats: ami,
                     ari, aki, bare, and ovf.
 –owner <TENANT_ID>   Tenant who should own image.
 –size <SIZE>        Size of image data (in bytes).
 –min-disk <DISK_GB>  Minimum size of disk needed to boot image (in
                     gigabytes).
 –min-ram <DISK_RAM>  Minimum amount of ram needed to boot image (in
                     megabytes).
 –location <IMAGE_URL>
                     URL where the data for this image already resides. For
                     example, if the image data is stored in swift, you
                     could specify
                     'swift://account:key@example.com/container/obj'.
 –file <FILE>        Local file that contains disk image to be uploaded
                     during update. Alternatively, images can be passed to
                     the client via stdin.
 –checksum <CHECKSUM>
                     Hash of image data used Glance can use for
                     verification.
 –copy-from <IMAGE_URL>
                     Similar to '–location' in usage, but this indicates
                     that the Glance server should immediately copy the
                     data and store it in its configured image store.
 –is-public [True|False]
```

```
                Make image accessible to the public.
–is-protected [True|False]
                Prevent image from being deleted.
–property <key=value>
                Arbitrary property to associate with image. May be
                used multiple times.
–purge-props        If this flag is present, delete all image properties
                not explicitly set in the update request. Otherwise,
                those properties not referenced are preserved.
–human-readable     Print image size in a human-friendly format.
```

To annotate an image with a property that describes the required VIF model, use the `--property` argument.

For example:

```
$ glance image-update --property hw_vif_model=e1000 f16-x86_64-openstack-sda
```

Valid model values vary depending on the `libvirt_type` setting:

libvirt_type setting	Supported model values
qemu or kvm	• virtio • ne2k_pci • pcnet • rtl8139 • e1000
xen	• netfront • ne2k_pci • pcnet • rtl8139 • e1000

Requesting an unsupported VIF model causes the guest instance to fail to launch.

Managing Images

Adding images and setting the access to them can be managed in glance, but you can create images by taking a snapshot of a running instance and view available images, set or delete image metadata, and delete an image, using the nova client.

glance Command Reference

```
usage: glance [–version] [-d] [-v] [-k] [–cert-file CERT_FILE]
          [–key-file KEY_FILE] [–os-cacert <ca-certificate-file>]
          [–ca-file OS_CACERT] [–timeout TIMEOUT] [–no-ssl-compression]
          [-f] [–dry-run] [–ssl] [-H ADDRESS] [-p PORT]
          [–os-username OS_USERNAME] [-I OS_USERNAME]
          [–os-password OS_PASSWORD] [-K OS_PASSWORD]
          [–os-tenant-id OS_TENANT_ID] [–os-tenant-name OS_TENANT_NAME]
```

```
        [-T OS_TENANT_NAME] [–os-auth-url OS_AUTH_URL] [-N OS_AUTH_URL]
        [–os-region-name OS_REGION_NAME] [-R OS_REGION_NAME]
        [–os-auth-token OS_AUTH_TOKEN] [-A OS_AUTH_TOKEN]
        [–os-image-url OS_IMAGE_URL] [-U OS_IMAGE_URL]
        [–os-image-api-version OS_IMAGE_API_VERSION]
        [–os-service-type OS_SERVICE_TYPE]
        [–os-endpoint-type OS_ENDPOINT_TYPE] [-S OS_AUTH_STRATEGY]
        <subcommand> ...

Command-line interface to the OpenStack Images API.

Positional arguments:
 <subcommand>
   add              DEPRECATED! Use image-create instead.
   clear            DEPRECATED!
   delete           DEPRECATED! Use image-delete instead.
   details          DEPRECATED! Use image-list instead.
   image-create     Create a new image.
   image-delete     Delete specified image(s).
   image-download   Download a specific image.
   image-list       List images you can access.
   image-members    DEPRECATED! Use member-list instead.
   image-show       Describe a specific image.
   image-update     Update a specific image.
   index            DEPRECATED! Use image-list instead.
   member-add       DEPRECATED! Use member-create instead.
   member-create    Share a specific image with a tenant.
   member-delete    Remove a shared image from a tenant.
   member-images    DEPRECATED! Use member-list instead.
   member-list      Describe sharing permissions by image or tenant.
   members-replace  DEPRECATED!
   show             DEPRECATED! Use image-show instead.
   update           DEPRECATED! Use image-update instead.
   help             Display help about this program or one of its
                    subcommands.

Optional arguments:
 –version           show program's version number and exit
 -d, –debug         Defaults to env[GLANCECLIENT_DEBUG]
 -v, –verbose       Print more verbose output
 -k, –insecure      Explicitly allow glanceclient to perform"insecure SSL"
                    (https) requests. The server certificate will not be
                    verified against anycertificate authorities. This
                    option should be used with caution.
 –cert-file CERT_FILE
                    Path of certificate file to use in SSL connection.
                    This file can optionally be prepended with the private
                    key.
 –key-file KEY_FILE Path of client key to use in SSL connection. This
                    option is not necessary if your key is prepended to
                    your cert file.
```

```
–os-cacert <ca-certificate-file>
                Path of CA TLS certificate(s) used to verify the remote
                server's certificate. Without this option glance looks
                for the default system CA certificates.
–ca-file OS_CACERT   DEPRECATED! Use –os-cacert.
–timeout TIMEOUT     Number of seconds to wait for a response
–no-ssl-compression  Disable SSL compression when using https.
-f, –force          Prevent select actions from requesting user
                confirmation.
–dry-run            DEPRECATED! Only used for deprecated legacy commands.
–ssl                DEPRECATED! Send a fully-formed endpoint using –os-
                image-url instead.
-H ADDRESS, –host ADDRESS
                DEPRECATED! Send a fully-formed endpoint using –os-
                image-url instead.
-p PORT, –port PORT  DEPRECATED! Send a fully-formed endpoint using –os-
                image-url instead.
–os-username OS_USERNAME
                Defaults to env[OS_USERNAME]
-I OS_USERNAME       DEPRECATED! Use –os-username.
–os-password OS_PASSWORD
                Defaults to env[OS_PASSWORD]
-K OS_PASSWORD       DEPRECATED! Use –os-password.
–os-tenant-id OS_TENANT_ID
                Defaults to env[OS_TENANT_ID]
–os-tenant-name OS_TENANT_NAME
                Defaults to env[OS_TENANT_NAME]
-T OS_TENANT_NAME    DEPRECATED! Use –os-tenant-name.
–os-auth-url OS_AUTH_URL
                Defaults to env[OS_AUTH_URL]
-N OS_AUTH_URL       DEPRECATED! Use –os-auth-url.
–os-region-name OS_REGION_NAME
                Defaults to env[OS_REGION_NAME]
-R OS_REGION_NAME    DEPRECATED! Use –os-region-name.
–os-auth-token OS_AUTH_TOKEN
                Defaults to env[OS_AUTH_TOKEN]
-A OS_AUTH_TOKEN, –auth_token OS_AUTH_TOKEN
                DEPRECATED! Use –os-auth-token.
–os-image-url OS_IMAGE_URL
                Defaults to env[OS_IMAGE_URL]
-U OS_IMAGE_URL, –url OS_IMAGE_URL
                DEPRECATED! Use –os-image-url.
–os-image-api-version OS_IMAGE_API_VERSION
                Defaults to env[OS_IMAGE_API_VERSION] or 1
–os-service-type OS_SERVICE_TYPE
                Defaults to env[OS_SERVICE_TYPE]
–os-endpoint-type OS_ENDPOINT_TYPE
                Defaults to env[OS_ENDPOINT_TYPE]
-S OS_AUTH_STRATEGY, –os_auth_strategy OS_AUTH_STRATEGY
                DEPRECATED! This option is completely ignored.
```

```
See "glance help COMMAND" for help on a specific command.
```

8. quantum Client

Table of Contents

This chapter describes how to use the quantum client.

To install the client, see Chapter 2, “Install the OpenStack Clients” [3].

Overview

Argument parts of API 2.0 command

In general, quantum client command arguments divide into three parts:

Known options

These options are listed in the command's help usage text.

Positional arguments

Positional arguments are mandatory information for an API resource. They must be given in the order.

Unknown options

Unknown options are complementary to known options. To define an unknown option, the format is `--optionname [type=int|bool|dict...] [list=true] [optionvalue]*`. There can be multiple option values for a certain optionname. When there is no optionvalue given, the option is regarded as a `bool` one and value is `true`. The type is python built-in type, such as `int`, `bool`, `float` and `dict`, defaulted to string if not given. Unknown options can be used to provide values for creating, updating a resource and to provide filters to list resources. It is also useful to implement API extension when the known options are not included in the command. pseudo argument `'--'` can be used if the options after it need take advantage of unknown options parser.

Features from cliff

Interactive mode

If there is no command specified, the quantum client will enter into interactive mode:

```
$quantum --os-username admin --os-password password --os-tenant-name admin --
os-auth-url http://localhost:5000/v2.0
(quantum) help

Shell commands (type help <topic>):
===================================
cmdenvironment  edit  hi       l   list  pause  r    save  shell      show
ed              help  history  li  load  py     run  set   shortcuts

Undocumented commands:
======================
EOF  eof  exit  q  quit

Application commands (type help <topic>):
=========================================
=========================================
agent-delete                   net-external-list           subnet-create
agent-list                     net-gateway-connect         subnet-delete
agent-show                     net-gateway-create          subnet-list
agent-update                   net-gateway-delete          subnet-show
dhcp-agent-list-hosting-net    net-gateway-disconnect      subnet-update
dhcp-agent-network-add         net-gateway-list
dhcp-agent-network-remove      net-gateway-show
ext-list                       net-gateway-update
ext-show                       net-list
floatingip-associate           net-list-on-dhcp-agent
floatingip-create              net-show
floatingip-delete              net-update
floatingip-disassociate        port-create
floatingip-list                port-delete
floatingip-show                port-list
help                           port-show
l3-agent-list-hosting-router   port-update
l3-agent-router-add            queue-create
l3-agent-router-remove         queue-delete
lb-healthmonitor-associate     queue-list
lb-healthmonitor-create        queue-show
lb-healthmonitor-delete        quota-delete
lb-healthmonitor-disassociate  quota-list
lb-healthmonitor-list          quota-show
lb-healthmonitor-show          quota-update
lb-healthmonitor-update        router-create
lb-member-create               router-delete
lb-member-delete               router-gateway-clear
lb-member-list                 router-gateway-set
lb-member-show                 router-interface-add
lb-member-update               router-interface-delete
lb-pool-create                 router-list
lb-pool-delete                 router-list-on-l3-agent
lb-pool-list                   router-port-list
lb-pool-show                   router-show
lb-pool-stats                  router-update
lb-pool-update                 security-group-create
lb-vip-create                  security-group-delete
lb-vip-delete                  security-group-list
lb-vip-list                    security-group-rule-create
lb-vip-show                    security-group-rule-delete
lb-vip-update                  security-group-rule-list
net-create                     security-group-rule-show
net-delete                     security-group-show
```

