

OpenStack

End User Guide

current (April 26, 2014)



OpenStack End User Guide

current (2014-04-26)

Copyright © 2014 OpenStack Foundation Some rights reserved.

OpenStack is an open-source cloud computing platform for public and private clouds. A series of interrelated projects deliver a cloud infrastructure solution. This guide shows OpenStack end users how to create and manage resources in an OpenStack cloud with the OpenStack dashboard and OpenStack client commands.



Except where otherwise noted, this document is licensed under
Creative Commons Attribution 3.0 License.
<http://creativecommons.org/licenses/by/3.0/legalcode>

Table of Contents

How can I use an OpenStack cloud?	7
Conventions	7
Document change history	8
1. OpenStack dashboard	1
Log in to the dashboard	2
Upload and manage images	6
Configure access and security for instances	8
Launch and manage instances	11
Create and manage networks	16
Create and manage object containers	18
Manage volumes	21
Launch and manage stacks	23
2. OpenStack command-line clients	25
Overview	25
Install the OpenStack command-line clients	27
Discover the version number for a client	30
Set environment variables using the OpenStack RC file	31
Manage images	33
Configure access and security for instances	39
Launch instances	43
Manage instances and hosts	54
Provide user data to instances	66
Use snapshots to migrate instances	67
Store metadata on a configuration drive	70
Create and manage networks	75
Manage objects and containers	80
Create and manage stacks	82
Measure cloud resources	85
Manage volumes	88
3. OpenStack Python SDK	96
Install the OpenStack SDK	96
Authenticate	96
Manage images	99
Configure access and security for instances	101
Networking	103
Compute	112
A. Community support	116
Documentation	116
ask.openstack.org	117
OpenStack mailing lists	117
The OpenStack wiki	118
The Launchpad Bugs area	118
The OpenStack IRC channel	119
Documentation feedback	119
OpenStack distribution packages	119

List of Figures

1.1. Project tab 3
1.2. Admin tab 4

List of Tables

2.1. OpenStack services and clients	25
2.2. Prerequisite software	27
2.3. Disk and CD-ROM bus model values	36
2.4. VIF model values	36
2.5. Description of configuration options for configdrive	74

List of Examples

3.1. Create router: complete code listing	108
3.2. Delete network: complete code listing	109
3.3. List routers: complete code listing	110
3.4. List security groups: complete code listing	111
3.5. List subnets: complete code listing	111
3.6. List servers: complete code listing	113
3.7. Create server: complete code listing	114
3.8. Delete: complete code listing	115

How can I use an OpenStack cloud?

As an OpenStack cloud end user, you can provision your own resources within the limits set by administrators.

The examples in this guide show you how to perform tasks by using the following methods:

- OpenStack dashboard. Use this web-based graphical interface, code named [horizon](#), to view, create, and manage resources.
- OpenStack command-line clients. Each core OpenStack project has a command-line client that you can use to run simple commands to view, create, and manage resources in a cloud and automate tasks by using scripts.

You can modify these examples for your specific use cases.

In addition to these ways of interacting with a cloud, you can access the OpenStack APIs directly or indirectly through [cURL](#) commands or open SDKs. You can automate access or build tools to manage resources and services by using the native OpenStack APIs or the EC2 compatibility API.

To use the OpenStack APIs, it helps to be familiar with HTTP/1.1, RESTful web services, the OpenStack services, and JSON or XML data serialization formats.

Conventions

The OpenStack documentation uses several typesetting conventions.

Notices

Notices take three forms:



Note

The information in a note is usually in the form of a handy tip or reminder.



Important

The information in an important notice is something you must be aware of before proceeding.



Warning

The information in warnings is critical. Warnings provide additional information about risk of data loss or security issues.

Command prompts

Commands prefixed with the # prompt are to be executed by the `root` user. These examples can also be executed by using the `sudo` command, if available.

Commands prefixed with the `$` prompt can be executed by any user, including `root`.

Document change history

This version of the guide replaces and obsoletes all previous versions. The following table describes the most recent changes:

Revision Date	Summary of Changes
January 31, 2014	<ul style="list-style-type: none">Removed the command reference appendix. This information is now in the OpenStack Command-Line Interface Reference.
December 30, 2013	<ul style="list-style-type: none">Added the OpenStack Python SDK chapter.
October 17, 2013	<ul style="list-style-type: none">Havana release.
August 19, 2013	<ul style="list-style-type: none">Editorial changes.
July 29, 2013	<ul style="list-style-type: none">First edition of this document.

1. OpenStack dashboard

Table of Contents

Log in to the dashboard	2
Upload and manage images	6
Configure access and security for instances	8
Launch and manage instances	11
Create and manage networks	16
Create and manage object containers	18
Manage volumes	21
Launch and manage stacks	23

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

Log in to the dashboard

The dashboard is available on the node with the `nova-dashboard` server role.

1. Ask the cloud operator for the host name or public IP address from which you can access the dashboard, and for your user name and password.
2. Open a web browser that has JavaScript and cookies enabled.



Note

To use the Virtual Network Computing (VNC) client for the dashboard, your browser must support HTML5 Canvas and HTML5 WebSockets. The VNC client is based on noVNC. For details, see [noVNC: HTML5 VNC Client](#). For a list of supported browsers, see [Browser support](#).

3. In the address bar, enter the host name or IP address for the dashboard.

```
https://ipAddressOrHostName/
```



Note

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

4. On the **Log In** page, enter your user name and password, and click **Sign In**.

The top of the window displays your user name. You can also access **Settings** or sign out of the dashboard.

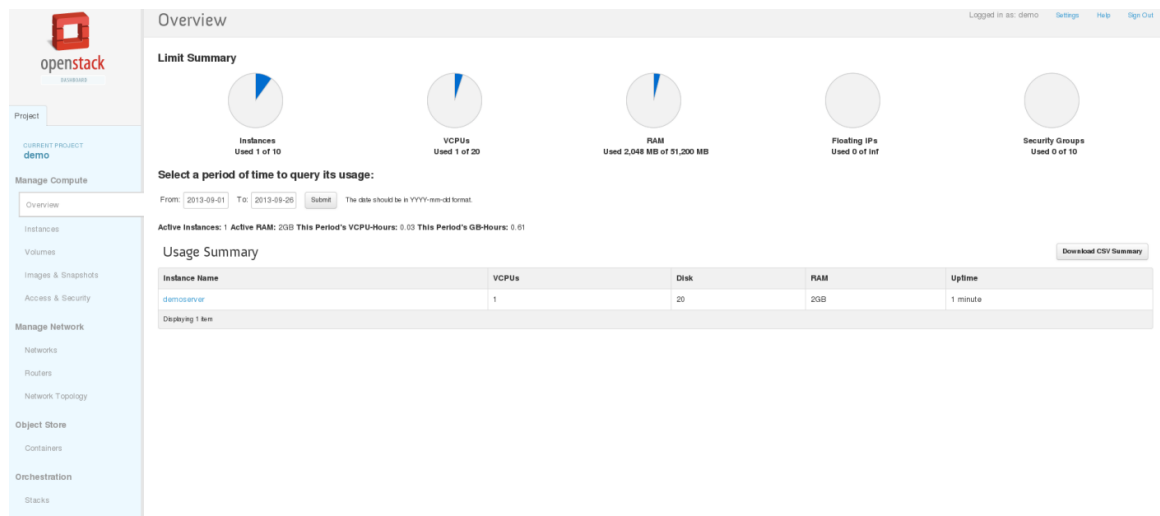
The visible tabs and functions in the dashboard depend on the access permissions, or *roles*, of the user you are logged in as.

- If you are logged in as an end user, the [Project](#) tab is displayed.
- If you are logged in as an administrator, the [Project](#) tab and [Admin](#) tab are displayed.

OpenStack dashboard—Project tab

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

From the **Project** tab, you can view and manage the resources in a selected project, including instances and images. You select the project from the **CURRENT PROJECT** list at the top of the tab.

Figure 1.1. Project tab

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

Manage Compute tabs

Overview

View reports for the project.

Instances

View, launch, create a snapshot from, stop, pause, or reboot instances, or connect to them through VNC.

Volumes

View, create, edit, and delete volumes.

Images & Snapshots

View images, instance snapshots, and volume snapshots created by project users, plus any images that are publicly available. Create, edit, and delete images, and launch instances from images and snapshots.

Access & Security

Use the following tabs to complete these tasks:

Security Groups. View, create, edit, and delete security groups and security group rules.

Keypairs. View, create, edit, import, and delete key pairs.

Floating IPs. Allocate an IP address to or release it from a project.

API Access. View API endpoints.

Manage Network tabs

Networks

Create and manage public and private networks.

Routers

Create and manage subnets.

Network Topology

View the network topology.

Object Store tab

Containers Create and manage object storage.

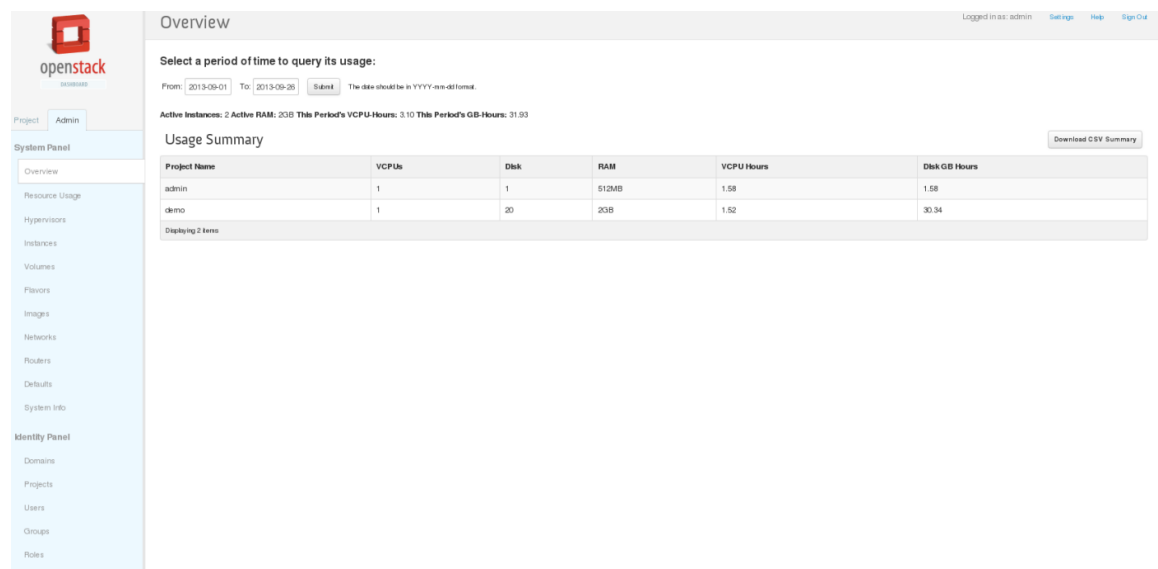
Orchestration tab

Stacks Use the REST API to orchestrate multiple composite cloud applications.

OpenStack dashboard—Admin tab

Administrative users can use the **Admin** tab to view usage and to manage instances, volumes, flavors, images, projects, users, services, and quotas.

Figure 1.2. Admin tab



Access the following categories to complete these tasks:

Overview View basic reports.

Resource Usage Use the following tabs to view the following usages:

Global Disk Usage. View the disk usage for all tenants as an average over the last 30 days.

Global Network Traffic Usage. View the network usage for all tenants as an average over the last 30 days.

Global Object Storage Usage. View the object storage usage for all tenants as an average over the last 30 days.

Global Network Usage. View the network usage for all tenants as an average over the last 30 days.

Stats. View the statistics of all resources.

Hypervisors View the hypervisor summary.

Instances	View, pause, resume, suspend, migrate, soft or hard reboot, and delete running instances that belong to users of some, but not all, projects. Also, view the log for an instance or access an instance through VNC.
Volumes	View, create, edit, and delete volumes and volume types.
Flavors	View, create, edit, view extra specifications for, and delete flavors. A flavor is size of an instance.
Images	View, create, edit properties for, and delete custom images.
Networks	View, create, edit properties for, and delete networks.
Routers	View, create, edit properties for, and delete routers.
Defaults	View default quota values. Quotas are hard-coded in OpenStack Compute and define the maximum allowable size and number of resources.
System Info	Use the following tabs to view the service information: Services. View a list of the services. Compute Services. View a list of all Compute services. Availability Zones. View the availability zones. Host Aggregates. View host aggregates. Network Agents. View the network agents.
Domains	View domains.
Projects	View, create, assign users to, remove users from, and delete projects.
Users	View, create, enable, disable, and delete users.
Groups	View, create, enable, disable, and delete groups.
Roles	View, create, enable, disable, and delete roles.

Upload and manage images

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

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



Note

You can also use the **glance** and **nova** command-line clients or the Image Service and Compute APIs to manage images. See [the section called "Manage images" \[33\]](#).

Upload an image

Follow this procedure to upload an image to a project.

1. Log in to the dashboard.
2. From the **CURRENT PROJECT** on the **Project** tab, select the appropriate project.
3. On the **Project** tab, click **Images**.
4. Click **Create Image**.

The Create An Image dialog box appears.

5. Enter the following values:

Name	Enter a name for the image.
Description	Optionally, enter a brief description of the image.
Image Source	Choose the image source from the list. Your choices are Image Location and Image File .
Image File or Image Location	Based on your selection for Image Source , you either enter the location URL of the image in the Image Location field. or browse to the image file on your system and add it.
Format	Select the correct format (for example, QCOW2) for the image.
Architecture	Specify the architecture. For example, <code>i386</code> for a 32-bit architecture or <code>x86-64</code> for a 64-bit architecture.
Minimum Disk (GB) and Minimum RAM (MB)	Leave these optional fields empty.
Public	Select this check box to make the image public to all users with access to the current project.
Protected	Select this check box to ensure that only users with permissions can delete the image.

6. Click **Create Image**.

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

Update an image

Follow this procedure to update an existing image.

1. Log in to the dashboard.
2. From the **CURRENT PROJECT** on the **Project** tab, select the appropriate project.
3. On the **Project** tab, click **Images**.
4. Select the image that you want to edit.
5. In the **Actions** column, click **More** and then select **Edit** from the list.
6. In the Update Image dialog box, you can perform the following actions:
 - Change the name of the image.
 - Select the **Public** check box to make the image public.
 - Clear the **Public** check box to make the image private.
7. Click **Update Image**.

Delete an image

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

1. Log in to the dashboard.
2. From the **CURRENT PROJECT** on the **Project** tab, select the appropriate project.
3. On the **Project** tab, click **Images**.
4. Select the images that you want to delete.
5. Click **Delete Images**.
6. In the **Confirm Delete Image** dialog box, click **Delete Images** to confirm the deletion.

Configure access and security for instances

Before you launch an instance, you should add security group rules to enable users to ping and use SSH to connect to the instance. To do so, you either [add rules to the default security group](#) or add a security group with rules.

Key pairs are SSH credentials that are injected into an instance when it is launched. To use key pair injection, the image that the instance is based on must contain the `cloud-init` package. Each project should have at least one key pair. For more information, see [the section called "Add a key pair" \[9\]](#).

If you have generated a key pair with an external tool, you can import it into OpenStack. The key pair can be used for multiple instances that belong to a project. For more information, see [the section called "Import a key pair" \[9\]](#).

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

Add a rule to the default security group

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

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

1. Log in to the dashboard, choose a project, and click **Access & Security**. The **Security Groups** tab shows the security groups that are available for this project.
2. Select the **default** security group and click **Edit Rules**.
3. To allow SSH access, click **Add Rule**.
4. In the Add Rule dialog box, enter the following values:

Rule	SSH
Remote	CIDR
CIDR	0.0.0.0/0



Note

To accept requests from a particular range of IP addresses, specify the IP address block in the **CIDR** box.

5. Click **Add**.

Instances will now have SSH port 22 open for requests from any IP address.

- To add an ICMP rule, click **Add Rule**.
- In the Add Rule dialog box, enter the following values:

Rule	All ICMP
Direction	Ingress
Remote	CIDR
CIDR	0.0.0.0/0

- Click **Add**.

Instances will now accept all incoming ICMP packets.

Add a key pair

Create at least one key pair for each project.

- Log in to the dashboard, choose a project, and click **Access & Security**.
- Click the **Keypairs** tab, which shows the key pairs that are available for this project.
- Click **Create Keypair**.
- In the Create Keypair dialog box, enter a name for your key pair, and click **Create Keypair**.
- Respond to the prompt to download the key pair.

Import a key pair

- Log in to the dashboard, choose a project, and click **Access & Security**.
- Click the **Keypairs** tab, which shows the key pairs that are available for this project.
- Click **Import Keypair**.
- In the Import Keypair dialog box, enter the name of your key pair, copy the public key into the **Public Key** box, and then click **Import Keypair**.
- Save the *.pem file locally.
- To change its permissions so that only you can read and write to the file, run the following command:

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



Note

If you are using the dashboard from a Windows computer, use PuTTYgen to load the *.pem file and convert and save it as *.ppk. For more information see the [WinSCP web page for PuTTYgen](#).

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

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

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

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

Allocate a floating IP address to an instance

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

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

1. Log in to the dashboard, choose a project, and click **Access & Security**.
2. Click the **Floating IPs** tab, which shows the floating IP addresses allocated to instances.
3. Click **Allocate IP to Project**.
4. Choose the pool from which to pick the IP address.
5. Click **Allocate IP**.
6. In the **Floating IPs** list, click **Associate**.
7. In the Manage Floating IP Associations dialog box, choose the following options:
 - The **IP Address** field is filled automatically, but you can add a new IP address by clicking the + button.
 - In the **Ports to be associated** field, select a port from the list.

The list shows all the instances with their fixed IP addresses.

8. Click **Associate**.



Note

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

To release the floating IP address back into the pool of addresses, click the **More** button and select the **Release Floating IP** option.

Launch and manage instances

Instances are virtual machines that run inside the cloud.

You can [launch an instance](#) from the following sources:

- Images uploaded to the OpenStack Image Service, as described in [the section called “Upload and manage images” \[6\]](#).
- Image that you have copied to a persistent volume. The instance launches from the volume, which is provided by the `cinder-volume` API through iSCSI.

Launch an instance

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

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

- To select the volume to from which to launch, launch an instance from an arbitrary image on the volume. The image that you select does not boot. Instead, it is replaced by the image on the volume that you choose in the next steps.

To boot a Xen image from a volume, the image you launch in must be the same type, fully virtualized or paravirtualized, as the one on the volume.


- Select the volume or volume snapshot from which to boot. Enter a device name. Enter `vda` for KVM images or `xvda` for Xen images.

1. Log in to the dashboard, choose a project, and click **Images & Snapshot**.

The dashboard shows the images that have been uploaded to OpenStack Image Service and are available for this project.

For details on creating images, see [Creating images manually](#) in the *OpenStack Virtual Machine Image Guide*.

2. Select an image and click **Launch**.
3. In the Launch Instance dialog box, specify the following values:

Details tab	
Availability Zone	By default, this value is set to the availability zone given by the cloud provider (for example, <code>us-west</code> or <code>apac-south</code>). For most cases, it could be <code>nova</code> .
Instance Name	Assign a name to the virtual machine. <div style="margin-top: 10px;">  <p>Note</p> <p>The name you assign here becomes the initial host name of the server. After the server is built, if you change the server name in the API or change the host name directly, the names are not updated in the dashboard.</p> <p>Server names are not guaranteed to be unique when created so you could have two instances with the same host name.</p> </div>
Flavor	Specify the size of the instance to launch.
Instance Count	To launch multiple instances, enter a value greater than 1. The default is 1.
Instance Boot Source	Your options are: <ul style="list-style-type: none"> • Boot from image—If you choose this option, a new field for Image Name displays. You can select the image from the list. • Boot from snapshot—If you choose this option, a new field for Instance Snapshot displays. You can select the snapshot from the list.

Details tab	
	<ul style="list-style-type: none"> • Boot from volume—If you choose this option, a new field for Volume displays. You can select the volume from the list. • Boot from image (creates a new volume) —With this option, you can boot from an image and create a volume by entering the Device Size and Device Name for your volume. • Boot from volume snapshot (creates a new volume)— Using this option, you can boot from a volume snapshot and create a new volume by choosing Volume Snapshot from a list and adding a Device Name for your volume. <p>Since you are launching an instance from an image, Boot from image is chosen by default.</p>
Image Name	This field changes based on your previous selection. Since you have chosen to launch an instance using an image, the Image Name field displays. Select the image name from the dropdown list.
Access & Security tab	
Keypair	Specify a key pair. If the image uses a static root password or a static key set (neither is recommended), you do not need to provide a key pair to launch the instance.
Security Groups	Activate the security groups that you want to assign to the instance. Security groups are a kind of cloud firewall that define which incoming network traffic is forwarded to instances. For details, see the section called “Add a rule to the default security group” [8] . If you have not created any security groups, you can assign only the default security group to the instance.
Networking tab	
Selected Networks	To add a network to the instance, click the + in the Available Networks field.
Post-Creation tab	
Customization Script	Specify a customization script that runs after your instance launches.

4. Click **Launch**.

The instance starts on a compute node in the cloud.

The **Instances** tab shows the instance's name, its private and public IP addresses, size, status, task, and power state.

If you did not provide a key pair, security groups, or rules, users can access the instance only from inside the cloud through VNC. Even pinging the instance is not possible without an ICMP rule configured. To access the instance through a VNC console, see [the section called “Access an instance through a console” \[60\]](#).

Connect to your instance by using SSH

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



Note

The user name is `ubuntu` for the Ubuntu cloud images on TryStack.

1. Copy the IP address for your instance.
2. Use the `ssh` command to make a secure connection to the instance. For example:

```
$ ssh -i MyKey.pem ubuntu@10.0.0.2
```

3. At the prompt, type `yes`.

Track usage for instances

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

1. Log in to the dashboard, choose a project, and click **Overview**.
2. To query the instance usage for a month, select a month and click **Submit**.
3. To download a summary, click **Download CSV Summary**.

Create an instance snapshot

1. Log in to the dashboard, choose a project, and click **Instances**.
2. Select the instance from which to create a snapshot.
3. From the **Actions** list, select **Create Snapshot**.
4. In the Create Snapshot dialog box, enter a name for the snapshot, and then click **Create Snapshot**.

The **Images & Snapshots** category shows the instance snapshot.

To launch an instance from the snapshot, select the snapshot and click **Launch**. Proceed with [the section called "Launch an instance" \[12\]](#).

Manage an instance

1. Log in to the dashboard, choose a project, and click **Instances**.
2. Select an instance.
3. In the **More** list in the **Actions** column, select the state.

You can resize or rebuild an instance. You can also choose to view the instance console log. Depending on the current state of the instance, you can pause, resume, suspend, soft or hard reboot, or terminate it.

Create and manage networks

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

Networking in OpenStack is complex. This section provides the basic instructions for creating a network and a router. For detailed information about managing networks, refer to the [OpenStack Cloud Administrator Guide](#).

Create a network

1. Log in to the dashboard, choose a project, and click **Networks**.
2. Click **Create Network**.
3. In the Create Network dialog box, specify the following values.

Network tab	
Network Name	Specify a name to identify the network.
Subnet tab	
Create Subnet	Select this check box to create a subnet You do not have to specify a subnet when you create a network, but if you do not, any attached instance receives an Error status.
Subnet Name	Specify a name for the subnet.
Network Address	Specify the IP address for the subnet.
IP Version	Select IPv4 or IPv6.
Gateway IP	Specify an IP address for a specific gateway. This parameter is optional.
Disable Gateway	Select this check box to disable a gateway IP address.
Subnet Detail tab	
Enable DHCP	Select this check box to enable DHCP.
Allocation Pools	Specify IP address pools.
DNS Name Servers	Specify a name for the DNS server.
Host Routes	Specify the IP address of host routes.

4. Click **Create**.

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

Create a router

1. Log in to the dashboard, choose a project, and click **Routers**.
2. Click **Create Router**.
3. In the Create Router dialog box, specify a name for the router and click **Create Router**.

The new router is now displayed in the **Routers** tab.

4. Click the new router's **Set Gateway** button.
5. In the **External Network** field, specify the network to which the router will connect, and then click **Set Gateway**.
6. To connect a private network to the newly created router, perform the following steps:
 - a. On the **Routers** tab, click the name of the router.
 - b. On the Router Details page, click **Add Interface**.
 - c. In the Add Interface dialog box, specify the following information:

Subnet	Select a subnet.
IP Address (optional)	Enter the router interface IP address for the selected subnet. Note: If this value is not set, then by default, the first host IP address in the subnet is used by OpenStack Networking.

The **Router Name** and **Router ID** fields are automatically updated.

7. Click **Add Interface**.

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

Create and manage object containers

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

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

Create a container

1. Log in to the dashboard, choose a project, and click **Containers**.
2. Click **Create Container**.
3. In the **Create Container** dialog box, enter a name for the container, choose if you want the access to be **Private** or **Public**.
4. Click **Create Container**.

You have successfully created a container.



Note

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

Upload an object

1. Log in to the dashboard, choose a project, and click **Containers** in the **Object Store** tab.
2. Select the container in which you want to store your object.
3. Click **Upload Object**.

The **Upload Object To Container: <name>** dialog box is displayed. *<name>* is the name of the container to which you are uploading the object.

4. Enter a name for the object.
5. Browse to and select the file that you want to upload.
6. Click **Upload Object**.

You have successfully uploaded an object to the container.

Manage an object

Edit an object

1. Log in to the dashboard, choose a project, and click **Containers** in the **Object Store** tab.
2. Select the container in which you want to store your object.
3. Click **More** and choose **Edit** from the dropdown list.

The **Edit Object** dialog box is displayed.

4. Browse to and select the file that you want to upload.
5. Click **Update Object**.



Note

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

Copy an object from one container to another

1. Log in to the dashboard, choose a project, and click **Containers** in the **Object Store** tab.
2. Select the container in which you want to store your object.
3. Click **More** and choose **Copy** from the dropdown list.
4. In the **Copy Object: launch** dialog box, enter or select the following values:

Destination Container	Choose the destination container from the dropdown list.
Path	Specify a path at which the new copy should be stored inside of the selected container.
Destination object name	Enter a name for the object in the new container.

5. Click **Copy Object**.

Create a metadata only object without a file

1. Log in to the dashboard, choose a project, and click **Containers** in the **Object Store** tab.
2. Select the container in which you want to store your object.
3. Click **Upload Object**.

The **Upload Object To Container: <name>** dialog box is displayed. *<name>* is the name of the container to which you are uploading the object.

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

4. Enter a name for the object.
5. Click **Update Object**.

Create a pseudo-folder

1. Log in to the dashboard, choose a project, and click **Containers** in the **Object Store** tab.
2. Select the container in which you want to store your object.
3. Click **Create Pseudo-folder**.

The **Create pseudo-folder in container <name>** dialog box is displayed. *<name>* is the name of the container to which you are uploading the object.

4. Enter a name for the pseudo-folder.

Pseudo-folders behave similarly to folders in your desktop operating system, with the exception that they are virtual collections defined by a common prefix on the object's name.

A slash (/) character is used as the delimiter for pseudo-folders in the Object Store.

5. Click **Create**.

Manage volumes

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

Create a volume

1. Log in to the dashboard, choose a project, and click **Volumes**.
2. Click **Create Volume**.

In the dialog box that opens, enter or select the following values.

Volume Name	Specify a name for the volume.
Description	Optionally, provide a brief description for the volume.
Type	Leave this field blank.
Size (GB)	The size of the volume in gigabytes.
Volume Source	Select one of the following options: <ul style="list-style-type: none">• No source, empty volume: creates an empty volume. Note: An empty volume does not contain a file system or a partition table.• Image: creates a volume from an image. Select the image from the list.

3. Click **Create Volume**.

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

Attach a volume to an instance

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

1. Log in to the dashboard, choose a project, and click **Volumes**.
2. Select the volume to add to an instance and click **Edit Attachments**.
3. In the Manage Volume Attachments dialog box, select an instance.
4. Enter the name of the device from which the volume is accessible by the instance.



Note

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

5. Click **Attach Volume**.

The dashboard shows the instance to which the volume is now attached and the device name.

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

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

Detach a volume from an instance

1. Log in to the dashboard, choose a project, and click **Volumes**.
2. Select the volume and click **Edit Attachments**.
3. Click **Detach Volume** and confirm your changes.

A message indicates whether the action was successful.

Create a snapshot from a volume

1. Log in to the dashboard, choose a project, and click **Volumes**.
2. Select a volume from which to create a snapshot.
3. From the **More** list, select **Create Snapshot**.
4. In the dialog box that opens, enter a snapshot name and a brief description.
5. Confirm your changes.

The dashboard shows the new volume snapshot in **Images & Snapshots**.

Delete a volume

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

1. Log in to the dashboard, choose a project, and click **Volumes**.
2. Select the check boxes for the volumes that you want to delete.
3. Click **Delete Volumes** and confirm your choice.

A message indicates whether the action was successful.

Launch and manage stacks

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

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

The template languages are described in [the Template Guide](#) in the [Heat developer documentation](#).

Launch a stack

1. Log in to the dashboard, choose a project, and click **Stacks** in the **Orchestration** category on the **Projects** tab.
2. Click **Launch Stack**.
3. In the Select Template dialog box, choose the source of the template from the list.
4. Depending on the source that you selected, enter the URL, browse to the file location, or directly include the template.
5. In the Launch Stack dialog box, specify the following values.

Stack Name	Enter a name to identify the stack.
Creation Timeout (minutes)	Specify the number of minutes that can elapse before the launch of the stack times out.
Rollback On Failure	Select this check box if you want if you want the service to roll back changes if the stack fails to launch.
Password for user "demo"	Specify the password that the default user will use when the stack is created.
DBUsername	Specify the name of the database user.
LinuxDistribution	Specify the Linux distribution that will be used in the stack.
DBRootPassword	Specify the root password for the database.
KeyName	Specify the name of the key pair that will be used to log into the stack.
DBName	Specify the name of the database.
DBPassword	Specify the password for the database.
InstanceType	Specify the flavor for the instance.

6. Click **Launch** to create a stack.
7. The dashboard shows the stack on the **Stacks** tab.

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

Topology	The topology of the stack.
Overview	The parameters and details of the stack.
Resources	The resources used by the stack.
Events	The events related to the stack.

Delete a stack

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

1. Log in to the dashboard, choose a project, and click **Stacks**.
2. Select the stack that you want to delete.
3. Click **Delete Stack**.
4. In the confirmation dialog box, click **Delete Stack** to confirm the deletion.

2. OpenStack command-line clients

Table of Contents

Overview	25
Install the OpenStack command-line clients	27
Discover the version number for a client	30
Set environment variables using the OpenStack RC file	31
Manage images	33
Configure access and security for instances	39
Launch instances	43
Manage instances and hosts	54
Provide user data to instances	66
Use snapshots to migrate instances	67
Store metadata on a configuration drive	70
Create and manage networks	75
Manage objects and containers	80
Create and manage stacks	82
Measure cloud resources	85
Manage volumes	88

Overview

You can use the OpenStack command-line clients to run simple commands that make API calls. You can run these commands from the command line or in scripts to automate tasks. If you provide OpenStack credentials, you can run these commands on any computer.

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

These open-source Python clients run on Linux or Mac OS X systems and are easy to learn and use. Each OpenStack service has its own command-line client. On some client commands, you can specify a **debug** parameter to show the underlying API request for the command. This is a good way to become familiar with the OpenStack API calls.

The following table lists the command-line client for each OpenStack service with its package name and description.

Table 2.1. OpenStack services and clients

Service	Client	Package	Description
Block Storage	cinder	python-cinderclient	Create and manage volumes.
Compute	nova	python-novaclient	Create and manage images, instances, and flavors.
Database Service	trove	python-troveclient	Create and manage databases.
Identity	keystone	python-keystoneclient	Create and manage users, tenants, roles, endpoints, and credentials.

Service	Client	Package	Description
Image Service	glance	python-glanceclient	Create and manage images.
Networking	neutron	python-neutronclient	Configure networks for guest servers. This client was previously called quantum .
Object Storage	swift	python-swiftclient	Gather statistics, list items, update metadata, and upload, download, and delete files stored by the Object Storage service. Gain access to an Object Storage installation for ad hoc processing.
Orchestration	heat	python-heatclient	Launch stacks from templates, view details of running stacks including events and resources, and update and delete stacks.
Telemetry	ceilometer	python-ceilometerclient	Create and collect measurements across OpenStack.

An OpenStack **common** client is in development.

Install the OpenStack command-line clients

Install the prerequisite software and the Python package for each OpenStack client.

Install the prerequisite software

The following table lists the software that you need to have to run the command-line clients, and provides installation instructions as needed.

Table 2.2. Prerequisite software

Prerequisite	Description
Python 2.6 or later	Currently, the clients do not support Python 3.
setuptools package	<p>Installed by default on Mac OS X.</p> <p>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.</p> <p>The recommended way to install setuptools on Microsoft Windows is to follow the documentation provided on the setuptools website. Another option is to use the unofficial binary installer maintained by Christoph Gohlke (http://www.lfd.uci.edu/~gohlke/pythonlibs/#setuptools).</p>
pip package	<p>To install the clients on a Linux, Mac OS X, or Microsoft Windows system, use pip. It is easy to use, ensures that you get the latest version of the clients from the Python Package Index, and lets you update or remove the packages later on.</p> <p>Install pip through the package manager for your system:</p> <p>MacOS.</p> <pre># easy_install pip</pre> <p>Microsoft Windows. Ensure that the <code>C:\Python27\Scripts</code> directory is defined in the <code>PATH</code> environment variable, and use the <code>easy_install</code> command from the setuptools package:</p> <pre>C:\>easy_install pip</pre> <p>Another option is to use the unofficial binary installer provided by Christoph Gohlke (http://www.lfd.uci.edu/~gohlke/pythonlibs/#pip).</p> <p>Ubuntu 12.04/14.04. A packaged version enables you to use <code>dpkg</code> or <code>aptitude</code> to install the <code>python-novaclient</code>:</p> <pre># aptitude install python-novaclient</pre> <p>Ubuntu and Debian.</p> <pre># aptitude install python-pip</pre> <p>Red Hat Enterprise Linux, CentOS, or Fedora. A packaged version available in RDO enables you to use <code>yum</code> to install the clients, or you can install pip and use it to manage client installation:</p> <pre># yum install python-pip</pre> <p>openSUSE 12.2 and earlier. A packaged version available in the Open Build Service enables you to use <code>rpm</code> or <code>zypper</code> to install the clients, or you can install pip and use it to manage client installation:</p> <pre># zypper install python-pip</pre>

Prerequisite	Description
	openSUSE 12.3 and later. A packaged version enables you to use rpm or zypper to install the clients. See the section called "Install the clients" [28]

Install the clients

When following the instructions in this section, replace *PROJECT* with the lowercase name of the client to install, such as **nova**. Repeat for each client. The following values are valid:

- `ceilometer` - Telemetry API
- `cinder` - Block Storage API and extensions
- `glance` - Image Service API
- `heat` - Orchestration API
- `keystone` - Identity service API and extensions
- `neutron` - Networking API
- `nova` - Compute API and extensions
- `swift` - Object Storage API
- `trove` - Database Service API

The following example shows the command for installing the nova client with *pip*.

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

Installing with pip

Use *pip* to install the OpenStack clients on a Linux, Mac OS X, or Microsoft Windows system. It is easy to use and ensures that you get the latest version of the client from the [Python Package Index](#). Also, *pip* enables you to update or remove a package.

Install each client separately by using the following command:

- For Mac OS X or Linux:

```
# pip install python-PROJECTclient
```

- For Microsoft Windows:

```
C:\>pip install python-PROJECTclient
```

Installing from packages

RDO and openSUSE have client packages that can be installed without *pip*.

On Red Hat Enterprise Linux, CentOS, or Fedora, use **yum** to install the clients from the packaged versions available in [RDO](#):

```
# yum install python-PROJECTclient
```

For openSUSE, use rpm or zypper to install the clients from the packaged versions available in [the Open Build Service](#):

```
# zypper install python-PROJECT
```

Upgrade or remove clients

To upgrade a client, add the `--upgrade` option to the `pip install` command:

```
# pip install --upgrade python-PROJECTclient
```

To remove the a client, run the `pip uninstall` command:

```
# pip uninstall python-PROJECTclient
```

What's next

Before you can run client commands, you must create and source the `PROJECT-openrc.sh` file to set environment variables. See [the section called "Set environment variables using the OpenStack RC file" \[31\]](#).

Discover the version number for a client

Run the following command to discover the version number for a client:

```
$ PROJECT --version
```

For example, to see the version number for the **nova** client, run the following command:

```
$ nova --version
```

The version number (2.15.0 in the example) is returned.

```
2.15.0
```

Set environment variables using the OpenStack RC file

To set the required environment variables for the OpenStack command-line clients, you must create an environment file called an OpenStack rc file, or `openrc.sh` file. If your OpenStack installation provides it, you can download the file from the OpenStack dashboard as an administrative user or any other user. This project-specific environment file contains the credentials that all OpenStack services use.

When you source the file, environment variables are set for your current shell. The variables enable the OpenStack client commands to communicate with the OpenStack services that run in the cloud.



Note

Defining environment variables using an environment file is not a common practice on Microsoft Windows. Environment variables are usually defined in the **Advanced** tab of the System Properties dialog box.

Download and source the OpenStack RC file

1. Log in to the OpenStack dashboard, choose the project for which you want to download the OpenStack RC file, and click **Access & Security**.
2. On the API Access tab, click **Download OpenStack RC File** and save the file. The filename will be of the form `PROJECT-openrc.sh` where `PROJECT` is the name of the project for which you downloaded the file.
3. Copy the `PROJECT-openrc.sh` file to the computer from which you want to run OpenStack commands.

For example, copy the file to the computer from which you want to upload an image with a **glance** client command.

4. On any shell from which you want to run OpenStack commands, source the `PROJECT-openrc.sh` file for the respective project.

In the following example, the `demo-openrc.sh` file is sourced for the demo project:

```
$ source demo-openrc.sh
```

5. When you are prompted for an OpenStack password, enter the password for the user who downloaded the `PROJECT-openrc.sh` file.

Create and source the OpenStack RC file

Alternatively, you can create the `PROJECT-openrc.sh` file from scratch, if for some reason you cannot download the file from the dashboard.

1. In a text editor, create a file named `PROJECT-openrc.sh` file and add the following authentication information:

```
export OS_USERNAME=username
export OS_PASSWORD=password
export OS_TENANT_NAME=projectName
export OS_AUTH_URL=https://identityHost:portNumber/v2.0
# The following lines can be omitted
export OS_TENANT_ID=tenantIDString
export OS_REGION_NAME=regionName
```

The following example shows the information for a project called `admin`, where the OS username is also `admin`, and the identity host is located at `controller`.

2. On any shell from which you want to run OpenStack commands, source the `PROJECT-openrc.sh` file for the respective project. In this example, you source the `admin-openrc.sh` file for the `admin` project:

```
$ source admin-openrc.sh
```



Note

You are not prompted for the password with this method. The password lives in clear text format in the `PROJECT-openrc.sh` file. Restrict the permissions on this file to avoid security problems. You can also remove the `OS_PASSWORD` variable from the file, and use the `--password` parameter with OpenStack client commands instead.

Override environment variable values

When you run OpenStack client commands, you can override some environment variable settings by using the options that are listed at the end of the **help** output of the various client commands. For example, you can override the `OS_PASSWORD` setting in the `PROJECT-openrc.sh` file by specifying a password on a **keystone** command, as follows:

```
$ keystone --os-password PASSWORD service-list
```

Where `PASSWORD` is your password.

Manage images

The cloud operator assigns roles to users. Roles determine who can upload and manage images. The operator might restrict image upload and management to only cloud administrators or operators.

You can upload images through the **glance** client or the Image Service API. You can also use the **nova** client to list images, set and delete image metadata, delete images, and take a snapshot of a running instance to create an image. After you upload an image, you cannot change it.

For details about image creation, see the [Virtual Machine Image Guide](#).