```
(quantum) net-list
+--------------------------------------+----------------
+--------------------------------------+
| id                                   | name           | subnets
                |
+--------------------------------------+----------------
+--------------------------------------+
| 11fc08b7-c3b2-4b0c-bd04-66e279d9c470 | public_net1    | 13cc61f6-b33b-495a-
a49f-83bdc9e439ab |
| 22f53ed1-3f3d-49c7-9162-7ba94d9c0a7e | private_mynet1 | b5a9b952-
dd4f-445a-89c5-f15d0707b8bd |
| 2a405f54-aea0-47d7-8a43-4d5129e22b35 | test1          |
                |
| d322e1ae-e068-4249-b9b3-7ed8b820bfa2 | mynetwork      |
                |
+--------------------------------------+----------------
+--------------------------------------+
```

Output format

We can use -h after each command to show the usage of each command:

```
(quantum) net-list -h
usage: quantum net-list [-h] [-f {csv,html,json,table,yaml}] [-c COLUMN]
                        [--quote {all,minimal,none,nonnumeric}]
                        [--request-format {json,xml}] [-D] [-F FIELD]
                        [-P SIZE] [--sort-key FIELD] [--sort-dir {asc,desc}]

List networks that belong to a given tenant.

optional arguments:
  -h, --help            show this help message and exit
  --request-format {json,xml}
                        the xml or json request format
  -D, --show-details    show detailed info
  -F FIELD, --field FIELD
                        specify the field(s) to be returned by server, can be
                        repeated
  -P SIZE, --page-size SIZE
                        specify retrieve unit of each request, then split one
                        request to several requests
  --sort-key FIELD      sort list by specified fields (This option can be
                        repeated), The number of sort_dir and sort_key should
                        match each other, more sort_dir specified will be
                        omitted, less will be filled with asc as default
                        direction
  --sort-dir {asc,desc}
                        sort list in specified directions (This option can be
                        repeated)

output formatters:
  output formatter options

  -f {csv,html,json,table,yaml}, --format {csv,html,json,table,yaml}
                        the output format, defaults to table
  -c COLUMN, --column COLUMN
                        specify the column(s) to include, can be repeated
```

```
CSV Formatter:
  --quote {all,minimal,none,nonnumeric}
                        when to include quotes, defaults to nonnumeric
```

We can see the output formatters cliff provides to each command. By default, the output format is `table`. Now we choose `csv` output to run the command `net-list`:

```
(quantum) net-list -f csv
"id","name","subnets"
"11fc08b7-c3b2-4b0c-bd04-66e279d9c470","public_net1","13cc61f6-b33b-495a-
a49f-83bdc9e439ab"
"22f53ed1-3f3d-49c7-9162-7ba94d9c0a7e","private_mynet1","b5a9b952-
dd4f-445a-89c5-f15d0707b8bd"
"2a405f54-aea0-47d7-8a43-4d5129e22b35","test1",""
"d322e1ae-e068-4249-b9b3-7ed8b820bfa2","mynetwork",""
```

Column selection

We can see `-c COLUMN` in previous usage output. It can be used to limit the output fields:

```
(quantum) net-list -c id -c name
+--------------------------------------+----------------+
| id                                   | name           |
+--------------------------------------+----------------+
| 11fc08b7-c3b2-4b0c-bd04-66e279d9c470 | public_net1    |
| 22f53ed1-3f3d-49c7-9162-7ba94d9c0a7e | private_mynet1 |
| 2a405f54-aea0-47d7-8a43-4d5129e22b35 | test1          |
| d322e1ae-e068-4249-b9b3-7ed8b820bfa2 | mynetwork      |
+--------------------------------------+----------------+
```

Features from API

Fields selection

If there are `'fields'` in request URL, V2.0 API will extract the list of fields to return. A sample of such URLs is `http://localhost:9696/v2.0/networks.json?fields=id&fields=name`

Quantum client supports this feature by `-F` option in known options part and `--fields` in unknown options part. For example, `quantum -F id net-list -- --fields name`. Only `xx-list` and `xx-show` commands support this feature.

Value filtering

Any other fields except the `'fields'` are used to filter resources. A sample of such URLs is `http://localhost:9696/v2.0/networks.json?name=test1&name=test2`. By the current quantum server's sample DB plugin, the filtering has the same meaning as a SQL clause: `name in ['test1', 'test2']`. Quantum client supports this feature by any key options in unknown option part. For example `quantum net-list -- --name test1 test2`. Only `xx-list` commands support this feature.

Sample quantum command

All commands are run with following environment varialbes set:

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

- List the extensions of the system:

```
$ quantum ext-list -c alias -c name
+-----------------+--------------------------+
| alias           | name                     |
+-----------------+--------------------------+
| agent_scheduler | Agent Schedulers         |
| binding         | Port Binding             |
| quotas          | Quota management support |
| agent           | agent                    |
| provider        | Provider Network         |
| router          | Quantum L3 Router        |
| lbaas           | LoadBalancing service    |
| extraroute      | Quantum Extra Route      |
+-----------------+--------------------------+
```

- Create a network:

```
$ quantum net-create net1
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 2d627131-c841-4e3a-ace6-f2dd75773b6d |
| name                      | net1                                 |
| provider:network_type     | vlan                                 |
| provider:physical_network | physnet1                             |
| provider:segmentation_id  | 1001                                 |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 3671f46ec35e4bbca6ef92ab7975e463     |
+---------------------------+--------------------------------------+
```

Note: Some fields of the created network are invisible to non-admin users.

- Create a network with specified provider network type:

```
$ quantum net-create net2 --provider:network-type local
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
```

```
| admin_state_up            | True                                 |
| id                        | 524e26ea-fad4-4bb0-b504-1ad0dc770e7a |
| name                      | net2                                 |
| provider:network_type     | local                                |
| provider:physical_network |                                      |
| provider:segmentation_id  |                                      |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 3671f46ec35e4bbca6ef92ab7975e463     |
+---------------------------+--------------------------------------+
```

Just as shown above, the unknown option `--provider:network-type` is used to create a `local` provider network.

- Create a subnet:

```
$ quantum subnet-create net1 192.168.2.0/24 --name subnet1
Created a new subnet:
+------------------+----------------------------------------------------+
| Field            | Value                                              |
+------------------+----------------------------------------------------+
| allocation_pools | {"start": "192.168.2.2", "end": "192.168.2.254"} |
| cidr             | 192.168.2.0/24                                     |
| dns_nameservers  |                                                    |
| enable_dhcp      | True                                               |
| gateway_ip       | 192.168.2.1                                        |
| host_routes      |                                                    |
| id               | 15a09f6c-87a5-4d14-b2cf-03d97cd4b456               |
| ip_version       | 4                                                  |
| name             | subnet1                                            |
| network_id       | 2d627131-c841-4e3a-ace6-f2dd75773b6d               |
| tenant_id        | 3671f46ec35e4bbca6ef92ab7975e463                   |
+------------------+----------------------------------------------------+
```

In the above command line, `net1` is the network name, `192.168.2.0/24` is the subnet's CIDR. They are positional arguments. `--name subnet1` is an unknown option, which specifies the subnet's name.

- Create a port with specified IP address:

```
$ quantum port-create net1 --fixed-ip ip_address=192.168.2.40
Created a new port:
+----------------------
+-------------------------------------------------------------------------------------
+
| Field               | Value
                                |
+----------------------
+-------------------------------------------------------------------------------------
+
| admin_state_up      | True
                                |
| binding:capabilities | {"port_filter": false}
                                |
```

```
| binding:vif_type     | ovs
                                |
| device_id            |
                                |
| device_owner         |
                                |
| fixed_ips            | {"subnet_id": "15a09f6c-87a5-4d14-
b2cf-03d97cd4b456", "ip_address": "192.168.2.40"} |
| id                   | f7a08fe4-e79e-4b67-bbb8-a5002455a493
                                |
| mac_address          | fa:16:3e:97:e0:fc
                                |
| name                 |
                                |
| network_id           | 2d627131-c841-4e3a-ace6-f2dd75773b6d
                                |
| status               | DOWN
                                |
| tenant_id            | 3671f46ec35e4bbca6ef92ab7975e463
                                |
+----------------------
+-------------------------------------------------------------------------------
+
```

In the above command line, `net1` is the network name, which is a positional argument. `--fixed-ip ip_address=192.168.2.40` is an option, which specifies the port's fixed IP address we wanted.

- Create a port without specified IP address:

```
$ quantum port-create net1
Created a new port:
+----------------------
+-------------------------------------------------------------------------------
+
| Field                | Value
                               |
+----------------------
+-------------------------------------------------------------------------------
+
| admin_state_up       | True
                               |
| binding:capabilities | {"port_filter": false}
                               |
| binding:vif_type     | ovs
                               |
| device_id            |
                               |
| device_owner         |
                               |
| fixed_ips            | {"subnet_id": "15a09f6c-87a5-4d14-
b2cf-03d97cd4b456", "ip_address": "192.168.2.2"} |
| id                   | baf13412-2641-4183-9533-de8f5b91444c
                               |
| mac_address          | fa:16:3e:f6:ec:c7
                               |
| name                 |
                               |
```

```
| network_id            | 2d627131-c841-4e3a-ace6-f2dd75773b6d
                                |
| status                | DOWN
                                |
| tenant_id             | 3671f46ec35e4bbca6ef92ab7975e463
                                |
+-----------------------
+-------------------------------------------------------------------------------
+
```

We can see that the system will allocate one IP address if we don't specify the IP address in command line.

- Query ports with speficied fixed IP addresses:

```
$ quantum port-list --fixed-ips ip_address=192.168.2.2 ip_address=192.168.2.
40
+--------------------------------------+------+-------------------
+-------------------------------------------------------------------------------
+
| id                                   | name | mac_address       |
 fixed_ips
         |
+--------------------------------------+------+-------------------
+-------------------------------------------------------------------------------
+
| baf13412-2641-4183-9533-de8f5b91444c |      | fa:16:3e:f6:ec:c7 |
 {"subnet_id": "15a09f6c-87a5-4d14-b2cf-03d97cd4b456", "ip_address": "192.
168.2.2"}  |
| f7a08fe4-e79e-4b67-bbb8-a5002455a493 |      | fa:16:3e:97:e0:fc |
 {"subnet_id": "15a09f6c-87a5-4d14-b2cf-03d97cd4b456", "ip_address": "192.
168.2.40"} |
+--------------------------------------+------+-------------------
+-------------------------------------------------------------------------------
+
```

`--fixed-ips ip_address=192.168.2.2 ip_address=192.168.2.40` is one unknown option.

How to find unknown options? The unknown options can be easily found by watching the output of `create_xxx` or `show_xxx` command. For example, in the port creation command, we see the fixed_ips fields, which can be used as an unknown option.

quantum Client Reference