List or get details for images (glance)

To get a list of images and to then get further details about a single image, use **glance image-list** and **glance image-show**.

```
$ glance image-list
```

ID	Disk Format	Container Format	Format	Size	Name	Status	Owner
397e713c-b95b-4186-ad46-6126863ea0a9	ami			25165824	cirros-0.3.2-x86_64-uec	active	
df430cc2-3406-4061-b635-a51c16e488ac	ami			4955792	cirros-0.3.2-x86_64-uec-kernel	active	
3cf852bd-2332-48f4-9ae4-7d926d50945e	ari			3714968	cirros-0.3.2-x86_64-uec-ramdisk	active	
7e5142af-1253-4634-bcc6-89482c5f2e8a	ami			14221312	myCirrosImage	active	

```
$ glance image-show myCirrosImage
```

Property	Value
Property 'base_image_ref'	397e713c-b95b-4186-ad46-6126863ea0a9
Property 'image_location'	snapshot
Property 'image_state'	available
Property 'image_type'	snapshot
Property 'instance_type_ephemeral_gb'	0
Property 'instance_type_flavorid'	2
Property 'instance_type_id'	5
Property 'instance_type_memory_mb'	2048
Property 'instance_type_name'	m1.small
Property 'instance_type_root_gb'	20
Property 'instance_type_rxtx_factor'	1
Property 'instance_type_swap'	0
Property 'instance_type_vcpu_weight'	None
Property 'instance_type_vcpus'	1
Property 'instance_uuid'	84c6e57d-a6b1-44b6-81eb-fcb36afd31b5
Property 'kernel_id'	df430cc2-3406-4061-b635-a51c16e488ac
Property 'owner_id'	66265572db174a7aa66eba661f58eb9e
Property 'ramdisk_id'	3cf852bd-2332-48f4-9ae4-7d926d50945e
Property 'user_id'	376744b5910b4b4da7d8e6cb483b06a8
checksum	8e4838effa1969ad591655d6485c7ba8
container_format	ami
created_at	2013-07-22T19:45:58
deleted	False
disk_format	ami
id	7e5142af-1253-4634-bcc6-89482c5f2e8a
is_public	False
min_disk	0
min_ram	0
name	myCirrosImage
owner	66265572db174a7aa66eba661f58eb9e
protected	False
size	14221312
status	active
updated_at	2013-07-22T19:46:42

When viewing a list of images, you can also use **grep** to filter the list, as follows:

```
$ glance image-list | grep 'cirros'
| 397e713c-b95b-4186-ad46-6126863ea0a9 | cirros-0.3.2-x86_64-uec | ami
| | ami | 25165824 | active |
| df430cc2-3406-4061-b635-a51c16e488ac | cirros-0.3.2-x86_64-uec-kernel | aki
| | aki | 4955792 | active |
| 3cf852bd-2332-48f4-9ae4-7d926d50945e | cirros-0.3.2-x86_64-uec-ramdisk | ari
| | ari | 3714968 | active |
```



Note

To store location metadata for images, which enables direct file access for a client, update the `/etc/glance/glance.conf` file with the following statements:

- `show_multiple_locations = True`
- `filesystem_store_metadata_file = filePath`, where `filePath` points to a JSON file that defines the mount point for OpenStack images on your system and a unique ID. For example:

```
[{
  "id": "2d9bb53f-70ea-4066-a68b-67960eaae673",
  "mountpoint": "/var/lib/glance/images/"
}]
```

After you restart the Image Service, you can use the following syntax to view the image's location information:

```
$ glance --os-image-api-version=2 image-show imageID
```

For example, using the image ID shown above, you would issue the command as follows:

```
$ glance --os-image-api-version=2 image-show 2d9bb53f-70ea-4066-a68b-67960eaae673
```

Create or update an image (glance)

To create an image, use **glance image-create**:

```
$ glance image-create imageName
```

To update an image by name or ID, use **glance image-update**:

```
$ glance image-update imageName
```

The following table lists the optional arguments that you can use with the **create** and **update** commands to modify image properties. For more information, refer to Image Service chapter in the [OpenStack Command-Line Interface Reference](#).

<code>--name NAME</code>	The name of the image.
<code>--disk-format DISK_FORMAT</code>	The disk format of the image. Acceptable formats are ami, ari, aki, vhd, vmdk, raw, qcow2, vdi, and iso.

<code>--container-format</code> <i>CONTAINER_FORMAT</i>	The container format of the image. Acceptable formats are ami, ari, aki, bare, and ovf.
<code>--owner</code> <i>TENANT_ID</i>	The tenant who should own the image.
<code>--size</code> <i>SIZE</i>	The size of image data, in bytes.
<code>--min-disk</code> <i>DISK_GB</i>	The minimum size of the disk needed to boot the image, in gigabytes.
<code>--min-ram</code> <i>DISK_RAM</i>	The minimum amount of RAM needed to boot the image, in megabytes.
<code>--location</code> <i>IMAGE_URL</i>	The URL where the data for this image resides. For example, if the image data is stored in swift, you could specify <code>swift://account:key@example.com/container/obj</code> .
<code>--file</code> <i>FILE</i>	Local file that contains the disk image to be uploaded during the update. Alternatively, you can pass images to the client through stdin.
<code>--checksum</code> <i>CHECKSUM</i>	Hash of image data to use for verification.
<code>--copy-from</code> <i>IMAGE_URL</i>	Similar to <code>--location</code> in usage, but indicates that the image server should immediately copy the data and store it in its configured image store.
<code>--is-public</code> [<i>True/False</i>]	Makes an image accessible for all the tenants.
<code>--is-protected</code> [<i>True/False</i>]	Prevents an image from being deleted.
<code>--property</code> <i>KEY=VALUE</i>	Arbitrary property to associate with image. This option can be used multiple times.
<code>--purge-props</code>	Deletes all image properties that are not explicitly set in the update request. Otherwise, those properties not referenced are preserved.
<code>--human-readable</code>	Prints the image size in a human-friendly format.

The following example shows the command that you would use to upload a CentOS 6.3 image in qcow2 format and configure it for public access:

```
$ glance image-create --name centos63-image --disk-format=qcow2 \
  --container-format=bare --is-public=True --file=./centos63.qcow2
```

The following example shows how to update an existing image with a properties that describe the disk bus, the CD-ROM bus, and the VIF model:

```
$ glance image-update \
  --property hw_disk_bus=scsi \
  --property hw_cdrom_bus=ide \
  --property hw_vif_model=e1000 \
  f16-x86_64-openstack-sda
```

Currently the libvirt virtualization tool determines the disk, CD-ROM, and VIF device models based on the configured hypervisor type (`libvirt_type` in `/etc/nova/nova.conf`). For the sake of optimal performance, libvirt defaults to using virtio for both disk and VIF (NIC) models. The disadvantage of this approach is that it is not possible to run operating systems that lack virtio drivers, for example, BSD, Solaris, and older versions of Linux and Windows.

If you specify a disk or CD-ROM bus model that is not supported, see [Table 2.3, “Disk and CD-ROM bus model values”](#) [36]. If you specify a VIF model that is not supported, the instance fails to launch. See [Table 2.4, “VIF model values”](#) [36].

ID	Name	Status	Task State	Power State	Networks
84c6e57d-a6b1-44b6-81eb-fcb36afd31b5	myCirrosServer	ACTIVE	None	Running	private=10.0.0.3

In this example, the server is named `myCirrosServer`.

- Use this server to create a snapshot:

```
$ nova image-create myCirrosServer myCirrosImage
```

The command creates a qemu snapshot and automatically uploads the image to your repository. Only the tenant that creates the image has access to it.

- Get details for your image to check its status:

```
$ nova image-show myCirrosImage
```

Property	Value
metadata owner_id	66265572db174a7aa66eba661f58eb9e
minDisk	0
metadata instance_type_name	m1.small
metadata instance_type_id	5
metadata instance_type_memory_mb	2048
id	7e5142af-1253-4634-bcc6-89482c5f2e8a
metadata instance_type_root_gb	20
metadata instance_type_rxtx_factor	1
metadata ramdisk_id	3cf852bd-2332-48f4-9ae4-7d926d50945e
metadata image_state	available
metadata image_location	snapshot
minRam	0
metadata instance_type_vcpus	1
status	ACTIVE
updated	2013-07-22T19:46:42Z
metadata instance_type_swap	0
metadata instance_type_vcpu_weight	None
metadata base_image_ref	397e713c-b95b-4186-ad46-6126863ea0a9
progress	100
metadata instance_type_flavorid	2
OS-EXT-IMG-SIZE:size	14221312
metadata image_type	snapshot
metadata user_id	376744b5910b4b4da7d8e6cb483b06a8
name	myCirrosImage
created	2013-07-22T19:45:58Z
metadata instance_uuid	84c6e57d-a6b1-44b6-81eb-fcb36afd31b5
server	84c6e57d-a6b1-44b6-81eb-fcb36afd31b5
metadata kernel_id	df430cc2-3406-4061-b635-a51c16e488ac
metadata instance_type_ephemeral_gb	0

The image status changes from `SAVING` to `ACTIVE`. Only the tenant who creates the image has access to it.

To launch an instance from your image, include the image ID and flavor ID, as in the following example:

```
$ nova boot newServer --image 7e5142af-1253-4634-bcc6-89482c5f2e8a \
--flavor 3
```

Property	Value
OS-EXT-STS:task_state	scheduling
image	myCirrosImage
OS-EXT-STS:vm_state	building
OS-EXT-SRV-ATTR:instance_name	instance-00000007
flavor	m1.medium
id	d7efd3e4-d375-46d1-9d57-372b6e4bdb7f
security_groups	[{'name': 'u'default'}]
user_id	376744b5910b4b4da7d8e6cb483b06a8
OS-DCF:diskConfig	MANUAL
accessIPv4	
accessIPv6	
progress	0
OS-EXT-STS:power_state	0
OS-EXT-AZ:availability_zone	nova
config_drive	
status	BUILD
updated	2013-07-22T19:58:33Z
hostId	
OS-EXT-SRV-ATTR:host	None
key_name	None
OS-EXT-SRV-ATTR:hypervisor_hostname	None
name	newServer
adminPass	jis88mN46RGP
tenant_id	66265572db174a7aa66eba661f58eb9e
created	2013-07-22T19:58:33Z
metadata	{}

Troubleshoot image creation

If you encounter problems in creating an image in Image Service or Compute, the following information may help you troubleshoot the creation process.

- You cannot create a snapshot from an instance that has an attached volume. Detach the volume, create the image, and re-mount the volume.
- Ensure that the version of qemu you are using is version 0.14 or later. Earlier versions of qemu result in an `unknown option -s` error message in the `nova-compute.log` file.
- Examine the `/var/log/nova-api.log` and `/var/log/nova-compute.log` log files for error messages.

Configure access and security for instances

When you launch a virtual machine, you can inject a *key pair*, which provides SSH access to your instance. For this to work, the image must contain the `cloud-init` package.

You create at least one key pair for each project. You can use the key pair for multiple instances that belong to that project. If you generate a key pair with an external tool, you can import it into OpenStack.

If an image uses a static root password or a static key set—neither is recommended—you must not provide a key pair when you launch the instance.

A *security group* is a named collection of network access rules that you use to limit the types of traffic that have access to instances. When you launch an instance, you can assign one or more security groups to it. If you do not create security groups, new instances are automatically assigned to the default security group, unless you explicitly specify a different security group.

The associated *rules* in each security group control the traffic to instances in the group. Any incoming traffic that is not matched by a rule is denied access by default. You can add rules to or remove rules from a security group, and you can modify rules for the default and any other security group.

You can modify the rules in a security group to allow access to instances through different ports and protocols. For example, you can modify rules to allow access to instances through SSH, to ping instances, or to allow UDP traffic; for example, for a DNS server running on an instance. You specify the following parameters for rules:

- **Source of traffic.** Enable traffic to instances from either IP addresses inside the cloud from other group members or from all IP addresses.
- **Protocol.** Choose TCP for SSH, ICMP for pings, or UDP.
- **Destination port on virtual machine.** Define a port range. To open a single port only, enter the same value twice. ICMP does not support ports; instead, you enter values to define the codes and types of ICMP traffic to be allowed.

Rules are automatically enforced as soon as you create or modify them.



Note

Instances that use the default security group cannot, by default, be accessed from any IP address outside of the cloud. If you want those IP addresses to access the instances, you must modify the rules for the default security group.

You can also assign a floating IP address to a running instance to make it accessible from outside the cloud. See [the section called “Manage IP addresses” \[54\]](#).

Add a key pair

You can generate a key pair or upload an existing public key.

1. To generate a key pair, run the following command:

```
$ nova keypair-add KEY_NAME > MY_KEY.pem
```

The command generates a key pair with the name that you specify for *KEY_NAME*, writes the private key to the *.pem* file that you specify, and registers the public key at the Nova database.

2. To set the permissions of the *.pem* file so that only you can read and write to it, run the following command:

```
$ chmod 600 MY_KEY.pem
```

Import a key pair

1. If you have already generated a key pair and the public key is located at *~/ .ssh/ id_rsa.pub*, run the following command to upload the public key:

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

The command registers the public key at the Nova database and names the key pair the name that you specify for *KEY_NAME*.

2. To ensure that the key pair has been successfully imported, list key pairs as follows:

```
$ nova keypair-list
```

Create and manage security groups

1. To list the security groups for the current project, including descriptions, enter the following command:

```
$ nova secgroup-list
```

2. To create a security group with a specified name and description, enter the following command:

```
$ nova secgroup-create SECURITY_GROUP_NAME GROUP_DESCRIPTION
```

3. To delete a specified group, enter the following command:

```
$ nova secgroup-delete SECURITY_GROUP_NAME
```



Note

You cannot delete the default security group for a project. Also, you cannot delete a security group that is assigned to a running instance.

Create and manage security group rules

Modify security group rules with the **nova secgroup-*-rule** commands. Before you begin, source the OpenStack RC file. For details, see [the section called “Set environment variables using the OpenStack RC file” \[31\]](#).

1. To list the rules for a security group, run the following command:

```
$ nova secgroup-list-rules SECURITY_GROUP_NAME
```

2. To allow SSH access to the instances, choose one of the following options:

- Allow access from all IP addresses, specified as IP subnet 0.0.0.0/0 in CIDR notation:

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

- Allow access only from IP addresses from other security groups (source groups) to access the specified port:

```
$ nova secgroup-add-group-rule --ip_proto tcp --from_port 22 \  
--to_port 22 SECURITY_GROUP_NAME SOURCE_GROUP_NAME
```

3. To allow ping of the instances, choose one of the following options:

- Allow ping from all IP addresses, specified as IP subnet 0.0.0.0/0 in CIDR notation:

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

This allows access to all codes and all types of ICMP traffic.

- Allow only members of other security groups (source groups) to ping instances:

```
$ nova secgroup-add-group-rule --ip_proto icmp --from_port -1 \  
--to_port -1 SECURITY_GROUP_NAME SOURCE_GROUP_NAME
```

4. To allow access through a UDP port, such as allowing access to a DNS server that runs on a VM, choose one of the following options:

- Allow UDP access from IP addresses, specified as IP subnet 0.0.0.0/0 in CIDR notation:

```
$ nova secgroup-add-rule SECURITY_GROUP_NAME udp 53 53 0.0.0.0/0
```

- Allow only IP addresses from other security groups (source groups) to access the specified port:

```
$ nova secgroup-add-group-rule --ip_proto udp --from_port 53 \  
--to_port 53 SECURITY_GROUP_NAME SOURCE_GROUP_NAME
```

Delete a security group

To delete a security group rule, specify the same arguments that you used to create the rule.

For example, to delete the security group rule that permits SSH access from all IP addresses, run the following command.

```
$ nova secgroup-delete-rule SECURITY_GROUP_NAME tcp 22 22 0.0.0.0/0
```

Launch instances

Instances are virtual machines that run inside the cloud.

Before you can launch an instance, gather the following parameters:

- The **instance source**. This can be an image, a snapshot, or a block storage volume that contains an image or snapshot.
- A **name** for your instance.
- The **flavor** for your instance, which defines the compute, memory, and storage capacity of nova computing instances. A flavor is an available hardware configuration for a server. It defines the "size" of a virtual server that can be launched.
- Any **user data** files: A user data file is a special key in the metadata service that holds a file that cloud-aware applications in the guest instance can access. For example, one application that uses user data is the [cloud-init](#) system, which is an open-source package from Ubuntu that is available on various Linux distributions and which handles early initialization of a cloud instance.
- Access and security credentials, which include one or both of the following credentials:
 - A **key pair** for your instance, which are SSH credentials that are injected into images when they are launched. For the key pair to be successfully injected, the image must contain the `cloud-init` package. Create at least one key pair for each project. If you already have generated a key pair with an external tool, you can import it into OpenStack. You can use the key pair for multiple instances that belong to that project.
 - A **security group**, which defines which incoming network traffic is forwarded to instances. Security groups hold a set of firewall policies, known as *security group rules*.
- If needed, you can assign a **floating (public) IP address** to a running instance.
- You can also attach a block storage device, or **volume**, for persistent storage.



Note

Instances that use the default security group cannot, by default, be accessed from any IP address outside of the cloud. If you want those IP addresses to access the instances, you must modify the rules for the default security group.

You can also assign a floating IP address to a running instance to make it accessible from outside the cloud. See [the section called "Manage IP addresses" \[54\]](#).

After you gather the parameters that you need to launch an instance, you can launch it from an [image](#) or a [volume](#). You can launch an instance directly from one of the available OpenStack images or from an image that you have copied to a persistent volume. The OpenStack Image Service provides a pool of images that are accessible to members of different projects.

Gather parameters to launch an instance

Before you begin, source the OpenStack RC file.

1. List the available flavors and note the ID of the flavor that you want to use for your instance.

```
$ nova flavor-list
```

ID	Name	Memory_MB	Disk	Ephemeral	Swap	VCPUs	RXTX_Factor	Is_Public
1	ml.tiny	512	0	0		1	1.0	True
2	ml.small	2048	20	0		1	1.0	True
3	ml.medium	4096	40	0		2	1.0	True
4	ml.large	8192	80	0		4	1.0	True
5	ml.xlarge	16384	160	0		8	1.0	True

2. List the available images and note the ID of the image from which you want to boot your instance.

```
$ nova image-list
```

ID	Name	Status	Server
397e713c-b95b-4186-ad46-6126863ea0a9	cirros-0.3.2-x86_64-uec	ACTIVE	
df430cc2-3406-4061-b635-a51c16e488ac	cirros-0.3.2-x86_64-uec-kernel	ACTIVE	
3cf852bd-2332-48f4-9ae4-7d926d50945e	cirros-0.3.2-x86_64-uec-ramdisk	ACTIVE	

You can also filter the image list by using **grep** to find a specific image, as follows:

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

```
df430cc2-3406-4061-b635-a51c16e488ac | cirros-0.3.2-x86_64-uec-kernel | ACTIVE |
```

3. List the available security groups and note the ID of the security group that you want to use for your instance.



Note

If you are an admin user, specify the `--all-tenants` parameter to list groups for all tenants.

```
$ nova secgroup-list --all-tenants
```

Id	Name	Description	Tenant_ID
2	default	default	66265572db174a7aa66eba661f58eb9e
1	default	default	b70d90d65e464582b6b2161cf3603ced

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

You can view rules for a specified security group:

```
$ nova secgroup-list-rules default
```

4. List the available key pairs and note the name of the key pair that you use for SSH access.

```
$ nova keypair-list
```


Launch an instance from an image

1. After you have all the parameters required to launch an instance, run the following command and specify the server name, flavor ID, and image ID. Optionally, you can provide a key name for access control and a security group for security. You can also include metadata key and value pairs. For example, you can add a description for your server by providing the `--meta description="My Server"` parameter.

You can pass user data in a local file at instance launch by using the `--user-data USER-DATA-FILE` parameter.

```
$ nova boot --flavor FLAVOR_ID --image IMAGE_ID --key-name KEY_NAME \
--user-data USER_DATA_FILE --security-groups SEC_GROUP --meta KEY=VALUE \
INSTANCE_NAME
```

The following example shows a the command for launching an instance called `MyCirrosServer` with the `m1.small` flavor (ID of 1), `cirros-0.3.2-x86_64-uec` image (ID of `397e713c-b95b-4186-ad46-6126863ea0a9`), the default security group, the `KeyPair01` key, and a user data file called `cloudinit.file`.

```
$ nova boot --flavor 1 --image 397e713c-b95b-4186-ad46-6126863ea0a9 \
--security-groups default --key-name KeyPair01 --user-data cloudinit.
file \
myCirrosServer
```

Depending on the parameters that you provide, the command returns a list of server properties.

A status of `BUILD` indicates that the instance has started, but is not yet online.

A status of `ACTIVE` indicates that the instance is active.

Property	Value
OS-EXT-STS:task_state	scheduling
image	cirros-0.3.2-x86_64-uec
OS-EXT-STS:vm_state	building
OS-EXT-SRV-ATTR:instance_name	instance-00000002
flavor	m1.small
id	b3cdc6c0-85a7-4904-ae85-71918f734048
security_groups	[{'u'name': 'u'default'}]
user_id	376744b5910b4b4da7d8e6cb483b06a8
OS-DCF:diskConfig	MANUAL
accessIPv4	
accessIPv6	
progress	0
OS-EXT-STS:power_state	0
OS-EXT-AZ:availability_zone	nova
config_drive	
status	BUILD
updated	2013-07-16T16:25:34Z
hostId	
OS-EXT-SRV-ATTR:host	None
key_name	None
OS-EXT-SRV-ATTR:hypervisor_hostname	None
name	myCirrosServer
adminPass	tVs5pL8HcPGw
tenant_id	66265572db174a7aa66eba661f58eb9e
created	2013-07-16T16:25:34Z
metadata	{'u'KEY': 'u'VALUE'}

Copy the server ID value from the `id` field in the output. You use this ID to get details for or delete your server.

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



Note

You can also place arbitrary local files into the instance file system at creation time by using the `--file <dst-path=src-path>` option. You can store up to five 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, you can use the `--file` option as shown in the following example:

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

2. Check if the instance is online:

```
$ nova list
```

The list shows the ID, name, status, and private (and if assigned, public) IP addresses for all instances in the project to which you belong:

ID	Name	Status	Task State	Power State	Networks
84c6e57d-a6b1-44b6-81eb-fcb36afd31b5	myCirrosServer	ACTIVE	None	Running	private=10.0.0.3
8a99547e-7385-4ad1-ae50-4ecfaad5f42	myInstanceFromVolume	ACTIVE	None	Running	private=10.0.0.4

If the status for the instance is `ACTIVE`, the instance is online.

To view the available options for the `nova list` command, run the following command:

```
$ nova help list
```



Note

If you did not provide a key pair, security groups, or rules, you can access the instance only from inside the cloud through VNC. Even pinging the instance is not possible.

Launch an instance from a volume

You can boot instances from a volume instead of an image. Use the `nova boot --block-device` parameter to define how volumes are attached to an instance when you create it. You can use the `--block-device` parameter with existing or new volumes that you create from a source image, volume, or snapshot.



Note

To attach a volume to a running instance, see [Manage volumes](#).

Create volume from image and boot instance

Use this procedure to create a volume from an image, and use it to boot an instance.

1. You can create a volume from an existing image, volume, or snapshot.

List available images:

```
$ nova image-list
+-----+-----+
| ID                                     | Name                               |
| Status | Server |                                     |
+-----+-----+
| e0b7734d-2331-42a3-b19e-067adc0da17d | cirros-0.3.2-x86_64-uec           | |
| ACTIVE |         |                                     |
| 75bf193b-237b-435e-8712-896c51484de9 | cirros-0.3.2-x86_64-uec-kernel    |
| ACTIVE |         |                                     |
| 19eee81c-f972-44e1-a952-1dceee148c47 | cirros-0.3.2-x86_64-uec-ramdisk   |
| ACTIVE |         |                                     |
+-----+-----+
```

2. To create a bootable volume from an image and launch an instance from this volume, use the `--block-device` parameter.

For example:

```
$ nova boot --flavor FLAVOR --block-device source=SOURCE,id=ID,dest=DEST,
size=SIZE,shutdown=PRESERVE,bootindex=INDEX NAME
```

The parameters are:

Parameter	Description
<code>--flavor FLAVOR</code>	The flavor ID or name.
<code>--block-device source=SOURCE,id=ID,dest=DEST,size=SIZE,shutdown=PRESERVE,bootindex=INDEX</code>	<ul style="list-style-type: none"> • SOURCE: The type of object used to create the block device. Valid values are volume, snapshot, image and blank. • ID: The ID of the source object. • DEST: The type of the target virtual device. Valid values are volume and local. • SIZE: The size of the volume that will be created.

Parameter	Description
	<ul style="list-style-type: none"> <i>PRESERVE</i>: What to do with the volume when the instance is terminated. <i>preserve</i> will not delete the volume, <i>remove</i> will. <i>INDEX</i>: Used to order the boot disks. Use 0 to boot from this volume.
<i>NAME</i>	The name for the server.

3. Create a bootable volume from an image, before the instance boots. The volume is not deleted when the instance is terminated:

```
$ nova boot --flavor 2 \
  --block-device source=image,id=e0b7734d-2331-42a3-b19e-067adc0da17d,
  dest=volume,size=10,shutdown=preserve,bootindex=0 \
  myInstanceFromVolume
+-----+
+-----+
| Property          | Value
+-----+
| OS-EXT-STS:task_state | scheduling
| image             | Attempt to boot from volume - no
image supplied
| OS-EXT-STS:vm_state  | building
| OS-EXT-SRV-ATTR:instance_name | instance-00000003
| OS-SRV-USG:launched_at | None
| flavor            | m1.small
| id                | 2e65c854-dba9-4f68-8f08-
fe332e546ecc
| security_groups    | [{u'name': u'default'}]
| user_id           | 352b37f5c89144d4ad0534139266d51f
| OS-DCF:diskConfig  | MANUAL
| accessIPv4         |
| accessIPv6         |
| progress           | 0
| OS-EXT-STS:power_state | 0
| OS-EXT-AZ:availability_zone | nova
| config_drive       |
| status             | BUILD
| updated            | 2014-02-02T13:29:54Z
```

```

| hostId |
| OS-EXT-SRV-ATTR:host | None |
| OS-SRV-USG:terminated_at | None |
| key_name | None |
| OS-EXT-SRV-ATTR:hypervisor_hostname | None |
| name | myInstanceFromVolume |
| adminPass | TzjqyGsRcJo9 |
| tenant_id | f7ac731cc11f40efbc03a9f9e1d1d21f |
| created | 2014-02-02T13:29:53Z |
| os-extended-volumes:volumes_attached | [] |
| metadata | {} |
+-----+
+-----+

```

- List volumes to see the bootable volume and its attached `myInstanceFromVolume` instance:

```

$ cinder list
+-----+-----+-----+-----+-----+
| ID | Status | Display Name | Size |
| Volume Type | Bootable | Attached to | |
+-----+-----+-----+-----+
| 2fff50ab-1a9c-4d45-ae60-1d054d6bc868 | in-use | | 10 |
| None | true | 2e65c854-dba9-4f68-8f08-fe332e546ecc | |
+-----+-----+-----+-----+

```

Attach non-bootable volume to an instance

Use the `--block-device` parameter to attach an existing, non-bootable volume to a new instance.

- Create a volume:

```

$ cinder create --display-name my-volume 8
+-----+-----+
| Property | Value |
+-----+-----+
| attachments | [] |
| availability_zone | nova |
| bootable | false |
| created_at | 2014-02-04T21:25:18.730961 |
| display_description | None |
| display_name | my-volume |
| id | 3195a5a7-fd0d-4ac3-b919-7ba6cbe11d46 |
| metadata | {} |
+-----+-----+

```

size	8
snapshot_id	None
source_volid	None
status	creating
volume_type	None

2. List volumes:

```
$ cinder list
+-----+-----+-----+-----+
|          ID          | Status | Display Name | Size |
| Volume Type | Bootable | Attached to |      |
+-----+-----+-----+-----+
| 3195a5a7-fd0d-4ac3-b919-7ba6cbe11d46 | available | my-volume | 8 |
| None | false | | |
+-----+-----+-----+-----+
```



Note

The volume is not bootable because it was not created from an image.

The volume is also entirely empty: It has no partition table and no file system.

3. Run this command to create an instance and boot it with the volume that is attached to this instance. An image is used as boot source:

```
$ nova boot --flavor 2 --image e0b7734d-2331-42a3-b19e-067adc0da17d \
  --block-device source=volume,id=3195a5a7-fd0d-4ac3-
b919-7ba6cbe11d46,dest=volume,shutdown=preserve \
  myInstanceWithVolume
+-----+-----+
| Property          | Value |
+-----+-----+
| OS-EXT-STS:task_state | scheduling |
| image             | e0b7734d-2331-42a3-
b19e-067adc0da17d |
| OS-EXT-STS:vm_state | building |
| OS-EXT-SRV-ATTR:instance_name | instance-00000003 |
| flavor            | m1.small |
| id                | 8ed8b0f9-70de-4662-
a16c-0b51ce7b17b4 |
| security_groups    | [{u'name': u'default'}] |
| user_id           | 352b37f5c89144d4ad0534139266d51f |
| OS-DCF:diskConfig | MANUAL |
+-----+-----+
```

```

| accessIPv4          |
| accessIPv6          |
| progress            | 0
| OS-EXT-STS:power_state | 0
| OS-EXT-AZ:availability_zone | nova
| config_drive        |
| status              | BUILD
| updated              | 2013-10-16T01:43:26Z
| hostId              |
| OS-EXT-SRV-ATTR:host | None
| OS-SRV-USG:terminated_at | None
| key_name            | None
| OS-EXT-SRV-ATTR:hypervisor_hostname | None
| name                 | myInstanceWithVolume
| adminPass           | BULD33uzYwhq
| tenant_id           | f7ac731cc11f40efbc03a9f9e1d1d21f
| created              | 2013-10-16T01:43:25Z
| os-extended-volumes:volumes_attached | [{u'id': u'3195a5a7-fd0d-4ac3-b919-7ba6cbe11d46'}]
| metadata             | {}
+-----+
+-----+

```

4. List volumes:

```
$ nova volume-list
```

Note that the volume is attached to a server:

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ID                    | Status    | Display Name | Size |
| Volume Type | Attached to |              |      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| 3195a5a7-fd0d-4ac3-b919-7ba6cbe11d46 | in-use    | my-volume    | 8    |
| None                | 8ed8b0f9-70de-4662-a16c-0b51ce7b17b4 |              |      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

Attach swap or ephemeral disk to an instance

Use the **nova boot** `--swap` parameter to attach a swap disk on boot or the **nova boot** `--ephemeral` parameter to attach an ephemeral disk on boot. When you terminate the instance, both disks are deleted.

Boot an instance with a 512 MB swap disk and 2 GB ephemeral disk:

```
$ nova boot --flavor FLAVOR --image IMAGE_ID --swap 512 --ephemeral size=2 NAME
```



Note

The flavor defines the maximum swap and ephemeral disk size. You cannot exceed these maximum values.

Manage instances and hosts

Instances are virtual machines that run inside the cloud on physical compute nodes. The Compute service manages instances. A host is the node on which a group of instances resides.

This section describes how to perform the different tasks involved in instance management, such as adding floating IP addresses, stopping and starting instances, and terminating instances. This section also discusses node management tasks.

Manage IP addresses

Each instance can have a private, or fixed, IP address and a public, or floating, one.

Private IP addresses are used for communication between instances, and public ones are used for communication with networks outside the cloud, including the Internet.

When you launch an instance, it is automatically assigned a private IP address that stays the same until you explicitly terminate the instance. Rebooting an instance has no effect on the private IP address.

A pool of floating IP addresses, configured by the cloud operator, is available in OpenStack Compute. You can allocate a certain number of these IP addresses to a project. The maximum number of floating IP addresses per project is defined by the quota.

After you allocate floating IP addresses to a project, you can add a floating IP address from this set to an instance of the project. You can assign a floating IP address to one instance at a time. Floating IP addresses can be disassociated from an instance and associated with another instance of the same project at any time.

List floating IP address information

- To list all floating IP addresses, run the following command:

```
$ nova floating-ip-bulk-list
```

project_id	address	instance_uuid	pool	interface
None	172.24.4.225	None	public	eth0
None	172.24.4.226	None	public	eth0
None	172.24.4.227	None	public	eth0
None	172.24.4.228	None	public	eth0
None	172.24.4.229	None	public	eth0
None	172.24.4.230	None	public	eth0
None	172.24.4.231	None	public	eth0
None	172.24.4.232	None	public	eth0
None	172.24.4.233	None	public	eth0
None	172.24.4.234	None	public	eth0
None	172.24.4.235	None	public	eth0
None	172.24.4.236	None	public	eth0
None	172.24.4.237	None	public	eth0
None	172.24.4.238	None	public	eth0
None	192.168.253.1	None	test	eth0
None	192.168.253.2	None	test	eth0
None	192.168.253.3	None	test	eth0
None	192.168.253.4	None	test	eth0

None	192.168.253.5	None	test	eth0
None	192.168.253.6	None	test	eth0

- To list all pools that provide floating IP addresses, run the following command:

```
$ nova floating-ip-pool-list
```

name
public
test

Assign floating IP addresses

You can assign floating IP addresses to a project and to an instance.

1. Run the following command to allocate a floating IP address to the current project. If more than one IP address pool is available, you can specify the pool from which to allocate the IP address. This example specifies the `public` pool:

```
$ nova floating-ip-create public
```

Ip	Instance Id	Fixed Ip	Pool
172.24.4.225	None	None	public

2. After at least one floating IP address is allocated to the project, assign an IP address to an instance in the project, as follows:

```
$ nova add-floating-ip INSTANCE_NAME_OR_ID FLOATING_IP
```

After you assign the IP address and configure security group rules for the instance, the instance is publicly available at the floating IP address.

3. To remove a floating IP address from an instance, specify the same arguments that you used to assign the IP address, but run the following command:

```
$ nova remove-floating-ip INSTANCE_NAME_OR_ID FLOATING_IP
```

4. To release a floating IP address from the current project, run the following command:

```
$ nova floating-ip-delete FLOATING_IP
```

The IP address is returned to the pool of IP addresses that are available for all projects. If an IP address is assigned to a running instance, it is disassociated from the instance.

Change the size of your server

You change the size of a server by changing its flavor.

1. Show information about your server, including its size, which is shown as the value of the `flavor` property.

```
$ nova show myCirrosServer
```

Property	Value
status	ACTIVE
updated	2013-07-18T15:08:20Z
OS-EXT-STS:task_state	None
OS-EXT-SRV-ATTR:host	devstack
key_name	None
image	cirros-0.3.2-x86_64-uec (397e713c-b95b-4186-ad46-6126863ea0a9)
private network	10.0.0.3
hostId	6e1e69b71ac9b1e6871f91e2dfc9a9b9ceca0f05db68172a81d45385
OS-EXT-STS:vm_state	active
OS-EXT-SRV-ATTR:instance_name	instance-00000005
OS-EXT-SRV-ATTR:hypervisor_hostname	devstack
flavor	m1.small (2)
id	84c6e57d-a6b1-44b6-81eb-fcb36afd31b5
security_groups	[[{'name': 'u'default'}]]
user_id	376744b5910b4b4da7d8e6cb483b06a8
name	myCirrosServer
created	2013-07-18T15:07:59Z
tenant_id	66265572db174a7aa66eba661f58eb9e
OS-DCF:diskConfig	MANUAL
metadata	{'description': 'u'Small test image', 'u'creator': 'u'joecool'}
accessIPv4	
accessIPv6	
progress	0
OS-EXT-STS:power_state	1
OS-EXT-AZ:availability_zone	nova
config_drive	

The size (flavor) of the server is `m1.small (2)`.

- List the available flavors with the following command:

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

- To resize the server, pass the server ID or name and the new flavor to the **nova resize** command. Include the `--poll` parameter to report the resize progress.

```
$ nova resize myCirrosServer 4 --poll
```

```
Instance resizing... 100% complete
Finished
```

- Show the status for your server:

```
$ nova list
+-----+-----+-----+-----+-----+
+-----+
| ID                               | Name          | Status |
| Networks                         |
+-----+-----+-----+-----+-----+
+-----+
| 84c6e57d-a6b1-44b6-81eb-fcb36afd31b5 | myCirrosServer | RESIZE |
| private=172.16.101.6, public=10.4.113.6 |
+-----+-----+-----+-----+-----+
+-----+
```

When the resize completes, the status becomes `VERIFY_RESIZE`.

- Confirm the resize:

```
$ nova resize-confirm 84c6e57d-a6b1-44b6-81eb-fcb36afd31b5
```

The server status becomes `ACTIVE`.

- If the resize fails or does not work as expected, you can revert the resize:

```
$ nova resize-revert 84c6e57d-a6b1-44b6-81eb-fcb36afd31b5
```

The server status becomes `ACTIVE`.

Search for an instance using IP address

You can search for an instance using the IP address parameter, `--ip`, with the **nova list** command.

```
$ nova list --ip IP_ADDRESS
```

The following example shows the results of a search on 10.0.0.4.

```
$ nova list --ip 10.0.0.4
+-----+-----+-----+-----+-----+
| ID                               | Name                               | Status | Task |
| State | Power State | Networks          | | |
+-----+-----+-----+-----+-----+
| 8a99547e-7385-4ad1-ae50-4ecfaad5f42 | myInstanceFromVolume             | ACTIVE | None |
|   Running   |   private=10.0.0.4   | | |
+-----+-----+-----+-----+-----+
```

Stop and start an instance

Use one of the following methods to stop and start an instance.

Pause and unpause an instance

- To pause an instance, run the following command:

```
$ nova pause INSTANCE_NAME
```

This command stores the state of the VM in RAM. A paused instance continues to run in a frozen state.

- To unpause the instance, run the following command:

```
$ nova unpause INSTANCE_NAME
```

Suspend and resume an instance

Administrative users might want to suspend an instance if it is infrequently used or to perform system maintenance. When you suspend an instance, its VM state is stored on disk, all memory is written to disk, and the virtual machine is stopped. Suspending an instance is similar to placing a device in hibernation; memory and vCPUs become available to create other instances.

- To initiate a hypervisor-level suspend operation, run the following command:

```
$ nova suspend INSTANCE_NAME
```

- To resume a suspended instance, run the following command:

```
$ nova resume INSTANCE_NAME
```

Reboot an instance

You can soft or hard reboot a running instance. A soft reboot attempts a graceful shut down and restart of the instance. A hard reboot power cycles the instance.

- By default, when you reboot a server, it is a soft reboot.

```
$ nova reboot SERVER
```

To perform a hard reboot, pass the `--hard` parameter, as follows:

```
$ nova reboot --hard SERVER
```

Delete an instance

When you no longer need an instance, you can delete it.

1. List all instances:

```
$ nova list
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ID                               | Name                               | Status |
+-----+-----+-----+-----+
| 84c6e57d-a6b1-44b6-81eb-fcb36afd31b5 | myCirrosServer                    | ACTIVE |
| None                               | Running                           | private=10.0.0.3 |
+-----+-----+-----+-----+
| 8a99547e-7385-4ad1-ae50-4ecfaaad5f42 | myInstanceFromVolume              | ACTIVE |
| None                               | Running                           | private=10.0.0.4 |
+-----+-----+-----+-----+
| d7efd3e4-d375-46d1-9d57-372b6e4bdb7f | newServer                          | ERROR  |
| None                               | NOSTATE                            |         |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

2. Run the **nova delete** command to delete the instance. The following example shows deletion of the `newServer` instance, which is in `ERROR` state:

```
$ nova delete newServer
```

The command does not notify that your server was deleted.

3. To verify that the server was deleted, run the **nova list** command:

```
$ nova list
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ID                               | Name                               | Status |
+-----+-----+-----+-----+
| 84c6e57d-a6b1-44b6-81eb-fcb36afd31b5 | myCirrosServer                    | ACTIVE |
| None                               | Running                           | private=10.0.0.3 |
+-----+-----+-----+-----+
| 8a99547e-7385-4ad1-ae50-4ecfaaad5f42 | myInstanceFromVolume              | ACTIVE |
| None                               | Running                           | private=10.0.0.4 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

The deleted instance does not appear in the list.

Access an instance through a console

To access an instance through a VNC console, run the following command:

```
$ nova get-vnc-console INSTANCE_NAME xvpvnc
```

The command returns a URL from which you can access your instance:

```
+-----+
+-----+
+
| Type   | Url
+-----+
+
| xvpvnc | http://166.78.190.96:6081/console?token=c83ae3a3-15c4-4890-8d45-
|        | aefb494a8d6c
+-----+
+-----+
+
```



Note

To access an instance through a non-VNC console, specify the *novnc* parameter instead of the *xvpvnc* parameter.

Manage bare-metal nodes

The bare-metal driver for OpenStack Compute manages provisioning of physical hardware by using common cloud APIs and tools such as Orchestration (Heat). The use case for this driver is for single tenant clouds such as a high-performance computing cluster or for deploying OpenStack itself.

If you use the bare-metal driver, you must create a network interface and add it to a bare-metal node. Then, you can launch an instance from a bare-metal image.



Note

Development efforts are focused on moving the driver out of the Compute code base in the Icehouse release.

You can list and delete bare-metal nodes. When you delete a node, any associated network interfaces are removed. You can list and remove network interfaces that are associated with a bare-metal node.

Commands

The following commands can be used to manage bare-metal nodes.

- **baremetal-interface-add.** Adds a network interface to a bare-metal node.
- **baremetal-interface-list.** Lists network interfaces associated with a bare-metal node.
- **baremetal-interface-remove.** Removes a network interface from a bare-metal node.

- **baremetal-node-create.** Creates a bare-metal node.
- **baremetal-node-delete.** Removes a bare-metal node and any associated interfaces.
- **baremetal-node-list.** Lists available bare-metal nodes.
- **baremetal-node-show.** Shows information about a bare-metal node.

Create a bare-metal node

When you create a bare-metal node, your PM address, username, and password should match those that are configured in your hardware's BIOS/IPMI configuration.

```
$ nova baremetal-node-create --pm_address=PM_ADDRESS --pm_user=PM_USERNAME \
  --pm_password=PM_PASSWORD $(hostname -f) 1 512 10 aa:bb:cc:dd:ee:ff
```

The following example shows the command and results from creating a node with the PM address 1.2.3.4, the PM username ipmi, and password ipmi.

```
$ 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

Add a network interface to the node:

For each NIC on the node, you must create an interface, specifying the interface's MAC address.