```
agent-delete                  Delete a given agent.
agent-list                    List agents.
agent-show                    Show information of a given agent.
agent-update                  Update a given agent.
dhcp-agent-list-hosting-net   List DHCP agents hosting a network.
dhcp-agent-network-add        Add a network to a DHCP agent.
dhcp-agent-network-remove     Remove a network from a DHCP agent.
ext-list                      List all exts.
ext-show                      Show information of a given resource
```

```
 floatingip-associate           Create a mapping between a floating ip and a
fixed ip.
 floatingip-create              Create a floating ip for a given tenant.
 floatingip-delete              Delete a given floating ip.
 floatingip-disassociate        Remove a mapping from a floating ip to a
fixed ip.
 floatingip-list                List floating ips that belong to a given
tenant.
 floatingip-show                Show information of a given floating ip.
 help                           print detailed help for another command
 l3-agent-list-hosting-router   List L3 agents hosting a router.
 l3-agent-router-add            Add a router to a L3 agent.
 l3-agent-router-remove         Remove a router from a L3 agent.
 lb-healthmonitor-associate     Create a mapping between a health monitor and
a pool.
 lb-healthmonitor-create        Create a healthmonitor
 lb-healthmonitor-delete        Delete a given healthmonitor.
 lb-healthmonitor-disassociate  Remove a mapping from a health monitor to a
pool.
 lb-healthmonitor-list          List healthmonitors that belong to a given
tenant.
 lb-healthmonitor-show          Show information of a given healthmonitor.
 lb-healthmonitor-update        Update a given healthmonitor.
 lb-member-create               Create a member
 lb-member-delete               Delete a given member.
 lb-member-list                 List members that belong to a given tenant.
 lb-member-show                 Show information of a given member.
 lb-member-update               Update a given member.
 lb-pool-create                 Create a pool
 lb-pool-delete                 Delete a given pool.
 lb-pool-list                   List pools that belong to a given tenant.
 lb-pool-show                   Show information of a given pool.
 lb-pool-stats                  Retrieve stats for a given pool.
 lb-pool-update                 Update a given pool.
 lb-vip-create                  Create a vip
 lb-vip-delete                  Delete a given vip.
 lb-vip-list                    List vips that belong to a given tenant.
 lb-vip-show                    Show information of a given vip.
 lb-vip-update                  Update a given vip.
 net-create                     Create a network for a given tenant.
 net-delete                     Delete a given network.
 net-external-list              List external networks that belong to a given
tenant
 net-gateway-connect            Add an internal network interface to a
router.
 net-gateway-create             Create a network gateway.
 net-gateway-delete             Delete a given network gateway.
 net-gateway-disconnect         Remove a network from a network gateway.
 net-gateway-list               List network gateways for a given tenant.
 net-gateway-show               Show information of a given network gateway.
 net-gateway-update             Update the name for a network gateway.
 net-list                       List networks that belong to a given tenant.
 net-list-on-dhcp-agent         List the networks on a DHCP agent.
 net-show                       Show information of a given network.
 net-update                     Update network's information.
 port-create                    Create a port for a given tenant.
 port-delete                    Delete a given port.
 port-list                      List ports that belong to a given tenant.
 port-show                      Show information of a given port.
 port-update                    Update port's information.
```

```
 queue-create                  Create a queue.
 queue-delete                  Delete a given queue.
 queue-list                    List queues that belong to a given tenant.
 queue-show                    Show information of a given queue.
 quota-delete                  Delete defined quotas of a given tenant.
 quota-list                    List defined quotas of all tenants.
 quota-show                    Show quotas of a given tenant
 quota-update                  Define tenant's quotas not to use defaults.
 router-create                 Create a router for a given tenant.
 router-delete                 Delete a given router.
 router-gateway-clear          Remove an external network gateway from a
router.
 router-gateway-set            Set the external network gateway for a
router.
 router-interface-add          Add an internal network interface to a
router.
 router-interface-delete       Remove an internal network interface from a
router.
 router-list                   List routers that belong to a given tenant.
 router-list-on-l3-agent       List the routers on a L3 agent.
 router-port-list              List ports that belong to a given tenant,
with specified router
 router-show                   Show information of a given router.
 router-update                 Update router's information.
 security-group-create         Create a security group.
 security-group-delete         Delete a given security group.
 security-group-list           List security groups that belong to a given
tenant.
 security-group-rule-create    Create a security group rule.
 security-group-rule-delete    Delete a given security group rule.
 security-group-rule-list      List security group rules that belong to a
given tenant.
 security-group-rule-show      Show information of a given security group
rule.
 security-group-show           Show information of a given security group.
 subnet-create                 Create a subnet for a given tenant.
 subnet-delete                 Delete a given subnet.
 subnet-list                   List networks that belong to a given tenant.
 subnet-show                   Show information of a given subnet.
 subnet-update                 Update subnet's information.
```

9. nova Client

Table of Contents

This chapter describes how to use the nova client.

To install the client, see Chapter 2, “Install the OpenStack Clients” [3].

List Instances, Images, and Flavors

Before you can go about the business of building your cloud, you want to know what images are available to you by asking the image service what kinds of configurations are available. The image service could be compared to iTunes for your cloud - you can view the playlist of images before using your favorite image to create a new instance in the cloud. To get the list of images, their names, status, and ID, use this command:

```
$ nova image-list
+--------------------------------------+----------------+--------+--------+
|                  ID                  |      Name      | Status | Server |
+--------------------------------------+----------------+--------+--------+
| 53b205cc-7abc-46eb-aa60-eabc449b4217 |  natty-image   | ACTIVE |        |
| 588d93af-645d-4312-a5b0-81347715a91b |   tty-image    | ACTIVE |        |
| ac6f83b7-078c-47bd-b4c2-4053282da49e | oneiric-image  | ACTIVE |        |
| e110fb7d-2a9e-4da5-923f-5565867ce87a | maverick-image | ACTIVE |        |
+--------------------------------------+----------------+--------+--------+
```

Next you need to know the relative sizes of each of these.

```
$ nova flavor-list
+----+-----------+-----------+------+-----------+------+-------+-------------+
| ID |    Name   | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor |
+----+-----------+-----------+------+-----------+------+-------+-------------+
| 1  |  m1.tiny  |    512    |  0   |     0     |      |   1   |     1.0     |
| 2  |  m1.small |    2048   |  10  |     20    |      |   1   |     1.0     |
| 3  | m1.medium |    4096   |  10  |     40    |      |   2   |     1.0     |
| 4  |  m1.large |    8192   |  10  |     80    |      |   4   |     1.0     |
| 5  | m1.xlarge |   16384   |  10  |    160    |      |   8   |     1.0     |
+----+-----------+-----------+------+-----------+------+-------+-------------+
```

You can also narrow down the list by using grep to find only the CentOS images with a command like this:

```
$ nova image-list | grep 'natty'

| 12 | natty-server-cloudimg-amd64-kernel   | ACTIVE |        |
| 13 | natty-server-cloudimg-amd64          | ACTIVE |        |
```

Launch an Instance

Launching a new instance on OpenStack.

Commands Used

This process uses the following commands:

- **nova boot**
- **nova list**
- **nova show**

Before Launch

With the information about what is available to you, you can choose the combination of image and flavor to create your virtual servers and launch instances.

Create Your Server with the nova Client

Procedure 9.1. To create and boot your server with the nova client:

1. Issue the following command. In the command, specify the server name, flavor ID, and image ID:

```
$ nova boot myUbuntuServer --image "3afe97b2-26dc-49c5-a2cc-a2fc8d80c001" --flavor 6
```

The command returns a list of server properties. The status field indicates whether the server is being built or is active. A status of `BUILD` indicates that your server is being built.

```
+--------------------------+--------------------------------------+
| Property                 | Value                                |
+--------------------------+--------------------------------------+
| OS-DCF:diskConfig        | AUTO                                 |
| accessIPv4               |                                      |
| accessIPv6               |                                      |
| adminPass                | ZbaYPZf6r2an                         |
| config_drive             |                                      |
| created                  | 2012-07-27T19:59:31Z                 |
| flavor                   | 8GB Standard Instance                |
| hostId                   |                                      |
| id                       | d8093de0-850f-4513-b202-7979de6c0d55 |
| image                    | Ubuntu 12.04                         |
| metadata                 | {}                                   |
| name                     | myUbuntuServer                       |
| progress                 | 0                                    |
| status                   | BUILD                                |
| tenant_id                | 345789                               |
| updated                  | 2012-07-27T19:59:31Z                 |
| user_id                  | 170454                               |
+--------------------------+--------------------------------------+
```

2. Copy the server ID value from the `id` field in the output. You use this ID to get details for your server to determine if it built successfully.

   Copy the administrative password value from the `adminPass` field. You use this value to log into your server.

Launch from a Volume

The Compute service has support for booting an instance from a volume.

Manually Creating a Bootable Volume

To manually create a bootable volume, mount the volume to an existing instance, and then build a volume-backed image. Here is an example based on exercises/boot_from_volume.sh. This example assumes that you have a running instance with a 1GB

volume mounted at /dev/vdc. These commands will make the mounted volume bootable using a CirrOS image. As root:

```
# mkfs.ext3 -b 1024 /dev/vdc 1048576
# mkdir /tmp/stage
# mount /dev/vdc /tmp/stage

# cd /tmp
# wget https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-
rootfs.img.gz
# gunzip cirros-0.3.0-x86_64-rootfs.img.gz
# mkdir /tmp/cirros
# mount /tmp/cirros-0.3.0-x86_64-rootfs.img /tmp/cirros

# cp -pr /tmp/cirros/* /tmp/stage
# umount /tmp/cirros
# sync
# umount /tmp/stage
```

Detach the volume once you are done.

Creating a Bootable Volume from an Image

Cinder has the ability to create a bootlable volume from an image stored in Glance.

```
# cinder create --image-id <image_id> --display-name my-bootable-vol <size>
```

This feature is also mirrored in Nova:

```
# nova volume-create --image-id <image_id> --display-name my-bootable-vol
 <size>
```

Note

As of Grizzly, the following block storage drivers are compatible: iSCSI-based, LVM, and Ceph.

Make sure you configure Cinder with the relevant Glance options:

Table 9.1. List of configuration flags for NFS

Flag Name	Type	Default	Description
glance_host	Optional	$my_ip	(StrOpt) default glance hostname or ip
glance_port	Optional	9292	(IntOpt) default glance port
glance_api_servers	Optional	$glance_host:$glance_port	(ListOpt) A list of the glance api servers available to cinder: ([hostname|ip]:port) (list value)
glance_api_version	Optional	1	(IntOpt) default version of the glance api to use
glance_num_retries	Optional	0	(IntOpt) Number retries when downloading an image from glance
glance_api_insecure	Optional	false	(BoolOpt) Allow to perform insecure SSL (https) requests to glance

Booting an instance from the volume

To boot a new instance from the volume, use the **nova boot** command with the `--block_device_mapping` flag. The output for **nova help boot** shows the following documentation about this flag:

```
 --block_device_mapping <dev_name=mapping>
                        Block device mapping in the format <dev_name>=
<id>:<type>:<size(GB)>:<delete_on_terminate>.
```

The command arguments are:

`dev_name`	A device name where the volume will be attached in the system at `/dev/`*`dev_name`*. This value is typically `vda`.
`id`	The ID of the volume to boot from, as shown in the output of **nova volume-list**.
`type`	This is either `snap`, which means that the volume was created from a snapshot, or anything other than `snap` (a blank string is valid). In the example above, the volume was not created from a snapshot, so we will leave this field blank in our example below.
`size (GB)`	The size of the volume, in GB. It is safe to leave this blank and have the Compute service infer the size.
`delete_on_terminate`	A boolean to indicate whether the volume should be deleted when the instance is terminated. True can be specified as `True` or `1`. False can be specified as `False` or `0`.

Note

Because of bug #1163566, you must specify an image when booting from a volume in Horizon, even though this image will not be used.

The following example will attempt boot from volume on the command line with ID=13, it will not delete on terminate. Replace the `--key_name` with a valid keypair name:

```
$ nova boot --flavor 2 --key_name mykey --block_device_mapping vda=13:::0
 boot-from-vol-test
```

Associating ssh keys with instances

Creating New Keys

The command:

```
$ nova keypair-add mykey > mykey.pem
```

will create a key named `mykey` which you can associate with instances. Save the file `mykey.pem` to a secure location as it will allow root access to instances the `mykey`key is associated with.

Uploading Existing Keys

The command:

```
$ nova keypair-add --pub-key mykey.pub mykey
```

will upload the existing public key `mykey.pub` and associate it with the name `mykey`. You will need to have the matching private key to access instances associated with this key.

Adding Keys to Your Instance

To associate a key with an instance on boot add `--key_name mykey` to your command line for example:

```
$ nova boot --image ubuntu-cloudimage --flavor 1 --key_name mykey
```

Insert metadata during launch

When booting a server, you can also add metadata, so that you can more easily identify it amongst your ever-growing elastic cloud. Use the `--meta` option with a key=value pair, where you can make up the string for both the key and the value. For example, you could add a description and also the creator of the server.

```
$ nova boot --image=natty-image --flavor=2 smallimage2 --meta description=
'Small test image' --meta creator=joecool
```

When viewing the server information, you can see the metadata included on the `metadata` line:

```
$ nova show smallimage2
+------------------------
+-------------------------------------------------------------+
|  Property  |          Value     |
+------------------------
+-------------------------------------------------------------+
|   OS-DCF:diskConfig  |          MANUAL     |
| OS-EXT-STS:power_state |     1     |
| OS-EXT-STS:task_state  |    None     |
|  OS-EXT-STS:vm_state  |          active     |
| accessIPv4  |         |
| accessIPv6  |         |
|      config_drive  |         |
|  created  |   2012-05-16T20:48:23Z    |
|   flavor  |         m1.small     |
|   hostId  |    de0c201e62be88c61aeb52f51d91e147acf6cf2012bb57892e528487  |
|     id   |  8ec95524-7f43-4cce-a754-d3e5075bf915   |
|   image   |       natty-image     |
|  key_name  |         |
|  metadata  | {u'description': u'Small test image', u'creator': u'joecool'} |
|    name   |       smallimage2     |
|    private network  |      172.16.101.11    |
|  progress  |     0     |
|     public network  |       10.4.113.11     |
```

```
|   status   |          ACTIVE     |
| tenant_id  |    e830c2fbb7aa4586adf16d61c9b7e482   |
|  updated   |   2012-05-16T20:48:35Z    |
|  user_id   |    de3f4e99637743c7b6d27faca4b800a9   |
+------------------------
+-------------------------------------------------------------+
```

Providing User Data to Instances

`User Data` is a special key in the metadata service which holds a file that cloud aware applications within the guest instance can access. For example the cloudinit system is an open source package from Ubuntu that handles early initialization of a cloud instance that makes use of this `user data`.

This user-data can be put in a file on your local system and then passed in at instance creation with the flag `--user-data <user-data-file>` for example:

```
    $ nova boot --image ubuntu-cloudimage --flavor 1 --user-data mydata.file
```

Injecting Files into Instances

Arbitrary local files can also be placed into the instance file system at creation time using the `--file <dst-path=src-path>` option. You may store up to 5 files. For example if you have a special authorized_keys file named `special_authorized_keysfile` that you want to put on the instance rather than using the regular ssh key injection for some reason you can use the following command:

```
$nova boot --image ubuntu-cloudimage --flavor 1 --file /root/.ssh/
authorized_keys=special_authorized_keysfile
```

Change Server Configuration

After you have created a server, you may need to increase its size, change the image used to build it, or perform other configuration changes.

Commands Used

This process uses the following commands:

- **nova resize***
- **nova rebuild**

Increase or Decrease Server Size

Server size is changed by applying a different flavor to the server. Before you begin, use nova flavor-list to review the flavors available to you.

```
$ nova flavor-list
+----+-----------+-----------+------+-----------+------+-------+-------------+
| ID |    Name   | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor |
```

```
+----+-----------+-----------+------+-----------+------+-------+-------------+
| 1  |  m1.tiny  |    512    |  0   |     0     |      |   1   |     1.0     |
| 2  |  m1.small |    2048   |  10  |     20    |      |   1   |     1.0     |
| 3  | m1.medium |    4096   |  10  |     40    |      |   2   |     1.0     |
| 4  |  m1.large |    8192   |  10  |     80    |      |   4   |     1.0     |
| 5  | m1.xlarge |   16384   |  10  |    160    |      |   8   |     1.0     |
+----+-----------+-----------+------+-----------+------+-------+-------------+
```

In this example, we'll take a server originally configured with the `m1.tiny` flavor and resize it to `m1.small`.

```
     $ nova show acdfb2c4-38e6-49a9-ae1c-50182fc47e35
+------------------------
+----------------------------------------------------------+
|        Property        |                            Value
     |
+------------------------
+----------------------------------------------------------+
|   OS-DCF:diskConfig    |                            MANUAL
     |
| OS-EXT-STS:power_state |                              1
     |
| OS-EXT-STS:task_state  |                             None
     |
|  OS-EXT-STS:vm_state   |                            active
     |
|       accessIPv4       |
     |
|       accessIPv6       |
     |
|      config_drive      |
     |
|        created         |                    2012-05-09T15:47:48Z
     |
|         flavor         |                          m1.tiny
     |
|         hostId         |
 de0c201e62be88c61aeb52f51d91e147acf6cf2012bb57892e528487 |
|           id           |           acdfb2c4-38e6-49a9-ae1c-50182fc47e35
     |
|         image          |                       maverick-image
     |
|        key_name        |
     |
|        metadata        |                            {}
     |
|          name          |                        resize-demo
     |
|    private network     |                        172.16.101.6
     |
|        progress        |                             0
     |
|     public network     |                         10.4.113.6
     |
|         status         |                           ACTIVE
     |
|       tenant_id        |             e830c2fbb7aa4586adf16d61c9b7e482
     |
```

```
|        updated         |                   2012-05-09T15:47:59Z
      |
|        user_id         |             de3f4e99637743c7b6d27faca4b800a9
      |
+------------------------
+-----------------------------------------------------------+
```

Use the resize command with the server's ID (6beefcf7-9de6-48b3-9ba9-e11b343189b3) and the ID of the desired flavor (2):

```
$ nova resize 6beefcf7-9de6-48b3-9ba9-e11b343189b3 2
```

While the server is rebuilding, its status will be displayed as RESIZING.

```
$ nova list
+--------------------------------------+-------------+--------
+----------------------------------------+
| ID                                   | Name        | Status | Networks
                          |
+--------------------------------------+-------------+--------
+----------------------------------------+
| 970e4ca0-f9b7-4c44-80ed-bf0152c96ae1 | resize-demo | RESIZE | private=172.
16.101.6, public=10.4.113.6 |
+--------------------------------------+-------------+--------
+----------------------------------------+
```

When the resize operation is completed, the status displayed is VERIFY_RESIZE. This prompts the user to verify that the operation has been successful; to confirm:

```
$ nova resize-confirm 6beefcf7-9de6-48b3-9ba9-e11b343189b3
```

However, if the operation has not worked as expected, you can revert it by doing:

```
$ nova resize-revert 6beefcf7-9de6-48b3-9ba9-e11b343189b3
```

In both cases, the server status should go back to ACTIVE.

Instance evacuation

As cloud administrator, while you are managing your cloud, you may get to the point where one of the cloud compute nodes fails. For example, due to hardware malfunction. At that point you may use server evacuation in order to make managed instances available again.

Before Evacuation

With the information about instance configuration, like if it is running on shared storage, you can choose the required evacuation parameters for your case. Use the **nova host-**

list command to list the hosts and find new host for the evacuated instance. In order to preserve user data on server disk, target host has to have preconfigured shared storage with down host. As well, you have to validate that the current vm host is down. Otherwise the evacuation will fail with error.

To evacuate your server without shared storage:

nova evacuate performs an instance evacuation from down host to specified host. The instance will be booted from a new disk, but will preserve the configuration, e.g. id, name, uid, ip...etc. New instance password can be passed to the command using the `--password <pwd>` option. If not given it will be generated and printed after the command finishes successfully.

```
$nova evacuate evacuated_server_name host_b
```

The command returns a new server password.

```
+-----------+--------------+
| Property  |    Value     |
+-----------+--------------+
| adminPass | kRAJpErnT4xZ |
+-----------+--------------+
```

Evacuate server to specified host and preserve user data

In order to preserve the user disk data on the evacuated server the OpenStack Compute should be deployed with shared filesystem. Refer to the shared storage section in the Configure migrations guide in order to configure your system. In this scenario the password will remain unchanged.