```
$ 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

Launch an instance from a bare-metal image:

A bare-metal instance is an instance created directly on a physical machine without any virtualization layer running underneath it. Nova retains power control via IPMI. In some situations, Nova may retain network control via Neutron and OpenFlow.

```
$ 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 ...
```



Note

Set the `--availability_zone` parameter to specify which zone or node to use to start the server. Separate the zone from the host name with a comma. For example:

```
$ nova boot --availability_zone=zone:HOST,NODE
```

host is optional for the `--availability_zone` parameter. You can specify simply `zone: ,node`. You must still use the comma.

List bare-metal nodes and interfaces:

Use the `nova baremetal-node-list` command to view all bare-metal nodes and interfaces. When a node is in use, its status includes the UUID of the instance that runs on it:

```
$ 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    |                      |      |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
```

Show details for a bare-metal node:

Use the `nova baremetal-node-show` command to view the details for a bare-metal node.

```
$ 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
---------------	------

Show usage statistics for hosts and instances

You can show basic statistics on resource usage for hosts and instances.



Note

For more sophisticated monitoring, see the [Ceilometer](#) project, which is under development. You can also use tools, such as [Ganglia](#) or [Graphite](#), to gather more detailed data.

Show host usage statistics

The following examples show the host usage statistics for a host called `devstack`.

- List the hosts and the nova-related services that run on them:

```
$ nova host-list
+-----+-----+-----+
| host_name | service | zone |
+-----+-----+-----+
| devstack  | conductor | internal |
| devstack  | compute  | nova    |
| devstack  | cert     | internal |
| devstack  | network  | internal |
| devstack  | scheduler | internal |
| devstack  | consoleauth | internal |
+-----+-----+-----+
```

- Get a summary of resource usage of all of the instances running on the host:

```
$ nova host-describe devstack
+-----+-----+-----+-----+-----+
| HOST          | PROJECT          | cpu | memory_mb | disk_gb |
+-----+-----+-----+-----+-----+
| devstack     | (total)          | 2   | 4003      | 157     |
| devstack     | (used_now)       | 3   | 5120      | 40      |
| devstack     | (used_max)       | 3   | 4608      | 40      |
| devstack     | b70d90d65e464582b6b2161cf3603ced | 1   | 512       | 0       |
| devstack     | 66265572db174a7aa66eba661f58eb9e | 2   | 4096      | 40      |
+-----+-----+-----+-----+-----+
```

The `cpu` column shows the sum of the virtual CPUs for instances running on the host.

The `memory_mb` column shows the sum of the memory (in MB) allocated to the instances that run on the host.

The `disk_gb` column shows the sum of the root and ephemeral disk sizes (in GB) of the instances that run on the host.

The row that has the value `used_now` in the `PROJECT` column shows the sum of the resources allocated to the instances that run on the host, plus the resources allocated to the virtual machine of the host itself.

The row that has the value `used_max` row in the `PROJECT` column shows the sum of the resources allocated to the instances that run on the host.



Note

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

Show instance usage statistics

- Get CPU, memory, I/O, and network statistics for an instance.

1. List instances:

```
$ nova list
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ID                | Name                | Status |
+-----+-----+-----+-----+
| Task State | Power State | Networks |
+-----+-----+-----+-----+
| 84c6e57d-a6b1-44b6-81eb-fcb36afd31b5 | myCirrosServer      | ACTIVE |
| None        | Running     | private=10.0.0.3 |
| 8a99547e-7385-4ad1-ae50-4ecfaaad5f42 | myInstanceFromVolume | ACTIVE |
| None        | Running     | private=10.0.0.4 |
+-----+-----+-----+-----+
```

2. Get diagnostic statistics:

```
$ nova diagnostics myCirrosServer
+-----+-----+
| Property          | Value              |
+-----+-----+
| vnet1_rx          | 1210744            |
| cpu0_time         | 19624610000000    |
| vda_read          | 0                  |
| vda_write         | 0                  |
| vda_write_req     | 0                  |
| vnet1_tx          | 863734             |
| vnet1_tx_errors   | 0                  |
| vnet1_rx_drop     | 0                  |
| vnet1_tx_packets  | 3855               |
| vnet1_tx_drop     | 0                  |
| vnet1_rx_errors   | 0                  |
| memory            | 2097152            |
| vnet1_rx_packets  | 5485               |
| vda_read_req      | 0                  |
| vda_errors        | -1                 |
+-----+-----+
```

- Get summary statistics for each tenant:

```
$ nova usage-list
Usage from 2013-06-25 to 2013-07-24:
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```


Tenant ID Disk GB-Hours	Instances	RAM MB-Hours	CPU Hours
b70d90d65e464582b6b2161cf3603ced 0.00	1	344064.44	672.00
66265572db174a7aa66eba661f58eb9e 6558.86	3	671626.76	327.94

Provide user data to instances

A *user data* file is a special key in the metadata service that holds a file that cloud-aware applications in the guest instance can access. For example, one application that uses user data is the [cloud-init](#) system, which is an open-source package from Ubuntu that is available on various Linux distributions and which handles early initialization of a cloud instance.

You can place user data in a local file and pass it through the `--user-data <user-data-file>` parameter at instance creation:

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

Use snapshots to migrate instances

To use snapshots to migrate instances from OpenStack projects to clouds, complete these steps.

1. In the source project, perform the following steps:
 1. [Create a snapshot of the instance.](#)
 2. [Download the snapshot as an image.](#)
2. In the destination project, perform the following steps:
 1. [Import the snapshot to the new environment.](#)
 2. [Boot a new instance from the snapshot.](#)



Note

Some cloud providers allow only administrators to perform this task.

Create a snapshot of the instance

1. Shut down the source VM before you take the snapshot to ensure that all data is flushed to disk. If necessary, list the instances to view get the instance name.

```
$ nova list
+-----+-----+-----+-----+
+-----+
| ID                | Name          | Status  | Networks |
+-----+-----+-----+-----+
| c41f3074-c82a-4837-8673-fa7e9fea7e11 | myInstance   | ACTIVE  | private=10.0.0.3 |
+-----+-----+-----+-----+
```

```
$ nova stop example
```

2. Use the **nova list** command to confirm that the instance shows a SHUTOFF status.

```
$ nova list
+-----+-----+-----+-----+
+-----+
| ID                | Name          | Status  | Networks |
+-----+-----+-----+-----+
| c41f3074-c82a-4837-8673-fa7e9fea7e11 | myInstance   | SHUTOFF | private=10.0.0.3 |
+-----+-----+-----+-----+
```

3. Use the **nova image-create** command to take a snapshot. Use the **nova image-list** command to check the status until the status is ACTIVE:

```
$ nova image-create --poll myInstance myInstanceSnapshot
Instance snapshotting... 50% complete

$ nova image-list
+-----+-----+
+-----+-----+
| ID                                     | Name                               |
| Status | Server |                                     |
+-----+-----+
+-----+-----+
| 657ebb01-6fae-47dc-986a-e49c4dd8c433 | cirros-0.3.2-x86_64-uec           | |
| ACTIVE |      |                                     |
| 72074c6d-bf52-4a56-a61c-02a17bf3819b | cirros-0.3.2-x86_64-uec-kernel    |
| ACTIVE |      |                                     |
| 3c5e5f06-637b-413e-90f6-ca7ed015ec9e | cirros-0.3.2-x86_64-uec-ramdisk   |
| ACTIVE |      |                                     |
| f30b204e-1ce6-40e7-b8d9-b353d4d84e7d | myInstanceSnapshot               |
| ACTIVE |      |                                     |
+-----+-----+
+-----+-----+
```

Download the snapshot as an image

1. Get the image ID:

```
$ nova image-list
+-----+-----+-----+
+-----+-----+-----+
| ID                                     | Name                               | Status |
| Server |                                     |        |
+-----+-----+-----+
+-----+-----+-----+
| f30b204e-1ce6-40e7-b8d9-b353d4d84e7d | myInstanceSnapshot | ACTIVE |
| c41f3074-c82a-4837-8673-fa7e9fea7e11 |                   |        |
+-----+-----+-----+
+-----+-----+-----+
```

2. Download the snapshot by using the image ID that was returned in the previous step:

```
$ glance image-download --file snapshot.raw f30b204e-1ce6-40e7-b8d9-
b353d4d84e7d
```



Note

The `glance image-download` command requires the image ID and cannot use the image name.

Ensure there is sufficient space on the destination file system for the image file.

3. Make the image available to the new environment, either through HTTP or with direct upload to a machine (`scp`).

Import the snapshot to new environment

In the new project or cloud environment, import the snapshot:

```
$ glance image-create --copy-from IMAGE_URL
```

Boot a new instance from the snapshot

In the new project or cloud environment, use the snapshot to create the new instance:

```
$ nova boot --flavor m1.tiny --image myInstanceSnapshot myNewInstance
```

Store metadata on a configuration drive

You can configure OpenStack to write metadata to a special configuration drive that attaches to the instance when it boots. The instance can mount this drive and read files from it to get information that is normally available through the [metadata service](#). This metadata is different from the user data.

One use case for using the configuration drive is to pass a networking configuration when you do not use DHCP to assign IP addresses to instances. For example, you might pass the IP address configuration for the instance through the configuration drive, which the instance can mount and access before you configure the network settings for the instance.

Any modern guest operating system that is capable of mounting an ISO 9660 or VFAT file system can use the configuration drive.

Requirements and guidelines

To use the configuration drive, you must follow the following requirements for the compute host and image.

Compute host requirements

- The following hypervisors support the configuration drive: libvirt, XenServer, Hyper-V, and VMWare.
- To use configuration drive with libvirt, XenServer, or VMWare, you must first install the `genisoimage` package on each compute host. Otherwise, instances do not boot properly.

Use the `mkisofs_cmd` flag to set the path where you install the `genisoimage` program. If `genisoimage` is in same path as the `nova-compute` service, you do not need to set this flag.

- To use configuration drive with Hyper-V, you must set the `mkisofs_cmd` value to the full path to an `mkisofs.exe` installation. Additionally, you must set the `qemu_img_cmd` value in the `hyperv` configuration section to the full path to an `qemu-img` command installation.

Image requirements

- An image built with a recent version of the cloud-init package can automatically access metadata passed through the configuration drive. The cloud-init package version 0.7.1 works with Ubuntu and Fedora based images, such as Red Hat Enterprise Linux.
- If an image does not have the cloud-init package installed, you must customize the image to run a script that mounts the configuration drive on boot, reads the data from the drive, and takes appropriate action such as adding the public key to an account. See [the section called "Configuration drive contents" \[72\]](#) for details about how data is organized on the configuration drive.
- If you use Xen with a configuration drive, use the `xenapi_disable_agent` configuration parameter to disable the agent.

Guidelines

- Do not rely on the presence of the EC2 metadata in the configuration drive, because this content might be removed in a future release. For example, do not rely on files in the `ec2` directory.
- When you create images that access configuration drive data and multiple directories are under the `openstack` directory, always select the highest API version by date that your consumer supports. For example, if your guest image supports the 2012-03-05, 2012-08-05, and 2013-04-13 versions, try 2013-04-13 first and fall back to a previous version if 2013-04-13 is not present.

Enable and access the configuration drive

1. To enable the configuration drive, pass the `--config-drive=true` parameter to the **nova boot** command.

The following example enables the configuration drive and passes user data, two files, and two key/value metadata pairs, all of which are accessible from the configuration drive:

```
$ nova boot --config-drive=true --image my-image-name --key-name mykey --
flavor 1 --user-data ./my-user-data.txt myinstance --file /etc/network/
interfaces=/home/myuser/instance-interfaces --file known_hosts=/home/
myuser/.ssh/known_hosts --meta role=webrowsers --meta essential=false
```

You can also configure the Compute service to always create a configuration drive by setting the following option in the `/etc/nova/nova.conf` file:

```
force_config_drive=true
```



Note

If a user passes the `--config-drive=true` flag to the **nova boot** command, an administrator cannot disable the configuration drive.

2. If your guest operating system supports accessing disk by label, you can mount the configuration drive as the `/dev/disk/by-label/configurationDriveVolumeLabel` device. In the following example, the configuration drive has the `config-2` volume label.

```
# mkdir -p /mnt/config
# mount /dev/disk/by-label/config-2 /mnt/config
```



Note

Ensure that you use at least version 0.3.1 of CirrOS for configuration drive support.

If your guest operating system does not use `udev`, the `/dev/disk/by-label` directory is not present.

You can use the **blkid** command to identify the block device that corresponds to the configuration drive. For example, when you boot the CirrOS image with the `m1.tiny` flavor, the device is `/dev/vdb`:

```
# blkid -t LABEL="config-2" -o device
```

```
/dev/vdb
```

Once identified, you can mount the device:

```
# mkdir -p /mnt/config
# mount /dev/vdb /mnt/config
```

Configuration drive contents

In this example, the contents of the configuration drive are as follows:

```
ec2/2009-04-04/meta-data.json
ec2/2009-04-04/user-data
ec2/latest/meta-data.json
ec2/latest/user-data
openstack/2012-08-10/meta_data.json
openstack/2012-08-10/user_data
openstack/content
openstack/content/0000
openstack/content/0001
openstack/latest/meta_data.json
openstack/latest/user_data
```

The files that appear on the configuration drive depend on the arguments that you pass to the **nova boot** command.

OpenStack metadata format

The following example shows the contents of the `openstack/2012-08-10/meta_data.json` and `openstack/latest/meta_data.json` files. These files are identical. The file contents are formatted for readability.

```
{
  "availability_zone": "nova",
  "files": [
    {
      "content_path": "/content/0000",
      "path": "/etc/network/interfaces"
    },
    {
      "content_path": "/content/0001",
      "path": "known_hosts"
    }
  ],
  "hostname": "test.novalocal",
  "launch_index": 0,
  "name": "test",
  "meta": {
    "role": "webservers",
    "essential": "false"
  }
},
```



```

"public_keys":{
  "mykey":"ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDBqUfVvCSez0/
Wfpd8dLLgZXV9GtXQ7hnMN+Z0OWQUyebVEHeylCXuin0uY1cAJMhUq8j98SiW
+cU0sU4J3x5l2+xilbodDmlBtFWVeLlOQINpfV1n8fKjHB
+ynPpelF6tMDvrFGULJs44t30BrujMXBe8Rq44cCk6wqyjATA3rQ== Generated by Nova\n"
},
  "uuid":"83679162-1378-4288-a2d4-70e13ec132aa"
}

```

Note the effect of the `--file /etc/network/interfaces=/home/myuser/instance-interfaces` argument that was passed to the `nova boot` command. The contents of this file are contained in the `openstack/content/0000` file on the configuration drive, and the path is specified as `/etc/network/interfaces` in the `meta_data.json` file.

EC2 metadata format

The following example shows the contents of the `ec2/2009-04-04/meta-data.json` and the `ec2/latest/meta-data.json` files. These files are identical. The file contents are formatted to improve readability.

```

{
  "ami-id": "ami-00000001",
  "ami-launch-index": 0,
  "ami-manifest-path": "FIXME",
  "block-device-mapping": {
    "ami": "sda1",
    "ephemeral0": "sda2",
    "root": "/dev/sda1",
    "swap": "sda3"
  },
  "hostname": "test.novalocal",
  "instance-action": "none",
  "instance-id": "i-00000001",
  "instance-type": "ml.tiny",
  "kernel-id": "aki-00000002",
  "local-hostname": "test.novalocal",
  "local-ipv4": null,
  "placement": {
    "availability-zone": "nova"
  },
  "public-hostname": "test.novalocal",
  "public-ipv4": "",
  "public-keys": {
    "0": {
      "openssh-key": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDBqUfVvCSez0/
Wfpd8dLLgZXV9GtXQ7hnMN+Z0OWQUyebVEHeylCXuin0uY1cAJMhUq8j98SiW
+cU0sU4J3x5l2+xilbodDmlBtFWVeLlOQINpfV1n8fKjHB
+ynPpelF6tMDvrFGULJs44t30BrujMXBe8Rq44cCk6wqyjATA3rQ== Generated by Nova\n"
    }
  },
  "ramdisk-id": "ari-00000003",
  "reservation-id": "r-7lfps8wj",
  "security-groups": [
    "default"
  ]
}

```

User data

The `openstack/2012-08-10/user_data`, `openstack/latest/user_data`, `ec2/2009-04-04/user-data`, and `ec2/latest/user-data` file are present only if the `--user-data` flag and the contents of the user data file are passed to the `nova boot` command.

Configuration drive format

The default format of the configuration drive as an ISO 9660 file system. To explicitly specify the ISO 9660 format, add the following line to the `/etc/nova/nova.conf` file:

```
config_drive_format=iso9660
```

By default, you cannot attach the configuration drive image as a CD drive instead of as a disk drive. To attach a CD drive, add the following line to the `/etc/nova/nova.conf` file:

```
config_drive_cdrom=true
```

For legacy reasons, you can configure the configuration drive to use VFAT format instead of ISO 9660. It is unlikely that you would require VFAT format because ISO 9660 is widely supported across operating systems. However, to use the VFAT format, add the following line to the `/etc/nova/nova.conf` file:

```
config_drive_format=vfat
```

If you choose VFAT, the configuration drive is 64 MB.

Configuration drive reference

The following table shows the configuration options for the configuration drive.

Table 2.5. Description of configuration options for configdrive

Configuration option = Default value	Description
[DEFAULT]	
<code>config_drive_format = iso9660</code>	(StrOpt) Config drive format. One of iso9660 (default) or vfat
<code>config_drive_skip_versions = 1.0 2007-01-19 2007-03-01 2007-08-29 2007-10-10 2007-12-15 2008-02-01 2008-09-01</code>	(StrOpt) List of metadata versions to skip placing into the config drive
<code>config_drive_tmpdir = None</code>	(StrOpt) Where to put temporary files associated with config drive creation
<code>force_config_drive = None</code>	(StrOpt) Set to force injection to take place on a config drive (if set, valid options are: always)
<code>mkisofs_cmd = genisoimage</code>	(StrOpt) Name and optionally path of the tool used for ISO image creation
[hyperv]	
<code>config_drive_cdrom = False</code>	(BoolOpt) Attaches the Config Drive image as a cdrom drive instead of a disk drive
<code>config_drive_inject_password = False</code>	(BoolOpt) Sets the admin password in the config drive image

Create and manage networks

Before you run commands, set the following environment variables:

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

Create networks

1. List the extensions of the system:

```
$ neutron 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	Neutron L3 Router
lbaas	LoadBalancing service
extraroute	Neutron Extra Route

2. Create a network:

```
$ neutron 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.

3. Create a network with specified provider network type:

```
$ neutron 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 previously, the unknown option `--provider:network-type` is used to create a local provider network.

Create subnets

- Create a subnet:

```
$ neutron 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

The `subnet-create` command has the following positional and optional parameters:

- The name or ID of the network to which the subnet belongs.
In this example, `net1` is a positional argument that specifies the network name.
- The CIDR of the subnet.
In this example, `192.168.2.0/24` is a positional argument that specifies the CIDR.
- The subnet name, which is optional.
In this example, `--name subnet1` specifies the name of the subnet.

Create routers

1. Create a router:

```
$ neutron router-create router1
```

```
Created a new router:
```

Field	Value
admin_state_up	True
external_gateway_info	
id	6e1f11ed-014b-4c16-8664-f4f615a3137a
name	router1
status	ACTIVE
tenant_id	7b5970fbe7724bf9b74c245e66b92abf

Take note of the unique router identifier returned, this will be required in subsequent steps.

2. Link the router to the external provider network:

```
$ neutron router-gateway-set ROUTER NETWORK
```

Replace *ROUTER* with the unique identifier of the router, replace *NETWORK* with the unique identifier of the external provider network.

3. Link the router to the subnet:

```
$ neutron router-interface-add ROUTER SUBNET
```

Replace *ROUTER* with the unique identifier of the router, replace *SUBNET* with the unique identifier of the subnet.

Create ports

1. Create a port with specified IP address:

```
$ neutron 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 previous command, `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.



Note

When creating a port, you can specify any unallocated IP in the subnet even if the address is not in a pre-defined pool of allocated IP addresses (set by your cloud provider).

2. Create a port without specified IP address:

```
$ neutron 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
+-----+
+-----+
+

```



Note

Note that the system allocates one IP address if you do not specify an IP address in the **neutron port-create** command.

3. Query ports with specified fixed IP addresses:

```
$ neutron 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.

Manage objects and containers

The OpenStack Object Storage Service provides the **swift** client, which is a command-line interface (CLI). Use this client to list objects and containers, upload objects to containers, and download or delete objects from containers. You can also gather statistics and update metadata for accounts, containers, and objects.

This client is based on the native swift client library, `client.py`, which seamlessly re-authenticates if the current token expires during processing, retries operations multiple times, and provides a processing concurrency of 10.

Create and manage containers

1. To create a container:

```
$ swift post CONTAINER
```

Replace `CONTAINER` with the name of your container.