```
$nova evacuate evacuated_server_name host_b --on-shared-storage
```

Stop and Start an Instance

There are two methods for stopping and starting an instance:

- **nova pause** / **nova unpause**
- **nova suspend** / **nova resume**

Pause and Unpause

nova pause stores the state of the VM in RAM. A paused instance continues to run, albeit in a "frozen" state.

Suspend and Resume

nova suspend initiates a hypervisor-level suspend operation. Suspending an instance stores the state of the VM on disk; all memory is written to disk and the virtual machine is stopped. Suspending an instance is thus similar to placing a device in hibernation, and makes memory and vCPUs available. Administrators may want to suspend an instance for system maintenance, or if the instance is not frequently used.

Rebooting an instance

nova reboot performs a reboot of a running instance. By default, this is a "soft" reboot, which will attempt a graceful shutdown and restart of the instance. To perform a "hard" reboot (i.e., a power cycle of the instance), pass the `--hard` flag as an argument.

Manage Security Groups

A security group is a named collection of network access rules that can be used to limit the types of traffic that have access to instances. When you spawn an instance, you can assign it to one or more groups. For each security group, the associated rules permit you to manage the allowed traffic to instances within the group. Any incoming traffic which is not matched by a rule is denied by default. At any time, it is possible to add or remove rules within a security group. Rules are automatically enforced as soon as they are created.

Before you begin, use **nova secgroup-list** to view the available security groups (specify `--all-tenants` if you are a cloud administrator wanting to view all tenants' groups) . You can also view the rules for a security group with **nova secgroup-list-rules**.

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

$ nova secgroup-list-rules default
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port |  IP Range | Source Group |
+-------------+-----------+---------+-----------+--------------+
|     tcp     |     80    |    80   | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

In this example, the default security group has been modified to allow HTTP traffic on the instance by permitting TCP traffic on Port 80.

Add or delete a security group

Security groups can be added with **nova secgroup-create**.

The following example shows the creation of the security group `secure1`. After the group is created, it can be viewed in the security group list.

```
$ nova secgroup-create secure1 "Test security group"
+---------+---------------------+
|   Name  |     Description     |
+---------+---------------------+
| secure1 | Test security group |
+---------+---------------------+

$ nova secgroup-list
+---------+---------------------+
|   Name  |     Description     |
+---------+---------------------+
| default |       default       |
| secure1 | Test security group |
+---------+---------------------+
```

Security groups can be deleted with **nova secgroup-delete**. The default security group cannot be deleted. The default security group contains these initial settings:

- All the traffic originated by the instances (outbound traffic) is allowed
- All the traffic destined to instances (inbound traffic) is denied
- All the instances inside the group are allowed to talk to each other

Note

You can add extra rules into the default security group for handling the egress traffic. Rules are ingress only at this time.

In the following example, the group `secure1` is deleted. When you view the security group list, it no longer appears.

```
$ nova secgroup-delete secure1
$ nova secgroup-list
+---------+-------------+
|   Name  | Description |
+---------+-------------+
| default |   default   |
+---------+-------------+
```

Modify security group rules

The security group rules control the incoming traffic that is allowed to the instances in the group, while all outbound traffic is automatically allowed.

Note

It is not possible to change the default outbound behaviour.

Every security group rule is a policy which allows you to specify inbound connections that are allowed to access the instance, by source address, destination port and IP protocol, (TCP, UDP or ICMP). Currently, ipv6 and other protocols cannot be managed with the security rules, making them permitted by default. To manage such, you can deploy a firewall in front of your OpenStack cloud to control other types of traffic. The command requires the following arguments for both TCP and UDP rules :

- <secgroup> ID of security group.
- <ip_proto> IP protocol (icmp, tcp, udp).
- <from_port> Port at start of range.
- <to_port> Port at end of range.
- <cidr> CIDR for address range.

For ICMP rules, instead of specifying a begin and end port, you specify the allowed ICMP code and ICMP type:

- <secgroup> ID of security group.
- <ip_proto> IP protocol (with icmp specified).
- <ICMP_code> The ICMP code.
- <ICMP_type> The ICMP type.
- <cidr> CIDR for the source address range.

Note

Entering "-1" for both code and type indicates that all ICMP codes and types should be allowed.

The CIDR notation

That notation allows you to specify a base IP address and a suffix that designates the number of significant bits in the IP address used to identify the network. For example, by specifying a 88.170.60.32/27, you specify 88.170.60.32 as the **base IP** and 27 as the **suffix**. Since you use an IPV4 format, there are only 5 bits available for the host part (32 minus 27). The 0.0.0.0/0 notation means you allow the entire IPV4 range, meaning allowing all addresses.

For example, in order to allow any IP address to access to a web server running on one of your instance inside the default security group:

```
$ nova secgroup-add-rule default tcp 80 80 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port |  IP Range | Source Group |
+-------------+-----------+---------+-----------+--------------+
|     tcp     |     80    |    80   | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

In order to allow any IP address to ping an instance inside the default security group (Code 0, Type 8 for the ECHO request.):

```
$ nova secgroup-add-rule default icmp 0 8 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port |  IP Range | Source Group |
```

```
+-------------+-----------+---------+-----------+--------------+
|     icmp    |     0     |    8    | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

```
$ nova secgroup-list-rules default
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port |  IP Range | Source Group |
+-------------+-----------+---------+-----------+--------------+
|     tcp     |     80    |    80   | 0.0.0.0/0 |              |
|     icmp    |     0     |    8    | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

In order to delete a rule, you need to specify the exact same arguments you used to create it:

- <secgroup> ID of security group.
- <ip_proto> IP protocol (icmp, tcp, udp).
- <from_port> Port at start of range.
- <to_port> Port at end of range.
- <cidr> CIDR for address range.

```
$ nova secgroup-delete-rule default tcp 80 80 0.0.0.0/0
```

Manage Floating IP Addresses

A floating IP address is an IP address (typically public) that can be dynamically assigned to an instance. Pools of floating IP addresses are created outside of python-novaclient with the **nova-manage floating *** commands. Refer to "`Configuring Public (Floating) IP Addresses`" in the *OpenStack Compute Administration Manual* for more information.

Before you begin, use **nova floating-ip-pool-list** to determine what floating IP pools are available.

```
$ nova floating-ip-pool-list
+------+
| name |
+------+
| nova |
+------+
```

In this example, the only available pool is `nova`.

Reserve and associate floating IP addresses

You can reserve floating IP addresses with the **nova floating-ip-create** command. This command reserves the addresses for the tenant, but does not immediately associate that address with an instance.

```
  $ nova floating-ip-create nova
+--------------+-------------+----------+------+
|      Ip      | Instance Id | Fixed Ip | Pool |
+--------------+-------------+----------+------+
| 50.56.12.232 |     None    |   None   | nova |
+--------------+-------------+----------+------+
```

The floating IP address has been reserved, and can now be associated with an instance with the **nova add-floating-ip** command. For this example, we'll associate this IP address with an image called `smallimage`.

```
   $ nova add-floating-ip smallimage 50.56.12.232
```

After the command is complete, you can confirm that the IP address has been associated with the **nova floating-ip-list** and **nova-list** commands.

```
$ nova floating-ip-list
+--------------+--------------------------------------+------------+------+
|      Ip      |             Instance Id              |  Fixed Ip  | Pool |
+--------------+--------------------------------------+------------+------+
| 50.56.12.232 | 542235df-8ba4-4d08-90c9-b79f5a77c04f | 10.4.113.9 | nova |
+--------------+--------------------------------------+------------+------+

$ nova list
+--------------------------------------+------------+--------
+-------------------------------------------------------+
|                  ID                  |    Name    | Status |
        Networks                       |
+--------------------------------------+------------+--------
+-------------------------------------------------------+
| 4bb825ea-ea43-4771-a574-ca86ab429dcb | tinyimage2 | ACTIVE |        public=
10.4.113.6; private=172.16.101.6        |
| 542235df-8ba4-4d08-90c9-b79f5a77c04f | smallimage | ACTIVE | public=10.4.
113.9, 50.56.12.232; private=172.16.101.9 |
+--------------------------------------+------------+--------
+-------------------------------------------------------+
```

The first table shows that the 50.56.12.232 is now associated with the `smallimage` instance ID, and the second table shows the IP address included under `smallimage`'s public IP addresses.

Remove and de-allocate a floating IP address

To remove a floating IP address from an instance, use the **nova remove-floating-ip** command.

```
$ nova remove-floating-ip smallimage 50.56.12.232
```

After the command is complete, you can confirm that the IP address has been associated with the **nova floating-ip-list** and **nova-list** commands.

```
$ nova floating-ip-list
+--------------+-------------+----------+------+
|      Ip      | Instance Id | Fixed Ip | Pool |
+--------------+-------------+----------+------+
| 50.56.12.232 |     None    |   None   | nova |
+--------------+-------------+----------+------+
$ nova list
+--------------------------------------+------------+--------
+----------------------------------------+
|                  ID                  |    Name    | Status |
 Networks                |
+--------------------------------------+------------+--------
+----------------------------------------+
| 4bb825ea-ea43-4771-a574-ca86ab429dcb | tinyimage2 | ACTIVE | public=10.4.
113.6; private=172.16.101.6 |
| 542235df-8ba4-4d08-90c9-b79f5a77c04f | smallimage | ACTIVE | public=10.4.
113.9; private=172.16.101.9 |
+--------------------------------------+------------+--------
+----------------------------------------+
```

You can now de-allocate the floating IP address, returning it to the pool so that it can be used by another tenant.

```
$ nova floating-ip-delete 50.56.12.232
```

In this example, 50.56.12.232 was the only IP address allocated to this tenant. Running **nova floating-ip-list** after the de-allocation is complete will return no results.

Manage Images

Adding images and setting the access to them can be managed in Glance, but you can create images by taking a snapshot of a running instance and view available images, set or delete image metadata, and delete an image, using the nova CLI.

Manage Volumes

Depending on the setup of your cloud provider, they may give you an endpoint to use to manage volumes, or there may be an extension under the covers. In either case, you can use the nova CLI to manage volumes.

```
   volume-attach       Attach a volume to a server.
   volume-create       Add a new volume.
   volume-delete       Remove a volume.
   volume-detach       Detach a volume from a server.
   volume-list         List all the volumes.
   volume-show         Show details about a volume.
   volume-snapshot-create
   Add a new snapshot.
   volume-snapshot-delete
   Remove a snapshot.
   volume-snapshot-list
   List all the snapshots.
   volume-snapshot-show
   Show details about a snapshot.
```

```
    volume-type-create  Create a new volume type.
    volume-type-delete  Delete a specific flavor
    volume-type-list    Print a list of available 'volume types'.
```

Terminate an Instance

When you no longer need an instance, use the **nova delete** command to terminate it. You can use the instance name or the ID string. You will not receive a notification indicating that the instance has been deleted, but if you run the **nova list** command, the instance will no longer appear in the list.

In this example, we will delete the instance `tinyimage`, which is experiencing an error condition.

```
$ nova list
+--------------------------------------+------------+--------
+-----------------------------------------+
|                  ID                  |    Name    | Status |
  Networks                  |
+--------------------------------------+------------+--------
+-----------------------------------------+
| 30ed8924-f1a5-49c1-8944-b881446a6a51 | tinyimage  | ERROR  | public=10.4.
113.11; private=172.16.101.11 |
| 4bb825ea-ea43-4771-a574-ca86ab429dcb | tinyimage2 | ACTIVE |  public=10.4.
113.6; private=172.16.101.6  |
| 542235df-8ba4-4d08-90c9-b79f5a77c04f | smallimage | ACTIVE |  public=10.4.
113.9; private=172.16.101.9  |
+--------------------------------------+------------+--------
+-----------------------------------------+
$ nova delete tinyimage
$ nova list
+--------------------------------------+------------+--------
+-----------------------------------------+
|                  ID                  |    Name    | Status |
  Networks                  |
+--------------------------------------+------------+--------
+-----------------------------------------+
| 4bb825ea-ea43-4771-a574-ca86ab429dcb | tinyimage2 | ACTIVE |  public=10.4.
113.6; private=172.16.101.6  |
| 542235df-8ba4-4d08-90c9-b79f5a77c04f | smallimage | ACTIVE |  public=10.4.
113.9; private=172.16.101.9  |
+--------------------------------------+------------+--------
+-----------------------------------------+
```

Get an Instance Console

When you need to get a VNC console directly to a server, you can use the `nova get-vnc-console` command to connect.

Managing Bare metal Nodes

If you are using the Bare metal driver, you must inform each Nova Compute host of the physical hardware that it should manage. This is done via the baremetal admin extension.

Create a node with **nova baremetal-node-create**, and then add network interface information to it with **nova baremetal-interface-add** . Nodes and interfaces can be listed and deleted. When a node is in use, its status includes the UUID of the Nova instance that is running on it.

```
$ nova baremetal-node-create --pm_address=1.2.3.4 --pm_user=ipmi --
pm_password=ipmi $(hostname -f) 1 512 10 aa:bb:cc:dd:ee:ff
+------------------+-------------------+
| Property         | Value             |
+------------------+-------------------+
| instance_uuid    | None              |
| pm_address       | 1.2.3.4           |
| interfaces       | []                |
| prov_vlan_id     | None              |
| cpus             | 1                 |
| memory_mb        | 512               |
| prov_mac_address | aa:bb:cc:dd:ee:ff |
| service_host     | ubuntu            |
| local_gb         | 10                |
| id               | 1                 |
| pm_user          | ipmi              |
| terminal_port    | None              |
+------------------+-------------------+

$ nova baremetal-interface-add 1 aa:bb:cc:dd:ee:ff
+-------------+-------------------+
| Property    | Value             |
+-------------+-------------------+
| datapath_id | 0                 |
| id          | 1                 |
| port_no     | 0                 |
| address     | aa:bb:cc:dd:ee:ff |
+-------------+-------------------+

$ nova baremetal-node-list
+----+--------+------+-----------+---------+------------------
  +------+------------+-------------+-------------+---------------+
| ID | Host   | CPUs | Memory_MB | Disk_GB | MAC Address
  | VLAN | PM Address | PM Username | PM Password | Terminal Port |
+----+--------+------+-----------+---------+------------------
  +------+------------+-------------+-------------+---------------+
| 1  | ubuntu | 1    | 512       | 10      | aa:bb:cc:dd:ee:ff
  | None | 1.2.3.4    | ipmi        |             | None          |
+----+--------+------+-----------+---------+------------------
  +------+------------+-------------+-------------+---------------+

$ nova boot --image my-baremetal-image --flavor my-baremetal-flavor test
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| status                      | BUILD                                |
| id                          | cc302a8f-cd81-484b-89a8-b75eb3911b1b |

... wait for instance to become active ...

$ nova baremetal-node-show 1
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
```

```
| instance_uuid    | cc302a8f-cd81-484b-89a8-b75eb3911b1b |
| pm_address       | 1.2.3.4                              |
| interfaces       |
  [{u'datapath_id': u'0', u'id': 1, u'port_no': 0, u'address':
 u'aa:bb:cc:dd:ee:ff'}] |
| prov_vlan_id     | None                                 |
| cpus             | 1                                    |
| memory_mb        | 512                                  |
| prov_mac_address | aa:bb:cc:dd:ee:ff                    |
| service_host     | ubuntu                               |
| local_gb         | 10                                   |
| id               | 1                                    |
| pm_user          | ipmi                                 |
| terminal_port    | None                                 |
+------------------+--------------------------------------+
```

Command List for nova Client

Example 9.1. nova Usage

```
usage: nova [--version] [--debug] [--os-cache] [--timings]
            [--timeout <seconds>] [--os-username <auth-user-name>]
            [--os-password <auth-password>]
            [--os-tenant-name <auth-tenant-name>] [--os-auth-url <auth-url>]
            [--os-region-name <region-name>] [--os-auth-system <auth-system>]
            [--service-type <service-type>] [--service-name <service-name>]
            [--volume-service-name <volume-service-name>]
            [--endpoint-type <endpoint-type>]
            [--os-compute-api-version <compute-api-ver>]
            [--os-cacert <ca-certificate>] [--insecure]
            [--bypass-url <bypass-url>]
            <subcommand> ...

Command-line interface to the OpenStack Nova API.
```

Example 9.2. Positional arguments

```
<subcommand>
    absolute-limits     Print a list of absolute limits for a user
    actions             Retrieve server actions.
    add-fixed-ip        Add new IP address to network.
    add-floating-ip     Add a floating IP address to a server.
    add-secgroup        Add a Security Group to a server.
    aggregate-add-host  Add the host to the specified aggregate.
    aggregate-create    Create a new aggregate with the specified details.
    aggregate-delete    Delete the aggregate by its id.
    aggregate-details   Show details of the specified aggregate.
    aggregate-list      Print a list of all aggregates.
    aggregate-remove-host
                        Remove the specified host from the specified
                        aggregate.
    aggregate-set-metadata
                        Update the metadata associated with the aggregate.
    aggregate-update    Update the aggregate's name and optionally
                        availability zone.
    baremetal-interface-add
                        Add a network interface to a baremetal node
```

```
    baremetal-interface-list
                        List network interfaces associated with a baremetal
                        node
    baremetal-interface-remove
                        Remove a network interface from a baremetal node
    baremetal-node-create
                        Create a baremetal node
    baremetal-node-delete
                        Remove a baremetal node and any associated interfaces
    baremetal-node-list
                        Print a list of available baremetal nodes
    baremetal-node-show
                        Show information about a baremetal node
    boot                Boot a new server.
    cloudpipe-create    Create a cloudpipe instance for the given project
    cloudpipe-list      Print a list of all cloudpipe instances.
    cloudpipe-update   Update a cloudpipe instance
    console-log         Get console log output of a server.
    credentials         Show user credentials returned from auth
    delete              Immediately shut down and delete a server.
    diagnostics         Retrieve server diagnostics.
    dns-create          Create a DNS entry for domain, name and ip.
    dns-create-private-domain
                        Create the specified DNS domain.
    dns-create-public-domain
                        Create the specified DNS domain.
    dns-delete          Delete the specified DNS entry.
    dns-delete-domain   Delete the specified DNS domain.
    dns-domains         Print a list of available dns domains.
    dns-list            List current DNS entries for domain and ip or domain
                        and name.
    endpoints           Discover endpoints that get returned from the
                        authenticate services
    evacuate            Evacuates server from failed host
    fixed-ip-get        Show information for a fixed IP
    fixed-ip-reserve    Reserve a fixed IP
    fixed-ip-unreserve  Unreserve fixed IP
    flavor-create       Create a new flavor
    flavor-delete       Delete a specific flavor
    flavor-key          Set or unset extra_spec for a flavor.
    flavor-list         Print a list of available 'flavors' (sizes of
                        servers).
    flavor-show         Show details about the given flavor.
    floating-ip-create  Allocate a floating IP for the current tenant.
    floating-ip-delete  De-allocate a floating IP.
    floating-ip-list    List floating ips for this tenant.
    floating-ip-pool-list
                        List all floating ip pools.

    get-password        Get password for a server.
    get-spice-console   Get a spice console to a server.
    get-vnc-console     Get a vnc console to a server.
```

```
    host-action         Perform a power action on a host.
    host-describe       Describe a specific host
    host-list           List all hosts by service
    host-update         Update host settings.
    hypervisor-list     List hypervisors.
    hypervisor-servers  List instances belonging to specific hypervisors.
```

```
   hypervisor-show     Display the details of the specified hypervisor.
   hypervisor-stats    Get hypervisor statistics over all compute nodes.
   hypervisor-uptime   Display the uptime of the specified hypervisor.
   image-create        Create a new image by taking a snapshot of a running
                       server.
   image-delete        Delete an image.
   image-list          Print a list of available images to boot from.
   image-meta          Set or Delete metadata on an image.
   image-show          Show details about the given image.
   keypair-add         Create a new key pair for use with instances
   keypair-delete      Delete keypair by its id
   keypair-list        Print a list of keypairs for a user
   list                List active servers.
   list-extensions     List available extensions
   live-migration      Migrates a running instance to a new machine.
   lock                Lock a server.
   meta                Set or Delete metadata on a server.
   migrate             Migrate a server.
   network-list        Print a list of available networks.
   network-show        Show details about the given network.
   pause               Pause a server.
   quota-class-show    List the quotas for a quota class.
   quota-class-update  Update the quotas for a quota class.
   quota-defaults      List the default quotas for a tenant.
   quota-show          List the quotas for a tenant.
   quota-update        Update the quotas for a tenant.
   rate-limits         Print a list of rate limits for a user
   reboot              Reboot a server.
   rebuild             Shutdown, re-image, and re-boot a server.
   remove-fixed-ip     Remove an IP address from a server.
   remove-floating-ip  Remove a floating IP address from a server.
   remove-secgroup     Remove a Security Group from a server.
   rename              Rename a server.
   rescue              Rescue a server.
   reset-state         Reset the state of an instance
   resize              Resize a server.
   resize-confirm      Confirm a previous resize.
   resize-revert       Revert a previous resize (and return to the previous
                       VM).
   resume              Resume a server.
   root-password       Change the root password for a server.
   secgroup-add-group-rule
                       Add a source group rule to a security group.
   secgroup-add-rule   Add a rule to a security group.
   secgroup-create     Create a security group.
   secgroup-delete     Delete a security group.
   secgroup-delete-group-rule
                       Delete a source group rule from a security group.
   secgroup-delete-rule
                       Delete a rule from a security group.
   secgroup-list       List security groups for the current tenant.
   secgroup-list-rules
                       List rules for a security group.

   service-disable     Enable the service
   service-enable      Enable the service
   service-list        Show a list of all running services. Filter by host
and
                       service name.
```

```
show                Show details about the given server.
ssh                 SSH into a server.
start               Start a server.
stop                Stop a server.
suspend             Suspend a server.
unlock              Unlock a server.
unpause             Unpause a server.
unrescue            Unrescue a server.
usage-list          List usage data for all tenants
volume-attach       Attach a volume to a server.
volume-create       Add a new volume.
volume-delete       Remove a volume.
volume-detach       Detach a volume from a server.
volume-list         List all the volumes.
volume-show         Show details about a volume.
volume-snapshot-create
                    Add a new snapshot.
volume-snapshot-delete
                    Remove a snapshot.
volume-snapshot-list
                    List all the snapshots.
volume-snapshot-show
                    Show details about a snapshot.
volume-type-create  Create a new volume type.
volume-type-delete  Delete a specific flavor.
volume-type-list    Print a list of available 'volume types'.
x509-create-cert    Create x509 cert for a user in tenant.
x509-get-root-cert  Fetches the x509 root cert.
bash-completion     Prints all of the commands and options to stdout so
                    that the
help                Display help about this program or one of its
                    subcommands.
network             Show a network
network-create      Create a network
network-delete      Delete a network
network-list        List networks
network             Show a network
network-create      Create a network
network-delete      Delete a network
network-list        List networks
virtual-interface-create
                    Add a new virtual interface to an instance
virtual-interface-delete
                    Removes the specified virtual interface from an
                    instance
virtual-interface-list
                    Add a new virtual interface to an instance
backup-schedule     Show or edit the backup schedule for a server.
backup-schedule-delete
                    Delete the backup schedule for a server.
backup              Backup a server.
service-config      List available service attributes.
service-details     List available service attributes.
service-disable     Disable a specified service.
service-enable      Enable a specified service.
service-list        List available service attributes.
service-servers     List available service attributes.
service-show        List available service attributes.
service-version     List available service attributes.
```

```
   baremetal-interface-add
                       Add a network interface to a baremetal node
   baremetal-interface-list
                       List network interfaces associated with a baremetal
                       node
   baremetal-interface-remove
                       Remove a network interface from a baremetal node
   baremetal-node-create
                       Create a baremetal node
   baremetal-node-delete
                       Remove a baremetal node and any associated interfaces
   baremetal-node-list
                       Print a list of available baremetal nodes
   baremetal-node-show
                       Show information about a baremetal node
   list-extensions     List all the os-api extensions that are available.
   net                 Show a network
   net-create          Create a network
   net-delete          Delete a network
   net-list            List networks
```