Users have roles on accounts. For example, a user with the admin role has full access to all containers and objects in an account. You can set access control lists (ACLs) at the container level and support lists for read and write access, which you set with the `X-Container-Read` and `X-Container-Write` header, respectively.

To give a user read access, use the **swift post** command with the `-r` parameter. To give a user write access, use the `-w` parameter.

The following example enables the `testuser` user to read objects in the container:

```
$ swift post -r 'testuser'
```

You can also use this command with a list of users.

If you use StaticWeb middleware to enable Object Storage to serve public web content, use `.r:`, followed by a list of allowed referrers.

The following command gives object access to all referring domains:

```
$ swift post -r '.r:*
```

2. To list all containers:

```
$ swift list
```

3. To check the status of containers:

```
$ swift stat
Account: AUTH_7b5970fbe7724bf9b74c245e77c03bcg
Containers: 2
Objects: 3
Bytes: 268826
Accept-Ranges: bytes
X-Timestamp: 1392683866.17952
Content-Type: text/plain; charset=utf-8
```


You can also use the **swift stat** command with the *ACCOUNT* or *CONTAINER* names as parameters.

```
$ swift stat CONTAINER
Account: AUTH_7b5970fbe7724bf9b74c245e77c03bcg
Container: storagel
Objects: 2
Bytes: 240221
Read ACL:
Write ACL:
Sync To:
Sync Key:
Accept-Ranges: bytes
X-Timestamp: 1392683866.20180
Content-Type: text/plain; charset=utf-8
```

Manage objects

1. To upload an object to a container:

```
$ swift upload CONTAINER OBJECT_FILENAME
```

To upload in chunks, for large files:

```
$ swift upload -S CHUNK_SIZE CONTAINER OBJECT_FILENAME
```

2. To check the status of the object:

```
$ swift stat CONTAINER OBJECT_FILENAME
Account: AUTH_7b5970fbe7724bf9b74c245e77c03bcg
Container: storagel
Object: images
Content Type: application/octet-stream
Content Length: 211616
Last Modified: Tue, 18 Feb 2014 00:40:36 GMT
ETag: 82169623d55158f70a0d720f238ec3ef
Meta Orig-Filename: images.jpg
Accept-Ranges: bytes
X-Timestamp: 1392684036.33306
```

3. To list objects in a container:

```
$ swift list CONTAINER OBJECT_FILENAME
```

4. To download an object from a container:

```
$ swift download CONTAINER OBJECT_FILENAME
```

Create and manage stacks

The template languages are described in [the Template Guide](#) in the [Heat developer documentation](#).

Create a stack from an example template file

1. To create a stack, or template, from an [example template file](#), run the following command:

```
$ heat stack-create mystack --template-file=/PATH_TO_HEAT_TEMPLATES/
WordPress_Single_Instance.template
    --parameters="InstanceType=m1.
large;DBUsername=USERNAME;DBPassword=PASSWORD;KeyName=HEAT_KEY;LinuxDistribution=
F17"
```

The `--parameters` values that you specify depend on the parameters that are defined in the template. If a website hosts the template file, you can specify the URL with the `--template-url` parameter instead of the `--template-file` parameter.

The command returns the following output:

```
+-----+-----+
+-----+-----+
| id                | stack_name  | stack_status
| creation_time     |             |
+-----+-----+
+-----+-----+
| 4c712026-dcd5-4664-90b8-0915494c1332 | mystack    |
CREATE_IN_PROGRESS | 2013-04-03T23:22:08Z |
+-----+-----+
+-----+-----+
```

2. You can also use the `stack-create` command to validate a template file without creating a stack from it.

To do so, run the following command:

```
$ heat stack-create mystack --template-file=/PATH_TO_HEAT_TEMPLATES/
WordPress_Single_Instance.template
```

If validation fails, the response returns an error message.

Get information about stacks

To explore the state and history of a particular stack, you can run a number of commands.

- To see which stacks are visible to the current user, run the following command:

```
$ 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 |
+-----+-----+-----+
+-----+
```

- To show the details of a stack, run the following command:

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

- A stack consists of a collection of resources.

To list the resources and their status, run the following command:

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

- To show the details for the specified resource in a stack, run the following command:

```
$ 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.

To display life-cycle events, run:

```
$ 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 |
+-----+-----+-----+-----+
+-----+
```

- To show the details for a particular event, run the following command:

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

Update a stack

- To update an existing stack from a modified template file, run a command like the following command:

```
$ 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  |
+-----+-----+-----+
| 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 |             |                 |
+-----+-----+-----+
+-----+
```

Some resources are updated in-place, while others are replaced with new resources.

Measure cloud resources

Telemetry measures cloud resources in OpenStack.

It collects information about how much, who, what, and when with regards to billing. Currently, metering is available through only the **ceilometer** command-line client.

To model data, Telemetry uses these abstractions:

Meter Measures a specific aspect of resource usage, such as the existence of a running instance, or ongoing performance, such as the CPU utilization for an instance. Meters exist for each type of resource. For example, a separate `cpu_util` meter exists for each instance. The life cycle of a meter is decoupled from the existence of its related resource. The meter persists after the resource goes away.

A meter has the following attributes:

- String name.
- A unit of measurement.
- A type. Indicates whether values increase monotonically (cumulative), are interpreted as a change from the previous value (delta), or are standalone and relate only to the current duration (gauge).

Sample An individual data point that is associated with a specific meter. Has the same attributes as the meter, with the addition of timestamp and value attributes. The value attribute is also known as the sample *volume*.

Statistic A set of data point aggregates over a time duration. (In contrast, a sample represents a single data point.) The Telemetry service employs these aggregation functions:

- **count**. The number of samples in each period.
- **max**. The maximum number of sample volumes in each period.
- **min**. The minimum number of sample volumes in each period.
- **avg**. The average of sample volumes over each period.
- **sum**. The sum of sample volumes over each period.

Alarm A set of rules that define a monitor and a current state, with edge-triggered actions associated with target states. Provides user-oriented Monitoring-as-a-Service and a general purpose utility for OpenStack. Orchestration auto scaling is a typical use-case. Alarms follow a tristate model of `ok`, `alarm`, and `insufficient data`. For conventional threshold-oriented alarms, a static threshold value and comparison operator govern state transitions. The comparison operator compares a selected meter statistic against an evaluation window of configurable length into the recent past.

This example uses the **heat** client to create an auto-scaling stack and the **ceilometer** client to measure resources.

1. Create an auto-scaling stack:

```
$ heat stack-create -f cfn/F17/AutoScalingCeilometer.yaml -P "KeyName=heat_key"
```

2. List the heat resources that were created:

```
$ heat resource-list
+-----+-----+
+-----+-----+
| resource_name          | resource_type          |
resource_status | updated_time          |
+-----+-----+
+-----+-----+
| CfnUser                | AWS::IAM::User        |
CREATE_COMPLETE | 2013-10-02T05:53:41Z |
| WebServerKeys          | AWS::IAM::AccessKey   |
CREATE_COMPLETE | 2013-10-02T05:53:42Z |
| LaunchConfig          | AWS::AutoScaling::LaunchConfiguration |
CREATE_COMPLETE | 2013-10-02T05:53:43Z |
| ElasticLoadBalancer    | AWS::ElasticLoadBalancing::LoadBalancer |
UPDATE_COMPLETE | 2013-10-02T05:55:58Z |
| WebServerGroup         | AWS::AutoScaling::AutoScalingGroup |
CREATE_COMPLETE | 2013-10-02T05:55:58Z |
| WebServerScaleDownPolicy | AWS::AutoScaling::ScalingPolicy |
CREATE_COMPLETE | 2013-10-02T05:56:00Z |
| WebServerScaleUpPolicy | AWS::AutoScaling::ScalingPolicy |
CREATE_COMPLETE | 2013-10-02T05:56:00Z |
| CPUAlarmHigh           | OS::Ceilometer::Alarm |
CREATE_COMPLETE | 2013-10-02T05:56:02Z |
| CPUAlarmLow            | OS::Ceilometer::Alarm |
CREATE_COMPLETE | 2013-10-02T05:56:02Z |
+-----+-----+
+-----+-----+
```

3. List the alarms that are set:

```
$ ceilometer alarm-list
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| Alarm ID              | Name                  |
State          | Enabled | Continuous | Alarm condition
|
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| 4f896b40-0859-460b-9c6a-b0d329814496 | as-CPUAlarmLow-i6qqgkf2fubs |
insufficient data | True   | False   | cpu_util < 15.0 during 1x 60s
|
| 75d8ecf7-afc5-4bdc-95ff-19ed9ba22920 | as-CPUAlarmHigh-sf4muyfruy5m |
insufficient data | True   | False   | cpu_util > 50.0 during 1x 60s
|
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

4. List the meters that are set:

```

$ ceilometer meter-list
+-----+-----+-----+
+-----+
+-----+
+-----+
| Name                | Type      | Unit      | Resource ID
|                   |           |           | Project ID
|                   |           |           |
+-----+-----+-----+
+-----+
+-----+
| cpu                 | cumulative | ns        | 3965b41b-81b0-4386-
bea5-6ec37c8841c1 | d1a2996d3b1f4e0e8645ba9650308011 |
bf03bf32e3884d489004ac995ff7a61c |
| cpu                 | cumulative | ns        | 62520a83-73c7-4084-
be54-275fe770ef2c | d1a2996d3b1f4e0e8645ba9650308011 |
bf03bf32e3884d489004ac995ff7a61c |
| cpu_util            | gauge      | %         | 3965b41b-81b0-4386-
bea5-6ec37c8841c1 | d1a2996d3b1f4e0e8645ba9650308011 |
bf03bf32e3884d489004ac995ff7a61c |
+-----+-----+-----+
+-----+
+-----+
+-----+

```

5. List samples:

```

$ ceilometer sample-list -m cpu_util
+-----+-----+-----+-----+
+-----+
+-----+
| Resource ID                | Name      | Type      | Volume
| Unit | Timestamp                |           |           |
+-----+-----+-----+-----+
+-----+
| 3965b41b-81b0-4386-bea5-6ec37c8841c1 | cpu_util | gauge     | 3.98333333333
| %      | 2013-10-02T10:50:12 |           |
+-----+-----+-----+-----+
+-----+
+-----+

```

6. View statistics:

```

$ ceilometer statistics -m cpu_util
+-----+-----+-----+-----+
+-----+
+-----+
+-----+
| Period | Period Start          | Period End          | Count | Min
| Max    | Sum                   | Avg                 | Duration | Duration
Start   | Duration End         |                     |           |
+-----+-----+-----+-----+
+-----+
+-----+
+-----+
| 0      | 2013-10-02T10:50:12 | 2013-10-02T10:50:12 | 1      | 3.
9833333333 | 3.9833333333 | 3.9833333333 | 3.9833333333 | 0.0
2013-10-02T10:50:12 | 2013-10-02T10:50:12 |
+-----+-----+-----+-----+
+-----+
+-----+
+-----+

```

Manage volumes

A volume is a detachable block storage device, similar to a USB hard drive. You can attach a volume to only one instance. To create and manage volumes, you use a combination of nova and cinder client commands.

This example creates a `my-new-volume` volume based on an image.

Create a volume

1. List images, and note the ID of the image to use for your volume:

```
$ nova image-list
```

ID	Name	Status	Server
397e713c-b95b-4186-ad46-6126863ea0a9	cirros-0.3.2-x86_64-uec	ACTIVE	
df430cc2-3406-4061-b635-a51c16e488ac	cirros-0.3.2-x86_64-uec-kernel	ACTIVE	
3cf852bd-2332-48f4-9ae4-7d926d50945e	cirros-0.3.2-x86_64-uec-ramdisk	ACTIVE	
7e5142af-1253-4634-bcc6-89482c5f2e8a	myCirrosImage	ACTIVE	84c6e57d-a6b1-44b6-81eb-fcb36afd31b5
89bcd424-9d15-4723-95ec-61540e8a1979	mynsnapshot	ACTIVE	f51ebd07-c33d-4951-8722-1df6aa8afaa4

2. List the availability zones, and note the ID of the availability zone in which to create your volume:

```
$ nova availability-zone-list
```

Name	Status
internal	available
- devstack	
- nova-conductor	enabled (-) 2013-07-25T16:50:44.000000
- nova-consoleauth	enabled (-) 2013-07-25T16:50:44.000000
- nova-scheduler	enabled (-) 2013-07-25T16:50:44.000000
- nova-cert	enabled (-) 2013-07-25T16:50:44.000000
- nova-network	enabled (-) 2013-07-25T16:50:44.000000
nova	available
- devstack	
- nova-compute	enabled (-) 2013-07-25T16:50:39.000000

3. Create a volume with 8 GB of space. Specify the availability zone and image:

```
$ cinder create 8 --display-name my-new-volume --image-id 397e713c-b95b-4186-ad46-6126863ea0a9 --availability-zone nova
```

Property	Value
attachments	[]
availability_zone	nova
bootable	false
created_at	2013-07-25T17:02:12.472269
display_description	None
display_name	my-new-volume
id	573e024d-5235-49ce-8332-be1576d323f8
image_id	397e713c-b95b-4186-ad46-6126863ea0a9

metadata	{}
size	8
snapshot_id	None
source_valid	None
status	creating
volume_type	None

- To verify that your volume was created successfully, list the available volumes:

```
$ cinder list
```

ID	Status	Display Name	Size	Volume Type
573e024d-5235-49ce-8332-be1576d323f8	available	my-new-volume	8	None
bd7cf584-45de-44e3-bf7f-f7b50bf235e3	available	my-bootable-vol	8	None

If your volume was created successfully, its status is `available`. If its status is `error`, you might have exceeded your quota.

Attach a volume to an instance

- Attach your volume to a server:

```
$ nova volume-attach 84c6e57d-a6b1-44b6-81eb-fcb36afd31b5
573e024d-5235-49ce-8332-be1576d323f8 /dev/vdb
```

Property	Value
device	/dev/vdb
serverId	84c6e57d-a6b1-44b6-81eb-fcb36afd31b5
id	573e024d-5235-49ce-8332-be1576d323f8
volumeId	573e024d-5235-49ce-8332-be1576d323f8

Note the ID of your volume.

- Show information for your volume:

```
$ cinder show 573e024d-5235-49ce-8332-be1576d323f8
```

Property	Value
attachments	[[{'device': u'/dev/vdb', 'server_id': u'84c6e57d-a6b1-44b6-81eb-fcb36afd31b5', 'id': u'573e024d-5235-49ce-8332-be1576d323f8', 'volume_id': u'573e024d-5235-49ce-8332-be1576d323f8'}]]
availability_zone	nova
bootable	true
created_at	2013-07-25T17:02:12.000000

```

| display_description | None |
| display_name       | my-new-volume |
| id                 | 573e024d-5235-49ce-8332-be1576d323f8 |
| metadata           | {} |
| os-vol-host-attr:host | devstack |
| os-vol-tenant-attr:tenant_id | 66265572db174a7aa66eba661f58eb9e |
| size               | 8 |
| snapshot_id        | None |
| source_volid       | None |
| status             | in-use |
| volume_image_metadata | {'u'kernel_id': u'df430cc2-3406-4061-b635-a51c16e488ac', u'image_id': u'397e713c-b95b-4186-ad46-6126863ea0a9', u'ramdisk_id': u'3cf852bd-2332-48f4-9ae4-7d926d50945e', u'image_name': u'cirros-0.3.2-x86_64-uec'} |
| volume_type        | None |
+-----+
+
+

```

The output shows that the volume is attached to the server with ID `84c6e57d-a6b1-44b6-81eb-fcb36afd31b5`, is in the nova availability zone, and is bootable.

Resize a volume

1. To resize your volume, you must first detach it from the server.

To detach the volume from your server, pass the server ID and volume ID to the command:

```
$ nova volume-detach 84c6e57d-a6b1-44b6-81eb-fcb36afd31b5
573e024d-5235-49ce-8332-be1576d323f8
```

The `volume-detach` command does not return any output.

2. List volumes:

```
$ cinder list
```

```

+-----+-----+-----+-----+-----+-----+
| ID | Status | Display Name | Size | Volume Type |
+-----+-----+-----+-----+-----+
| 573e024d-5235-49ce-8332-be1576d323f8 | available | my-new-volume | 8 | None |
| true | | | | |
| bd7cf584-45de-44e3-bf7f-f7b50bf235e3 | available | my-bootable-vol | 8 | None |
| true | | | | |
+-----+-----+-----+-----+-----+

```

Note that the volume is now available.

3. Resize the volume by passing the volume ID and the new size (a value greater than the old one) as parameters:

```
$ cinder extend 573e024d-5235-49ce-8332-be1576d323f8 10
```

The **extend** command does not return any output.

Delete a volume

1. To delete your volume, you must first detach it from the server.

To detach the volume from your server and check for the list of existing volumes, see steps 1 and 2 mentioned in [the section called "Resize a volume" \[90\]](#).

2. Delete the volume:

```
$ cinder delete my-new-volume
```

The delete command does not return any output.

3. List the volumes again, and note that the status of your volume is `deleting`:

```
$ cinder list
```

```
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
|          ID          | Status | Display Name | Size | Volume Type |
+-----+-----+-----+-----+-----+
| Bootable | Attached to |          |          |          |
+-----+-----+-----+-----+-----+
| 573e024d-5235-49ce-8332-be1576d323f8 | deleting | my-new-volume | 8 | None |
| true | |          |          |          |
| bd7cf584-45de-44e3-bf7f-f7b50bf235e3 | available | my-bootable-vol | 8 | None |
| true | |          |          |          |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
```

When the volume is fully deleted, it disappears from the list of volumes:

```
$ cinder list
```

```
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
|          ID          | Status | Display Name | Size | Volume Type |
+-----+-----+-----+-----+-----+
| Bootable | Attached to |          |          |          |
+-----+-----+-----+-----+-----+
| bd7cf584-45de-44e3-bf7f-f7b50bf235e3 | available | my-bootable-vol | 8 | None |
| true | |          |          |          |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
```

Transfer a volume

You can transfer a volume from one owner to another by using the **cinder transfer*** commands. The volume donor, or original owner, creates a transfer request and sends the created transfer ID and authorization key to the volume recipient. The volume recipient, or new owner, accepts the transfer by using the ID and key.



Note

The procedure for volume transfer is intended for tenants (both the volume donor and recipient) within the same cloud.

Use cases include:

- Create a custom bootable volume or a volume with a large data set and transfer it to the end customer.
- For bulk import of data to the cloud, the data ingress system creates a new Block Storage volume, copies data from the physical device, and transfers device ownership to the end user.

Create a volume transfer request

1. While logged in as the volume donor, list available volumes:

```
$ cinder list
```

ID	Status	Display Name	Size
72bfce9f-cacf-477a-a092-bf57a7712165	error	None	1
alcdace0-08e4-4dc7-b9dc-457e9bcfe25f	available	None	1

2. As the volume donor, request a volume transfer authorization code for a specific volume:

```
$ cinder transfer-create volumeID
```

The volume must be in an 'available' state or the request will be denied. If the transfer request is valid in the database (that is, it has not expired or been deleted), the volume is placed in an awaiting transfer state. For example:

```
$ cinder transfer-create alcdace0-08e4-4dc7-b9dc-457e9bcfe25f
```

Property	Value
auth_key	b2c8e585cbc68a80
created_at	2013-10-14T15:20:10.121458
id	6e4e9aa4-bed5-4f94-8f76-df43232f44dc
name	None
volume_id	alcdace0-08e4-4dc7-b9dc-457e9bcfe25f



Note

Optionally, you can specify a name for the transfer by using the `--display-name` *displayName* parameter.

- Send the volume transfer ID and authorization key to the new owner (for example, by email).
- View pending transfers:

```
$ cinder transfer-list
```

```
+-----+
+-----+-----+
|          ID          |          VolumeID          |
|  Name  |          |          |
+-----+-----+
+-----+-----+
| 6e4e9aa4-bed5-4f94-8f76-df43232f44dc | a1cdace0-08e4-4dc7-
b9dc-457e9bcfe25f | None |
+-----+-----+
+-----+-----+
```

- After the volume recipient, or new owner, accepts the transfer, you can see that the transfer is no longer available:

```
$ cinder transfer-list
```

```
+-----+-----+-----+
| ID | Volume ID | Name |
+-----+-----+-----+
+-----+-----+-----+
```

Accept a volume transfer request

- As the volume recipient, you must first obtain the transfer ID and authorization key from the original owner.
- Display the transfer request details using the ID:

```
$ cinder transfer-show transferID
```

For example:

```
$ cinder transfer-show 6e4e9aa4-bed5-4f94-8f76-df43232f44dc
```

```
+-----+-----+-----+
| Property |          Value          |
+-----+-----+-----+
| created_at | 2013-10-14T15:20:10.000000 |
| id | 6e4e9aa4-bed5-4f94-8f76-df43232f44dc |
| name | None |
| volume_id | a1cdace0-08e4-4dc7-b9dc-457e9bcfe25f |
+-----+-----+-----+
```

- Accept the request:

```
$ cinder transfer-accept transferID authKey
```

For example:

```
$ cinder transfer-accept 6e4e9aa4-bed5-4f94-8f76-df43232f44dc
b2c8e585cbc68a80
```

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

Property	Value
id	6e4e9aa4-bed5-4f94-8f76-df43232f44dc
name	None
volume_id	a1cdace0-08e4-4dc7-b9dc-457e9bcfe25f



Note

If you do not have a sufficient quota for the transfer, the transfer is refused.

Delete a volume transfer

1. List available volumes and their statuses:

```
$ cinder list
```

ID	Status	Display Name
Size Volume Type Bootable Attached to		
72bfce9f-cacf-477a-a092-bf57a7712165	error	None
1 None false		
a1cdace0-08e4-4dc7-b9dc-457e9bcfe25f	awaiting-transfer	None
1 None false		

2. Find the matching transfer ID:

```
$ cinder transfer-list
```

ID	VolumeID
Name	
a6da6888-7cdf-4291-9c08-8c1f22426b8a	a1cdace0-08e4-4dc7-b9dc-457e9bcfe25f
None	

3. Delete the volume:

```
$ cinder transfer-delete transferID
```

For example:

```
$ cinder transfer-delete a6da6888-7cdf-4291-9c08-8c1f22426b8a
```

4. The transfer list is now empty and the volume is again available for transfer:

```
$ cinder transfer-list
```

ID	Volume ID	Name
----	-----------	------

```

+-----+-----+-----+
+-----+-----+-----+
$ cinder list
+-----+-----+-----+-----+-----+
|          ID          | Status | Display Name | Size |
| Volume Type | Bootable | Attached to |      |
+-----+-----+-----+-----+-----+
| 72bfce9f-cacf-477a-a092-bf57a7712165 | error  |      None    | 1    |
|      None    | false  |              |      |
| a1cdace0-08e4-4dc7-b9dc-457e9bcfe25f | available |      None    | 1    |
|      None    | false  |              |      |
+-----+-----+-----+-----+-----+
+-----+-----+-----+

```

Set a volume to read-only access

To give multiple users shared, secure access to the same data, you can set a volume to read-only access.

Run this command to set a volume to read-only access:

```
$ cinder readonly-mode-update VOLUME BOOLEAN
```

Where *VOLUME* is the ID of the target volume and *BOOLEAN* is a flag that enables read-only or read/write access to the volume.

Valid values for *BOOLEAN* are:

- `true`. Sets the read-only flag in the volume. When you attach the volume to an instance, the instance checks for this flag to determine whether to restrict volume access to read-only.
- `false`. Sets the volume to read/write access.

3. OpenStack Python SDK

Table of Contents

Install the OpenStack SDK	96
Authenticate	96
Manage images	99
Configure access and security for instances	101
Networking	103
Compute	112

Use the OpenStack Python Software Development Kit (SDK) to write Python automation scripts that create and manage resources in your OpenStack cloud. The SDK implements Python bindings to the OpenStack API, which enables you to perform automation tasks in Python by making calls on Python objects rather than making REST calls directly. All OpenStack command-line tools are implemented using the Python SDK.

You should also be familiar with:

- RESTful web services
- HTTP/1.1
- JSON and XML data serialization formats

Install the OpenStack SDK

Each OpenStack project has its own Python library. These libraries are bundled with the command-line clients. For example, the Python bindings for the Compute API are bundled with the `python-novaclient` package.

For details about how to install the clients, see [install the OpenStack command-line clients](#).

Authenticate

When using the SDK, you must authenticate against an OpenStack endpoint before you can use OpenStack services. Each project uses a slightly different syntax for authentication.

You must typically authenticate against a specific version of a service. For example, a client might need to authenticate against Identity v2.0.

Python scripts that use the OpenStack SDK must have access to the credentials contained in the [OpenStack RC file](#). Because credentials are sensitive information, do not include them in your scripts. This guide assumes that users source the `PROJECT-openrc.sh` file and access the credentials by using the environment variables in the Python scripts.

Authenticate against an Identity endpoint

To authenticate against the Identity v2.0 endpoint, instantiate a `keystoneclient.v_20.client.Client` object:


```
from os import environ as env
import keystoneclient.v2_0.client as ksclient
keystone = ksclient.Client(auth_url=env['OS_AUTH_URL'],
                           username=env['OS_USERNAME'],
                           password=env['OS_PASSWORD'],
                           tenant_name=env['OS_TENANT_NAME'],
                           region_name=env['OS_REGION_NAME'])
```

After you instantiate a `Client` object, you can retrieve the token by accessing its `auth_token` attribute object:

```
import keystoneclient.v2_0.client as ksclient
keystone = ksclient.Client(...)
print keystone.auth_token
```

If the OpenStack cloud is configured to use public-key infrastructure (PKI) tokens, the Python script output looks something like this:

```
MIIQYQYJKoZIhvcNAQcCoIIQQjCCED4CAQEExCTAHBgUrDgMCGjCCDqcGCSqGSIB3DQEHAaCCDpgE
gg6UeyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3NlZWRFYXQiOiAiMjAxMy0xMC0yMFQxNj01NjoyNi
4zNTg2MjUiLCAiZXhwaXJlcyl6IClYMDZlTEwLTixVDE2OjU2OjI2WiIsICJpZCI6ICJwbGFjZWWh
...
R3g14FJ0BxtTPbo6WarZ+sA3PZwdgIDyGNI-00qv-8ih4gJC9C6wBCelldUXJ0Mn7BN-SfuxkooV66
e090bcKjTWet3CC8IEj7a6LyLRVTdvmKGA5-pgp2mS5fb3G2mIad4Zeeb-zQn9V3Xf9WUGxuiVulHn
fhuUpJT-s9mU7+WEC3-8qkcBjEpqVCvMpmM4INI=
```



Note

This example shows a subset of a PKI token. A complete token is over 5000 characters long.

Authenticate against an Image Service endpoint

To authenticate against an Image Service endpoint, instantiate a `glanceclient.v2.client.Client` object:

```
from os import environ as env
import glanceclient.v2.client as glclient
import keystoneclient.v2_0.client as ksclient

keystone = ksclient.Client(auth_url=env['OS_AUTH_URL'],
                           username=env['OS_USERNAME'],
                           password=env['OS_PASSWORD'],
                           tenant_name=env['OS_TENANT_NAME'],
                           region_name=env['OS_REGION_NAME'])

glance_endpoint = keystone.service_catalog.url_for(service_type='image')
glance = glclient.Client(glance_endpoint, token=keystone.auth_token)
```

Authenticate against a Compute endpoint

To authenticate against a Compute endpoint, instantiate a `novaclient.v_1_1.client.Client` object:

```
from os import environ as env
import novaclient.v1_1.client as nvclient
nova = nvclient.Client(auth_url=env['OS_AUTH_URL'],
                      username=env['OS_USERNAME'],
                      api_key=env['OS_PASSWORD'],
                      project_id=env['OS_TENANT_NAME'],
                      region_name=env['OS_REGION_NAME'])
```

Alternatively, you can instantiate a `novaclient.client.Client` object and pass the version number:

```
from os import environ as env
import novaclient
nova = novaclient.client.Client("1.1", auth_url=env['OS_AUTH_URL'],
                              username=env['OS_USERNAME'],
                              api_key=env['OS_PASSWORD'],
                              project_id=env['OS_TENANT_NAME'],
                              region_name=env['OS_REGION_NAME'])
```

If you authenticate against an endpoint that uses a custom authentication back end, you must load the authentication plug-in and pass it to the constructor.

The Rackspace public cloud is an OpenStack deployment that uses a custom authentication back end. To authenticate against this cloud, you must install the [rackspace-novaclient](#) library that contains the Rackspace authentication plug-in, called `rackspace`. The following Python code shows the additional modifications required to instantiate a `Client` object that can authenticate against the Rackspace custom authentication back end.

```
import novaclient.auth_plugin
import novaclient.v1_1.client as nvclient
from os import environ as env
auth_system = 'rackspace'
auth_plugin = novaclient.auth_plugin.load_plugin('rackspace')
nova = nvclient.Client(auth_url=env['OS_AUTH_URL'],
                      username=env['OS_USERNAME'],
                      api_key=env['OS_PASSWORD'],
                      project_id=env['OS_TENANT_NAME'],
                      region_name=env['OS_REGION_NAME'],
                      auth_system='rackspace',
                      auth_plugin=auth_plugin)
```

If you set the `OS_AUTH_SYSTEM` environment variable, check for this variable in your Python script to determine whether you need to load a custom authentication back end:

```
import novaclient.auth_plugin
import novaclient.v1_1.client as nvclient
from os import environ as env
auth_system = os.get('OS_AUTH_SYSTEM')
if auth_system and auth_system != "keystone":
    auth_plugin = novaclient.auth_plugin.load_plugin(auth_system)
else:
    auth_plugin = None
nova = nvclient.Client(auth_url=env['OS_AUTH_URL'],
                      username=env['OS_USERNAME'],
                      api_key=env['OS_PASSWORD'],
                      project_id=env['OS_TENANT_NAME'],
                      region_name=env['OS_REGION_NAME'],
                      auth_system=auth_system,
                      auth_plugin=auth_plugin)
```

Authenticate against a Networking endpoint

To authenticate against a Networking endpoint, instantiate a `neutronclient.v2_0.client.Client` object:

```
from os import environ as env
from neutronclient.v2_0 import client as neutronclient
neutron = neutronclient.Client(auth_url=env['OS_AUTH_URL'],
                               username=env['OS_USERNAME'],
                               password=env['OS_PASSWORD'],
                               tenant_name=env['OS_TENANT_NAME'],
                               region_name=env['OS_REGION_NAME'])
```

You can also authenticate by explicitly specifying the endpoint and token:

```
from os import environ as env
import keystoneclient.v2_0.client as ksclient
from neutronclient.v2_0 import client as neutronclient
keystone = ksclient.Client(auth_url=env['OS_AUTH_URL'],
                           username=env['OS_USERNAME'],
                           password=env['OS_PASSWORD'],
                           tenant_name=env['OS_TENANT_NAME'],
                           region_name=env['OS_REGION_NAME'])
endpoint_url = keystone.service_catalog.url_for(service_type='network')
token = keystone.auth_token
neutron = neutronclient.Client(endpoint_url=endpoint_url, token=token)
```

Manage images

When working with images in the SDK, you will call both `glance` and `nova` methods.

List images

To list the available images, call the `glanceclient.v2.images.Controller.list` method:

```
import glanceclient.v2.client as glclient
glance = glclient.Client(...)
images = glance.images.list()
```

The `images` method returns a Python generator, as shown in the following interaction with the Python interpreter:

```
>>> images = glance.images.list()
>>> images
<generator object list at 0x105e9c2d0>
>>> list(images)
[{'checksum': u'f8a2eeee2dc65b3d9b6e63678955bd83',
  'container_format': u'ami',
  'created_at': u'2013-10-20T14:28:10Z',
  'disk_format': u'ami',
  'file': u'/v2/images/dbc9b2db-51d7-403d-b680-3f576380b00c/file',
  'id': u'dbc9b2db-51d7-403d-b680-3f576380b00c',
  'kernel_id': u'c002c82e-2cfa-4952-8461-2095b69c18a6',
  'min_disk': 0,
  'min_ram': 0,
```

```
u'name': u'cirros-0.3.2-x86_64-uec',
u'protected': False,
u'ramdisk_id': u'4c1c9b4f-3fe9-425a-alec-1d8fd90b4db3',
u'schema': u'/v2/schemas/image',
u'size': 25165824,
u'status': u'active',
u'tags': [],
u'updated_at': u'2013-10-20T14:28:11Z',
u'visibility': u'public'},
{'u'checksum': u'69c33642f44ca552ba4bb8b66ad97e85',
u'container_format': u'ari',
u'created_at': u'2013-10-20T14:28:09Z',
u'disk_format': u'ari',
u'file': u'/v2/images/4c1c9b4f-3fe9-425a-alec-1d8fd90b4db3/file',
u'id': u'4c1c9b4f-3fe9-425a-alec-1d8fd90b4db3',
u'min_disk': 0,
u'min_ram': 0,
u'name': u'cirros-0.3.2-x86_64-uec-ramdisk',
u'protected': False,
u'schema': u'/v2/schemas/image',
u'size': 3714968,
u'status': u'active',
u'tags': [],
u'updated_at': u'2013-10-20T14:28:10Z',
u'visibility': u'public'},
{'u'checksum': u'c352f4e7121c6eae958bc1570324f17e',
u'container_format': u'aki',
u'created_at': u'2013-10-20T14:28:08Z',
u'disk_format': u'aki',
u'file': u'/v2/images/c002c82e-2cfa-4952-8461-2095b69c18a6/file',
u'id': u'c002c82e-2cfa-4952-8461-2095b69c18a6',
u'min_disk': 0,
u'min_ram': 0,
u'name': u'cirros-0.3.2-x86_64-uec-kernel',
u'protected': False,
u'schema': u'/v2/schemas/image',
u'size': 4955792,
u'status': u'active',
u'tags': [],
u'updated_at': u'2013-10-20T14:28:09Z',
u'visibility': u'public'}]
```

Get image by ID

To retrieve an image object from its ID, call the `glanceclient.v2.images.Controller.get` method:

```
import glanceclient.v2.client as glclient
image_id = 'c002c82e-2cfa-4952-8461-2095b69c18a6'
glance = glclient.Client(...)
image = glance.images.get(image_id)
```

Get image by name

The Image Service Python bindings do not support the retrieval of an image object by name. However, the Compute Python bindings enable you to get an image object by name. To get an image object by name, call the `novaclient.v1_1.images.ImageManager.find` method:

```
import novaclient.v1_1.client as nvclient
name = "cirros"
nova = nvclient.Client(...)
image = nova.images.find(name=name)
```

Upload an image

To upload an image, call the `glanceclient.v2.images.ImageManager.create` method:

```
import glanceclient.v2.client as glclient
imagefile = "/tmp/myimage.img"
glance = glclient.Client(...)
with open(imagefile) as fimage:
    glance.images.create(name="myimage", is_public=True, disk_format="qcow2",
                        container_format="bare", data=fimage)
```

Configure access and security for instances

When working with images in the SDK, you will call `novaclient` methods.

Add a keypair

To generate a keypair, call the `novaclient.v1_1.keypairs.KeypairManager.create` method:

```
import novaclient.v1_1.client as nvclient
nova = nvclient.Client(...)
keypair_name = "staging"
keypair = nova.keypairs.create(name=keypair_name)
print keypair.private_key
```

The Python script output looks something like this:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAE8XkaMqInSPfy0hMfWO+OZRtIgrQAbQkNcaNHmv2GN2G6xZ1b\nuBRux5Xk/6SZ
ABaNPmlnRWm/ZDHnxCsFTcAl2LYOQXx3Cl2qKNY4r2di4G48GAkd\n7k51DP2RgQatUM8npO0CD9PU
...
mmrceYYK08/lQ7JKLmVkdzdQKt77+v1oBBuHiykLfI6h1m77NRDw9r8cV\nzcyYeoALifpJTPMKKS8
ECfDCuDn/vc9K1He8CRaJHF8AMLQLM3MN
-----END RSA PRIVATE KEY-----
```

You typically write the private key to a file to use it later. The file must be readable and writable by only the file owner; otherwise, the SSH client will refuse to read the private key file. It is safest to create the file with the appropriate permissions, as shown in the following example:

```
import novaclient.v1_1.client as nvclient
import os
nova = nvclient.Client(...)
keypair_name = "staging"
private_key_filename = "/home/alice/id-staging"
keypair = nova.keypairs.create(name=keypair_name)

# Create a file for writing that can only be read and written by owner
fp = os.open(private_key_filename, os.O_WRONLY | os.O_CREAT, 0o600)
with os.fdopen(fp, 'w') as f:
    f.write(keypair.private_key)
```

Import a keypair

If you have already generated a keypair with the public key located at `~/.ssh/id_rsa.pub`, pass the contents of the file to the `novaclient.v1_1.keypairs.KeypairManager.create` method to import the public key to Compute:

```
import novaclient.v1_1.client as nvclient
import os.path
with open(os.path.expanduser('~/.ssh/id_rsa.pub')) as f:
    public_key = f.read()
nova = nvclient.Client(...)
nova.keypairs.create('mykey', public_key)
```

List keypairs

To list keypairs, call the `novaclient.v1_1.keypairs.KeypairManager.list` method:

```
import novaclient.v1_1.client as nvclient
nova = nvclient.Client(...)
keypairs = nova.keypairs.list()
```

Create and manage security groups

To list security groups for the current project, call the `novaclient.v1_1.security_groups.SecurityGroupManager.list` method:

```
import novaclient.v1_1.client as nvclient
nova = nvclient.Client(...)
security_groups = nova.security_groups.list()
```

To create a security group with a specified name and description, call the `novaclient.v1_1.security_groups.SecurityGroupManager.create` method:

```
import novaclient.v1_1.client as nvclient
nova = nvclient.Client(...)
nova.security_groups.create(name="web", description="Web servers")
```

To delete a security group, call the `novaclient.v1_1.security_groups.SecurityGroupManager.delete` method, passing either a `novaclient.v1_1.security_groups.SecurityGroup` object or group ID as an argument:

```
import novaclient.v1_1.client as nvclient
nova = nvclient.Client(...)
group = nova.security_groups.find(name="web")
nova.security_groups.delete(group)
# The following lines would also delete the group:
# nova.security_groups.delete(group.id)
# group.delete()
```

Create and manage security group rules

Access the security group rules from the `rules` attribute of a `novaclient.v1_1.security_groups.SecurityGroup` object:

```
import novaclient.v1_1.client as nvclient
nova = nvclient.Client(...)
group = nova.security_groups.find(name="web")
print group.rules
```

To add a rule, to a security group, call the [novaclient.v1_1.security_group_rules.SecurityGroupRuleManager.create](#) method:

```
import novaclient.v1_1.client as nvclient
nova = nvclient.Client(...)
group = nova.security_groups.find(name="web")
# Add rules for ICMP, tcp/80 and tcp/443
nova.security_group_rules.create(group.id, ip_protocol="icmp",
                                from_port=-1, to_port=-1)
nova.security_group_rules.create(group.id, ip_protocol="tcp",
                                from_port=80, to_port=80)
nova.security_group_rules.create(group.id, ip_protocol="tcp",
                                from_port=443, to_port=443)
```

Networking

To use the information in this section, you should have a general understanding of OpenStack Networking, OpenStack Compute, and the integration between the two. You should also have access to a plug-in that implements the Networking API v2.0.

Set environment variables

Make sure that you set the relevant environment variables.

As an example, see the sample shell file that sets these variables to get credentials:

```
export OS_USERNAME="admin"
export OS_PASSWORD="password"
export OS_TENANT_NAME="admin"
export OS_AUTH_URL="http://IPADDRESS/v2.0"
```

Get credentials

The examples in this section use the `get_credentials` method:

```
def get_credentials():
    d = {}
    d['username'] = os.environ['OS_USERNAME']
    d['password'] = os.environ['OS_PASSWORD']
    d['auth_url'] = os.environ['OS_AUTH_URL']
    d['tenant_name'] = os.environ['OS_TENANT_NAME']
    return d
```

This code resides in the `credentials.py` file, which all samples import.

Use the `get_credentials()` method to populate and get a dictionary:

```
credentials = get_credentials()
```

Get Nova credentials

Few examples in this section use the `get_nova_credentials` method:

```
def get_nova_credentials():
    d = {}
    d['username'] = os.environ['OS_USERNAME']
    d['api_key'] = os.environ['OS_PASSWORD']
    d['auth_url'] = os.environ['OS_AUTH_URL']
    d['project_id'] = os.environ['OS_TENANT_NAME']
    return d
```

This code resides in the `credentials.py` file, which all samples import.

Use the `get_nova_credentials()` method to populate and get a dictionary:

```
nova_credentials = get_nova_credentials()
```

Print values

The examples in this section use the `print_values` and `print_values_server` methods:

```
def print_values(val, type):
    if type == 'ports':
        val_list = val['ports']
    if type == 'networks':
        val_list = val['networks']
    if type == 'routers':
        val_list = val['routers']
    for p in val_list:
        for k, v in p.items():
            print("%s : %s" % (k, v))
        print('\n')
```

```
def print_values_server(val, server_id, type):
    if type == 'ports':
        val_list = val['ports']

    if type == 'networks':
        val_list = val['networks']
    for p in val_list:
        bool = False
        for k, v in p.items():
            if k == 'device_id' and v == server_id:
                bool = True
        if bool:
            for k, v in p.items():
                print("%s : %s" % (k, v))
            print('\n')
```

This code resides in the `utils.py` file, which all samples import.

Create network

The following program creates a network:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
```



```
from neutronclient.v2_0 import client
from credentials import get_credentials

network_name = 'sample_network'
credentials = get_credentials()
neutron = client.Client(**credentials)
try:
    body_sample = {'network': {'name': network_name,
                               'admin_state_up': True}}

    netw = neutron.create_network(body=body_sample)
    net_dict = netw['network']
    network_id = net_dict['id']
    print('Network %s created' % network_id)

    body_create_subnet = {'subnets': [{'cidr': '192.168.199.0/24',
                                         'ip_version': 4, 'network_id': network_id}]}

    subnet = neutron.create_subnet(body=body_create_subnet)
    print('Created subnet %s' % subnet)
finally:
    print("Execution completed")
```

List networks

The following program lists networks:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from neutronclient.v2_0 import client
from credentials import get_credentials
from utils import print_values

credentials = get_credentials()
neutron = client.Client(**credentials)
netw = neutron.list_networks()

print_values(netw, 'networks')
```

For `print_values` see [the section called "Print values" \[104\]](#).

Create ports

The following program creates a port:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from neutronclient.v2_0 import client
import novaclient.v1_1.client as nvclient
from credentials import get_credentials
from credentials import get_nova_credentials

credentials = get_nova_credentials()
nova_client = nvclient.Client(**credentials)

# Replace with server_id and network_id from your environment
```

```
server_id = '9a52795a-a70d-49a8-a5d0-5b38d78bd12d'
network_id = 'ce5d204a-93f5-43ef-bd89-3ab99ad09a9a'
server_detail = nova_client.servers.get(server_id)
print(server_detail.id)

if server_detail != None:
    credentials = get_credentials()
    neutron = client.Client(**credentials)

    body_value = {
        "port": {
            "admin_state_up": True,
            "device_id": server_id,
            "name": "port1",
            "network_id": network_id
        }
    }
    response = neutron.create_port(body=body_value)
    print(response)
```

For `get_nova_credentials` see [the section called "Get Nova credentials" \[103\]](#).