Example 9.3. Optional arguments

```
   --version             show program's version number and exit
   --debug                  Print debugging output
   --os-cache               Use the auth token cache.
   --timings                Print call timing info
   --timeout <seconds>   Set HTTP call timeout (in seconds)
   --os-username <auth-user-name>
                            Defaults to env[OS_USERNAME].
   --os-password <auth-password>
                            Defaults to env[OS_PASSWORD].
   --os-tenant-name <auth-tenant-name>
                            Defaults to env[OS_TENANT_NAME].
   --os-auth-url <auth-url>
                            Defaults to env[OS_AUTH_URL].
   --os-region-name <region-name>
                            Defaults to env[OS_REGION_NAME].
   --os-auth-system <auth-system>
                            Defaults to env[OS_AUTH_SYSTEM].
   --service-type <service-type>
                            Defaults to compute for most actions
   --service-name <service-name>
                            Defaults to env[NOVA_SERVICE_NAME]
   --volume-service-name <volume-service-name>
                            Defaults to env[NOVA_VOLUME_SERVICE_NAME]
   --endpoint-type <endpoint-type>
                            Defaults to env[NOVA_ENDPOINT_TYPE] or publicURL.
   --os-compute-api-version <compute-api-ver>
                            Accepts 1.1, defaults to
env[OS_COMPUTE_API_VERSION].
   --os-cacert <ca-certificate>
                            Specify a CA bundle file to use in verifying a
TLS
   (https) server certificate. Defaults to env[OS_CACERT]
   --insecure               Explicitly allow novaclient to perform "insecure"
SSL
                            (https) requests. The server's certificate will
not be
```

```
                            verified against any certificate authorities.
This
                            option should be used with caution.
   --bypass-url <bypass-url>
                            Use this API endpoint instead of the Service
Catalog
```

Usage statistics

The nova command-line tool can provide some basic statistics on resource usage for hosts and instances.

For more sophisticated monitoring, see the Ceilometer project, which is currently under development. You may also wish to consider installing tools such as Ganglia or Graphite if you require access to more detailed data.

Host usage statistics

Use the **nova host-list** command to list the hosts and the nova-related services that are running on them:

```
$ nova host-list
+------------------+-------------+
| host_name        | service     |
+------------------+-------------+
| c2-compute-01    | compute     |
| c2-compute-01    | network     |
| c2-compute-02    | compute     |
| c2-compute-02    | network     |
| c2-compute-03    | compute     |
| c2-compute-03    | network     |
| c2-compute-04    | compute     |
| c2-compute-04    | network     |
| c2-controller-01 | cert        |
| c2-controller-01 | consoleauth |
| c2-controller-01 | scheduler   |
+------------------+-------------+
```

Use the **nova host-describe** command to retrieve a summary of resource usage of all of the instances running on the host. The "cpu" column is the sum of the virtual CPUs of all of the instances running on the host, the "memory_mb" column is the sum of the memory (in MB) allocated to the instances running on the hosts, and the "disk_gb" column is the sum of the root and ephemeral disk sizes (in GB) of the instances running on the hosts.

Note that these values are computed using only information about the flavors of the instances running on the hosts. This command does not query the CPU usage, memory usage, or hard disk usage of the physical host.

```
$ nova host-describe c2-compute-01
+---------------+----------------------------------+-----+-----------
+---------+
| HOST          | PROJECT                          | cpu | memory_mb | disk_gb
 |
+---------------+----------------------------------+-----+-----------
+---------+
| c2-compute-01 | (total)                          | 24  | 96677     | 492
 |
```

```
| c2-compute-01 | (used_max)                          | 2   | 2560      | 0
 |
| c2-compute-01 | (used_now)                          | 4   | 7168      | 0
 |
| c2-compute-01 | f34d8f7170034280a42f6318d1a4af34 | 2   | 2560      | 0
 |
+---------------+-------------------------------------+-----+-----------
+---------+
```

Instance usage statistics

Use the **nova diagnostics** command to retrieve CPU, memory, I/O and network statistics from an instance:

```
$ nova diagnostics ubuntu
+------------------+---------------+
| Property         | Value         |
+------------------+---------------+
| cpu0_time        | 1138410000000 |
| memory           | 524288        |
| memory-actual    | 524288        |
| memory-rss       | 591664        |
| vda_errors       | -1            |
| vda_read         | 334864384     |
| vda_read_req     | 13851         |
| vda_write        | 2985382912    |
| vda_write_req    | 177180        |
| vnet4_rx         | 45381339      |
| vnet4_rx_drop    | 0             |
| vnet4_rx_errors  | 0             |
| vnet4_rx_packets | 106426        |
| vnet4_tx         | 37513574      |
| vnet4_tx_drop    | 0             |
| vnet4_tx_errors  | 0             |
| vnet4_tx_packets | 162200        |
+------------------+---------------+
```

Use the **nova usage-list** command to get summary statistics for each tenant:

```
$ nova usage-list
Usage from 2012-10-10 to 2012-11-08:
+----------------------------------+-----------+--------------+-----------
+---------------+
| Tenant ID                        | Instances | RAM MB-Hours | CPU Hours |
 Disk GB-Hours |
+----------------------------------+-----------+--------------+-----------
+---------------+
| 0eec5c34a7a24a7a8ddad27cb81d2706 | 8         | 240031.10    | 468.81    | 0.
00          |
| 92a5d9c313424537b78ae3e42858fd4e | 5         | 483568.64    | 236.12    | 0.
00          |
| f34d8f7170034280a42f6318d1a4af34 | 106       | 16888511.58  | 9182.88   | 0.
00          |
+----------------------------------+-----------+--------------+-----------
+---------------+
```

10. cinder Client

Table of Contents

This chapter describes how to use the cinder client.

To install the client, see Chapter 2, “Install the OpenStack Clients” [3].

cinder Command Reference

```
Command-line interface to the OpenStack Volume API.

Positional arguments:
  <subcommand>
    absolute-limits     Print a list of absolute limits for a user
    create              Add a new volume.
    credentials         Show user credentials returned from auth
    delete              Remove a volume.
    endpoints           Discover endpoints that get returned from the
                        authenticate services
    extra-specs-list    Print a list of current 'volume types and extra specs'
                        (Admin Only).
    list                List all the volumes.
    quota-class-show    List the quotas for a quota class.
    quota-class-update  Update the quotas for a quota class.
    quota-defaults      List the default quotas for a tenant.
    quota-show          List the quotas for a tenant.
    quota-update        Update the quotas for a tenant.
    rate-limits         Print a list of rate limits for a user
    rename              Rename a volume.
    show                Show details about a volume.
    snapshot-create     Add a new snapshot.
    snapshot-delete     Remove a snapshot.
    snapshot-list       List all the snapshots.
    snapshot-rename     Rename a snapshot.
    snapshot-show       Show details about a snapshot.
    type-create         Create a new volume type.
    type-delete         Delete a specific volume type
    type-key            Set or unset extra_spec for a volume type.
    type-list           Print a list of available 'volume types'.
    bash-completion     Prints all of the commands and options to stdout so
                        that the
    help                Display help about this program or one of its
                        subcommands.
    list-extensions     List all the os-api extensions that are available.

Optional arguments:
  --debug               Print debugging output
  --os-username <auth-user-name>
                        Defaults to env[OS_USERNAME].
  --os-password <auth-password>
```

```
                        Defaults to env[OS_PASSWORD].
  --os-tenant-name <auth-tenant-name>
                        Defaults to env[OS_TENANT_NAME].
  --os-auth-url <auth-url>
                        Defaults to env[OS_AUTH_URL].
  --os-region-name <region-name>
                        Defaults to env[OS_REGION_NAME].
  --service-type <service-type>
                        Defaults to compute for most actions
  --service-name <service-name>
                        Defaults to env[CINDER_SERVICE_NAME]
  --volume-service-name <volume-service-name>
                        Defaults to env[CINDER_VOLUME_SERVICE_NAME]
  --endpoint-type <endpoint-type>
                        Defaults to env[CINDER_ENDPOINT_TYPE] or publicURL.
  --os-volume-api-version <compute-api-ver>
                        Accepts 1,defaults to env[OS_VOLUME_API_VERSION].
  --os-cacert <ca-certificate>
                        Specify a CA bundle file to use in verifying a TLS
                        (https) server certificate. Defaults to env[OS_CACERT]
  --retries <retries>   Number of retries.

See "cinder help COMMAND" for help on a specific command.
```

11. swift Client

Table of Contents

This chapter describes how to use the swift client.

To install the client, see Chapter 2, “Install the OpenStack Clients” [3].

swift Command Reference

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

Commands:
  stat [container] [object]
    Displays information for the account, container, or object depending on
 the
    args given (if any).
  list [options] [container]
    Lists the containers for the account or the objects for a container. -p or
    --prefix is an option that will only list items beginning with that
 prefix.
    -d or --delimiter is option (for container listings only) that will roll
 up
    items with the given delimiter (see Cloud Files general documentation for
    what this means).
  upload [options] container file_or_directory [file_or_directory] [...]
    Uploads to the given container the files and directories specified by the
    remaining args. -c or --changed is an option that will only upload files
    that have changed since the last upload. -S <size> or --segment-size
 <size>
    and --leave-segments are options as well (see --help for more).
  post [options] [container] [object]
    Updates meta information for the account, container, or object depending
 on
    the args given. If the container is not found, it will be created
    automatically; but this is not true for accounts and objects. Containers
    also allow the -r (or --read-acl) and -w (or --write-acl) options. The -m
    or --meta option is allowed on all and used to define the user meta data
    items to set in the form Name:Value. This option can be repeated. Example:
    post -m Color:Blue -m Size:Large
  download --all OR download container [options] [object] [object] ...
    Downloads everything in the account (with --all), or everything in a
    container, or a list of objects depending on the args given. For a single
    object download, you may use the -o [--output] <filename> option to
    redirect the output to a specific file or if "-" then just redirect to
    stdout.
  delete [options] --all OR delete container [options] [object] [object] ...
    Deletes everything in the account (with --all), or everything in a
    container, or a list of objects depending on the args given. Segments of
    manifest objects will be deleted as well, unless you specify the
    --leave-segments option.
```

```
Example:
  swift -A https://auth.api.rackspacecloud.com/v1.0 -U user -K key stat
```

12. heat Client

Table of Contents

This chapter describes how to use the heat client. This service orchestrates multiple composite cloud applications using a REST API that emulates the AWS CloudFormation API.

To install the client, see Chapter 2, “Install the OpenStack Clients” [3].

Create Stack

The following command will create a stack, or template, from an example template file:

```
$ heat stack-create mystack --template-file=/path/to/heat/templates/
WordPress_Single_Instance.template
--parameters="InstanceType=m1.large;DBUsername=wp;DBPassword=
verybadpassword;KeyName=heat_key;LinuxDistribution=F17"
+--------------------------------------+---------------+--------------------
+----------------------+
| id                                   | stack_name    | stack_status       |
 creation_time        |
+--------------------------------------+---------------+--------------------
+----------------------+
| 4c712026-dcd5-4664-90b8-0915494c1332 | mystack       | CREATE_IN_PROGRESS |
 2013-04-03T23:22:08Z |
+--------------------------------------+---------------+--------------------
+----------------------+
```

The values specified in `--parameters` will depend on what parameters are defined in the template. If the template file is hosted on a website then the URL can be specified with `--template-url` instead of `--template-file`

As a template file is being developed, it can be validated without having to attempt stack creation:

```
$ heat stack-create mystack --template-file=/path/to/heat/templates/
WordPress_Single_Instance.template
```

The response will contain an error message if validation failed.

List Stacks

To see what stacks are visible to the current user:

```
$ heat stack-list
```

```
+--------------------------------------+--------------+-----------------
+----------------------+
| id                                   | stack_name    | stack_status    |
 creation_time        |
+--------------------------------------+--------------+-----------------
+----------------------+
| 4c712026-dcd5-4664-90b8-0915494c1332 | mystack       | CREATE_COMPLETE |
 2013-04-03T23:22:08Z |
| 7edc7480-bda5-4e1c-9d5d-f567d3b6a050 | my-otherstack | CREATE_FAILED   |
 2013-04-03T23:28:20Z |
+--------------------------------------+--------------+-----------------
+----------------------+
```

Stack Details

There are a number of commands to explore the state and history of a particular stack. First, to show the details of a stack:

```
$ heat stack-show mystack
```

A stack consists of a collection of resources. This command will list all the resources in a stack, including their current status:

```
$ heat resource-list mystack
+----------------------+--------------------+-----------------
+----------------------+
| logical_resource_id | resource_type      | resource_status | updated_time
      |
+----------------------+--------------------+-----------------
+----------------------+
| WikiDatabase        | AWS::EC2::Instance | CREATE_COMPLETE |
 2013-04-03T23:25:56Z |
+----------------------+--------------------+-----------------
+----------------------+
```

This command will show the details for the specified resource in a stack:

```
$ heat resource-show mystack WikiDatabase
```

Some resources have associated metadata which can change throughout the life-cycle of a resource:

```
$ heat resource-metadata mystack WikiDatabase
```

A series of events is generated during the life-cycle of a stack. This command will display those events.

```
$ heat event-list mystack
+----------------------+----+-----------------------+-----------------
+----------------------+
| logical_resource_id | id | resource_status_reason | resource_status |
 event_time           |
+----------------------+----+-----------------------+-----------------
+----------------------+
| WikiDatabase        | 1  | state changed          | IN_PROGRESS     |
 2013-04-03T23:22:09Z |
```

```
| WikiDatabase        | 2  | state changed          | CREATE_COMPLETE |
 2013-04-03T23:25:56Z |
+---------------------+----+------------------------+-----------------
+----------------------+
```

This command will show the details for a particular event:

```
$ heat event-show WikiDatabase 1
```

Update Stack

The following command is an example of updating an existing stack from a modified template file:

```
$ heat stack-update mystack --template-file=/path/to/heat/templates/
WordPress_Single_Instance_v2.template
--parameters="InstanceType=m1.large;DBUsername=wp;DBPassword=
verybadpassword;KeyName=heat_key;LinuxDistribution=F17"
+--------------------------------------+--------------+-----------------
+----------------------+
| id                                   | stack_name    | stack_status    |
 creation_time        |
+--------------------------------------+--------------+-----------------
+----------------------+
| 4c712026-dcd5-4664-90b8-0915494c1332 | mystack       | UPDATE_COMPLETE |
 2013-04-03T23:22:08Z |
| 7edc7480-bda5-4e1c-9d5d-f567d3b6a050 | my-otherstack | CREATE_FAILED   |
 2013-04-03T23:28:20Z |
+--------------------------------------+--------------+-----------------
+----------------------+
```

Depending on what is being updated, some resources will be updated in-place, while others will be replaced with new resources.

Heat command reference

```
usage: heat [-d] [-v] [-k] [--cert-file CERT_FILE] [--key-file KEY_FILE]
            [--ca-file CA_FILE] [--timeout TIMEOUT]
            [--os-username OS_USERNAME] [--os-password OS_PASSWORD]
            [--os-tenant-id OS_TENANT_ID] [--os-tenant-name OS_TENANT_NAME]
            [--os-auth-url OS_AUTH_URL] [--os-region-name OS_REGION_NAME]
            [--os-auth-token OS_AUTH_TOKEN] [--os-no-client-auth]
            [--heat-url HEAT_URL] [--heat-api-version HEAT_API_VERSION]
            [--os-service-type OS_SERVICE_TYPE]
            [--os-endpoint-type OS_ENDPOINT_TYPE] [-t]
            <subcommand> ...

  stack-create        Create the stack
  stack-delete        Delete the stack
  stack-list          List the user's stacks
  stack-show          Describe the stack
  stack-update        Update the stack
  resource-list       Show list of resources belonging to a stack
  resource-metadata   List resource metadata
  resource-show       Describe the resource
  event-list          List events for a stack
  event-show          Describe the event
```

```
  template-show       Get the template for the specified stack
  template-validate   Validate a template with parameters
  help                Display help about this program or one of its
 subcommands.

Optional arguments:
  -d, --debug           Defaults to env[HEATCLIENT_DEBUG]
  -v, --verbose         Print more verbose output
  -k, --insecure        Explicitly allow the client to perform "insecure" SSL
                        (https) requests. The server's certificate will not be
                        verified against any certificate authorities. This
                        option should be used with caution.
  --cert-file CERT_FILE
                        Path of certificate file to use in SSL connection.
                        This file can optionally be prepended with the private
                        key.
  --key-file KEY_FILE   Path of client key to use in SSL connection. This
                        option is not necessary if your key is prepended to
                        your cert file.
  --ca-file CA_FILE     Path of CA SSL certificate(s) used to verify the
                        remote server's certificate. Without this option the
                        client looks for the default system CA certificates.
  --timeout TIMEOUT     Number of seconds to wait for a response
  --os-username OS_USERNAME
                        Defaults to env[OS_USERNAME]
  --os-password OS_PASSWORD
                        Defaults to env[OS_PASSWORD]
  --os-tenant-id OS_TENANT_ID
                        Defaults to env[OS_TENANT_ID]
  --os-tenant-name OS_TENANT_NAME
                        Defaults to env[OS_TENANT_NAME]
  --os-auth-url OS_AUTH_URL
                        Defaults to env[OS_AUTH_URL]
  --os-region-name OS_REGION_NAME
                        Defaults to env[OS_REGION_NAME]
  --os-auth-token OS_AUTH_TOKEN
                        Defaults to env[OS_AUTH_TOKEN]
  --os-no-client-auth   Do not contact keystone for a token. Defaults to
                        env[OS_NO_CLIENT_AUTH]
  --heat-url HEAT_URL   Defaults to env[HEAT_URL]
  --heat-api-version HEAT_API_VERSION
                        Defaults to env[HEAT_API_VERSION] or 1
  --os-service-type OS_SERVICE_TYPE
                        Defaults to env[OS_SERVICE_TYPE]
  --os-endpoint-type OS_ENDPOINT_TYPE
                        Defaults to env[OS_ENDPOINT_TYPE]
  -t, --token-only      Only send a token for auth, do not send username and
                        password as well.

See "heat help COMMAND" for help on a specific command.
```