For `get_credentials` see [the section called "Get credentials" \[103\]](#).

List ports

The following program lists ports:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from neutronclient.v2_0 import client
from credentials import get_credentials
from utils import print_values

credentials = get_credentials()
neutron = client.Client(**credentials)
ports = neutron.list_ports()
print_values(ports, 'ports')
```

For `get_credentials` see [the section called "Get credentials" \[103\]](#).

For `print_values` see [the section called "Print values" \[104\]](#).

List server ports

The following program lists the ports for a server:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from neutronclient.v2_0 import client
import novaclient.v1_1.client as nvclient
from credentials import get_credentials
from credentials import get_nova_credentials
from utils import print_values_server

credentials = get_nova_credentials()
```

```
nova_client = nvclient.Client(**credentials)

# change these values according to your environment

server_id = '9a52795a-a70d-49a8-a5d0-5b38d78bd12d'
network_id = 'ce5d204a-93f5-43ef-bd89-3ab99ad09a9a'
server_detail = nova_client.servers.get(server_id)
print(server_detail.id)

if server_detail is not None:
    credentials = get_credentials()
    neutron = client.Client(**credentials)
    ports = neutron.list_ports()

    print_values_server(ports, server_id, 'ports')
    body_value = {'port': {
        'admin_state_up': True,
        'device_id': server_id,
        'name': 'port1',
        'network_id': network_id,
    }}

    response = neutron.create_port(body=body_value)
    print(response)
```

Create router and add port to subnet

This example queries OpenStack Networking to create a router and add a port to a subnet.

To create a router and add a port to a subnet

1. Import the following modules:

```
from neutronclient.v2_0 import client
import novaclient.v1_1.client as nvclient
from credentials import get_credentials
from credentials import get_nova_credentials
from utils import print_values_server
```

2. Get Nova Credentials. See [the section called "Get Nova credentials" \[103\]](#).
3. Instantiate the `nova_client` client object by using the `credentials` dictionary object:

```
nova_client = nvclient.Client(**credentials)
```

4. Create a router and add a port to the subnet:

```
# Replace with server_id and network_id from your environment

router_id = '72cf1682-60a8-4890-b0ed-6bad7d9f5466'
network_id = '81bf592a-9e3f-4f84-a839-ae87df188dc1'

credentials = get_credentials()
neutron = client.Client(**credentials)
router = neutron.show_router(router_id)
print(router)
body_value = {'port': {
    'admin_state_up': True,
```

```
'device_id': router_id,
'name': 'port1',
'network_id': network_id,
}}

response = neutron.create_port(body=body_value)
print(response)
print("Execution Completed")
```

Example 3.1. Create router: complete code listing

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from neutronclient.v2_0 import client
import novaclient.v1_1.client as nvclient
from credentials import get_credentials
from credentials import get_nova_credentials
from utils import print_values_server

credentials = get_nova_credentials()
nova_client = nvclient.Client(**credentials)

# Replace with server_id and network_id from your environment

router_id = '72cf1682-60a8-4890-b0ed-6bad7d9f5466'
network_id = '81bf592a-9e3f-4f84-a839-ae87df188dc1'
try:
    credentials = get_credentials()
    neutron = client.Client(**credentials)
    router = neutron.show_router(router_id)
    print(router)
    body_value = {'port': {
        'admin_state_up': True,
        'device_id': router_id,
        'name': 'port1',
        'network_id': network_id,
    }}

    response = neutron.create_port(body=body_value)
    print(response)
finally:
    print("Execution completed")
```

Delete a network

This example queries OpenStack Networking to delete a network.

To delete a network

1. Import the following modules:

```
from neutronclient.v2_0 import client
from credentials import get_credentials
```

2. Get credentials. See [the section called "Get credentials" \[103\]](#).
3. Instantiate the neutron client object by using the credentials dictionary object:

```
neutron = client.Client(**credentials)
```

4. Delete the network:

```
body_sample = {'network': {'name': network_name,
                           'admin_state_up': True}}

netw = neutron.create_network(body=body_sample)
net_dict = netw['network']
network_id = net_dict['id']
print('Network %s created' % network_id)

body_create_subnet = {'subnets': [{'cidr': '192.168.199.0/24',
                                     'ip_version': 4, 'network_id': network_id}]}

subnet = neutron.create_subnet(body=body_create_subnet)
print('Created subnet %s' % subnet)

neutron.delete_network(network_id)
print('Deleted Network %s' % network_id)

print("Execution completed")
```

Example 3.2. Delete network: complete code listing

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from neutronclient.v2_0 import client
from credentials import get_credentials

network_name = 'temp_network'
credentials = get_credentials()
neutron = client.Client(**credentials)
try:
    body_sample = {'network': {'name': network_name,
                               'admin_state_up': True}}

    netw = neutron.create_network(body=body_sample)
    net_dict = netw['network']
    network_id = net_dict['id']
    print('Network %s created' % network_id)

    body_create_subnet = {'subnets': [{'cidr': '192.168.199.0/24',
                                         'ip_version': 4, 'network_id': network_id}]}

    subnet = neutron.create_subnet(body=body_create_subnet)
    print('Created subnet %s' % subnet)

    neutron.delete_network(network_id)
    print('Deleted Network %s' % network_id)
finally:
    print("Execution Completed")
```

List routers

This example queries OpenStack Networking to list all routers.

To list routers

1. Import the following modules:

```
from neutronclient.v2_0 import client
from credentials import get_credentials
from utils import print_values
```

2. Get credentials. See [the section called "Get credentials" \[103\]](#).
3. Instantiate the neutron client object by using the credentials dictionary object:

```
neutron = client.Client(**credentials)
```

4. List the routers

```
routers_list = neutron.list_routers(retrieve_all=True)
print_values(routers_list, 'routers')
print("Execution completed")
```

For print_values see [the section called "Print values" \[104\]](#).

Example 3.3. List routers: complete code listing

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from neutronclient.v2_0 import client
from credentials import get_credentials
from utils import print_values

try:
    credentials = get_credentials()
    neutron = client.Client(**credentials)
    routers_list = neutron.list_routers(retrieve_all=True)
    print_values(routers_list, 'routers')
finally:
    print("Execution completed")
```

List security groups

This example queries OpenStack Networking to list security groups.

To list security groups

1. Import the following modules:

```
from neutronclient.v2_0 import client
from credentials import get_credentials
from utils import print_values
```

2. Get credentials. See [the section called "Get credentials" \[103\]](#).
3. Instantiate the neutron client object by using the credentials dictionary object:

```
neutron = client.Client(**credentials)
```

4. List Security groups

```
sg = neutron.list_security_groups()
print(sg)
```

Example 3.4. List security groups: complete code listing

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from neutronclient.v2_0 import client
from credentials import get_credentials
from utils import print_values

credentials = get_credentials()
neutron = client.Client(**credentials)
sg = neutron.list_security_groups()
print(sg)
```



Note

OpenStack Networking security groups are case-sensitive while the nova-network security groups are case-insensitive.

List subnets

This example queries OpenStack Networking to list subnets.

To list subnets

1. Import the following modules:

```
from neutronclient.v2_0 import client
from credentials import get_credentials
from utils import print_values
```

2. Get credentials. See [the section called "Get credentials" \[103\]](#).
3. Instantiate the neutron client object by using the credentials dictionary object:

```
neutron = client.Client(**credentials)
```

4. List subnets:

```
subnets = neutron.list_subnets()
print(subnets)
```

Example 3.5. List subnets: complete code listing

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from neutronclient.v2_0 import client
from credentials import get_credentials
from utils import print_values

credentials = get_credentials()
neutron = client.Client(**credentials)
subnets = neutron.list_subnets()
print(subnets)
```

Compute

To use the information in this section, you should have a general understanding of OpenStack Compute.

Set environment variables

Please see [the section called “Authenticate” \[96\]](#) on how to setup environmental variables and authenticate against Compute API endpoints.

Get nova credentials v2

The examples in this section use the `get_nova_credentials_v2` method:

```
def get_nova_credentials_v2():
    d = {}
    d['version'] = '2'
    d['username'] = os.environ['OS_USERNAME']
    d['api_key'] = os.environ['OS_PASSWORD']
    d['auth_url'] = os.environ['OS_AUTH_URL']
    d['project_id'] = os.environ['OS_TENANT_NAME']
    return d
```

This code resides in the `credentials.py` file, which all samples import.

Use the `get_nova_credentials_v2()` method to populate and get a dictionary:

```
credentials = get_nova_credentials_v2()
```

List servers v2

The following program lists servers using the v2 APIs:

To list the servers

1. Import the following modules:

```
from credentials import get_nova_credentials_v2
from novaclient.client import Client
```

2. Get Nova Credentials. See [the section called “Get nova credentials v2” \[112\]](#).
3. Instantiate the `nova_client` client object by using the `credentials` dictionary object:

```
nova_client = Client(**credentials)
```

4. Get the list of servers by calling `servers.list` on `nova_client` object:

```
print(nova_client.servers.list())
```


Example 3.6. List servers: complete code listing

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from credentials import get_nova_credentials_v2
from novaclient.client import Client

credentials = get_nova_credentials_v2()
nova_client = Client(**credentials)

print(nova_client.servers.list())
```

Create a server v2

The following program creates a server (VM) using the v2 APIs:

To create a server

1. Import the following modules:

```
import time
from credentials import get_nova_credentials_v2
from novaclient.client import Client
```

2. Get Nova Credentials. See [the section called "Get nova credentials v2" \[112\]](#).
3. Instantiate the `nova_client` client object by using the `credentials` dictionary object:

```
nova_client = Client(**credentials)
```

4. In this step, search for the flavor and image to be used for creating a server. The following code assumes `cirros` image and `m1.tiny` are being used.

```
image = nova_client.images.find(name="cirros")
flavor = nova_client.flavors.find(name="m1.tiny")
```

- 5.
6. In this step determine the network with which the server is going to be attached. Use this along with flavor and image to create the server.

```
net_id = 'd05a7729-4bcf-4149-9d8f-6a4764520a04'
nic_d = [{'net-id': net_id}]
instance = nova_client.servers.create(name="vm2", image=image,
                                       flavor=flavor, key_name="keypair-1",
                                       nics=nic_d)
```

7. Sleep for 5 secs and check if the server/vm got created by calling `nova_client.servers.list()`

```
print("Sleeping for 5s after create command")
time.sleep(5)
print("List of VMs")
print(nova_client.servers.list())
```

Example 3.7. Create server: complete code listing

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import time
from credentials import get_nova_credentials_v2
from novaclient.client import Client

try:
    credentials = get_nova_credentials_v2()
    nova_client = Client(**credentials)

    image = nova_client.images.find(name="cirros")
    flavor = nova_client.flavors.find(name="m1.tiny")
    net_id = 'd05a7729-4bcf-4149-9d8f-6a4764520a04'
    nic_d = [{'net-id': net_id}]
    instance = nova_client.servers.create(name="vm2", image=image,
                                          flavor=flavor, key_name="keypair-1",
                                          nics=nic_d)
    print("Sleeping for 5s after create command")
    time.sleep(5)
    print("List of VMs")
    print(nova_client.servers.list())
finally:
    print("Execution Completed")
```

Delete server v2

The following program deletes a Server (VM) using the v2 API:

To Delete a Server

1. Import the following modules:

```
import time
from credentials import get_nova_credentials_v2
from novaclient.client import Client
```

2. Get Nova Credentials. See [the section called "Get nova credentials v2" \[112\]](#).
3. Instantiate the `nova_client` client object by using the `credentials` dictionary object:

```
nova_client = Client(**credentials)
```

4. Check if the server "vm1" exists using the following steps
 1. Get the list of servers: `servers_list`
 2. Iterate over the `servers_list` and compare name with "vm1"
 3. If true set the variable name `server_exists` as True and break from the for loop

```
servers_list = nova_client.servers.list()
server_del = "vml"
server_exists = False

for s in servers_list:
    if s.name == server_del:
        print("This server %s exists" % server_del)
        server_exists = True
        break
```

5. If the server exists execute delete method of `nova_client.servers` object.

```
nova_client.servers.delete(s)
```

Example 3.8. Delete: complete code listing

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from credentials import get_nova_credentials_v2
from novaclient.client import Client

credentials = get_nova_credentials_v2()
nova_client = Client(**credentials)

servers_list = nova_client.servers.list()
server_del = "vml"
server_exists = False

for s in servers_list:
    if s.name == server_del:
        print("This server %s exists" % server_del)
        server_exists = True
        break
if not server_exists:
    print("server %s does not exist" % server_del)
else:
    print("deleting server.....")
    nova_client.servers.delete(s)
    print("server %s deleted" % server_del)
```

Appendix A. Community support

Table of Contents

Documentation	116
ask.openstack.org	117
OpenStack mailing lists	117
The OpenStack wiki	118
The Launchpad Bugs area	118
The OpenStack IRC channel	119
Documentation feedback	119
OpenStack distribution packages	119

The following resources are available to help you run and use OpenStack. The OpenStack community constantly improves and adds to the main features of OpenStack, but if you have any questions, do not hesitate to ask. Use the following resources to get OpenStack support, and troubleshoot your installations.

Documentation

For the available OpenStack documentation, see docs.openstack.org.

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

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

- [Installation Guide for Debian 7.0](#)
- [Installation Guide for openSUSE and SUSE Linux Enterprise Server](#)
- [Installation Guide for Red Hat Enterprise Linux, CentOS, and Fedora](#)
- [Installation Guide for Ubuntu 12.04/14.04 \(LTS\)](#)

The following books explain how to configure and run an OpenStack cloud:

- [Cloud Administrator Guide](#)
- [Configuration Reference](#)
- [Operations Guide](#)
- [High Availability Guide](#)
- [Security Guide](#)

- [Virtual Machine Image Guide](#)

The following books explain how to use the OpenStack dashboard and command-line clients:

- [API Quick Start](#)
- [End User Guide](#)
- [Admin User Guide](#)
- [Command-Line Interface Reference](#)

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

- [OpenStack API Complete Reference \(HTML\)](#)
- [API Complete Reference \(PDF\)](#)
- [OpenStack Block Storage Service API v2 Reference](#)
- [OpenStack Compute API v2 and Extensions Reference](#)
- [OpenStack Identity Service API v2.0 Reference](#)
- [OpenStack Image Service API v2 Reference](#)
- [OpenStack Networking API v2.0 Reference](#)
- [OpenStack Object Storage API v1 Reference](#)

The [Training Guides](#) offer software training for cloud administration and management.

ask.openstack.org

During the set up or testing of OpenStack, you might have questions about how a specific task is completed or be in a situation where a feature does not work correctly. Use the ask.openstack.org site to ask questions and get answers. When you visit the <http://ask.openstack.org> site, scan the recently asked questions to see whether your question has already been answered. If not, ask a new question. Be sure to give a clear, concise summary in the title and provide as much detail as possible in the description. Paste in your command output or stack traces, links to screen shots, and any other information which might be useful.

OpenStack mailing lists

A great way to get answers and insights is to post your question or problematic scenario to the OpenStack mailing list. You can learn from and help others who might have similar issues. To subscribe or view the archives, go to <http://lists.openstack.org/cgi-bin/mailman/listinfo/openstack>. You might be interested in the other mailing lists for specific projects or development, which you can find [on the wiki](#). A description of all mailing lists is available at <http://wiki.openstack.org/MailingLists>.

The OpenStack wiki

The [OpenStack wiki](#) contains a broad range of topics but some of the information can be difficult to find or is a few pages deep. Fortunately, the wiki search feature enables you to search by title or content. If you search for specific information, such as about networking or nova, you can find a large amount of relevant material. More is being added all the time, so be sure to check back often. You can find the search box in the upper-right corner of any OpenStack wiki page.

The Launchpad Bugs area

The OpenStack community values your set up and testing efforts and wants your feedback. To log a bug, you must sign up for a Launchpad account at <https://launchpad.net/+login>. You can view existing bugs and report bugs in the Launchpad Bugs area. Use the search feature to determine whether the bug has already been reported or already been fixed. If it still seems like your bug is unreported, fill out a bug report.

Some tips:

- Give a clear, concise summary.
- Provide as much detail as possible in the description. Paste in your command output or stack traces, links to screen shots, and any other information which might be useful.
- Be sure to include the software and package versions that you are using, especially if you are using a development branch, such as, "Juno release" vs `git commit bc79c3ecc55929bac585d04a03475b72e06a3208`.
- Any deployment-specific information is helpful, such as whether you are using Ubuntu 14.04 or are performing a multi-node installation.

The following Launchpad Bugs areas are available:

- [Bugs: OpenStack Block Storage \(cinder\)](#)
- [Bugs: OpenStack Compute \(nova\)](#)
- [Bugs: OpenStack Dashboard \(horizon\)](#)
- [Bugs: OpenStack Identity \(keystone\)](#)
- [Bugs: OpenStack Image Service \(glance\)](#)
- [Bugs: OpenStack Networking \(neutron\)](#)
- [Bugs: OpenStack Object Storage \(swift\)](#)
- [Bugs: Bare Metal \(ironic\)](#)
- [Bugs: Data Processing Service \(sahara\)](#)
- [Bugs: Database Service \(trove\)](#)

- [Bugs: Orchestration \(heat\)](#)
- [Bugs: Telemetry \(ceilometer\)](#)
- [Bugs: Queue Service \(marconi\)](#)
- [Bugs: OpenStack API Documentation \(api.openstack.org\)](#)
- [Bugs: OpenStack Documentation \(docs.openstack.org\)](#)

The OpenStack IRC channel

The OpenStack community lives in the #openstack IRC channel on the Freenode network. You can hang out, ask questions, or get immediate feedback for urgent and pressing issues. To install an IRC client or use a browser-based client, go to <http://webchat.freenode.net/>. You can also use Colloquy (Mac OS X, <http://colloquy.info/>), mIRC (Windows, <http://www.mirc.com/>), or XChat (Linux). When you are in the IRC channel and want to share code or command output, the generally accepted method is to use a Paste Bin. The OpenStack project has one at <http://paste.openstack.org>. Just paste your longer amounts of text or logs in the web form and you get a URL that you can paste into the channel. The OpenStack IRC channel is #openstack on irc.freenode.net. You can find a list of all OpenStack IRC channels at <https://wiki.openstack.org/wiki/IRC>.

Documentation feedback

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

OpenStack distribution packages

The following Linux distributions provide community-supported packages for OpenStack:

- **Debian:** <http://wiki.debian.org/OpenStack>
- **CentOS, Fedora, and Red Hat Enterprise Linux:** <http://openstack.redhat.com/>
- **openSUSE and SUSE Linux Enterprise Server:** <http://en.opensuse.org/Portal:OpenStack>
- **Ubuntu:** <https://wiki.ubuntu.com/ServerTeam/CloudArchive>