



Red Hat Enterprise Linux OpenStack Platform 7 Instances and Images Guide

Managing Instances, Images, Volumes and Containers

OpenStack Team

Red Hat Enterprise Linux OpenStack Platform 7 Instances and Images Guide

Managing Instances, Images, Volumes and Containers

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Copyright © 2017 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

The Instances and Images guide provides procedures for the management of instances, images, volumes and containers of a Red Hat Enterprise Linux OpenStack Platform environment.

Table of Contents

PREFACE	3
CHAPTER 1. IMAGE SERVICE	4
1.1. IMAGE SERVICE: NEW FEATURES	4
1.2. MANAGE IMAGES	5
CHAPTER 2. CONFIGURE OPENSTACK COMPUTE STORAGE	20
2.1. ARCHITECTURE OVERVIEW	20
2.2. CONFIGURATION	21
CHAPTER 3. VIRTUAL MACHINE INSTANCES	24
3.1. MANAGE INSTANCES	24
3.2. MANAGE INSTANCE SECURITY	33
3.3. MANAGE FLAVORS	37
3.4. MANAGE HOST AGGREGATES	44
3.5. SCHEDULE HOSTS AND CELLS	47
3.6. EVACUATE INSTANCES	55
3.7. MANAGE INSTANCE SNAPSHOTS	58
CHAPTER 4. MANAGE VOLUMES	60
4.1. BASIC VOLUME USAGE AND CONFIGURATION	60
4.2. ADVANCED VOLUME CONFIGURATION	66
CHAPTER 5. MANAGE CONTAINERS	80
5.1. CREATE A CONTAINER	80
5.2. CREATE PSEUDO FOLDER FOR CONTAINER	80
5.3. UPLOAD AN OBJECT	80
5.4. COPY AN OBJECT	81
5.5. DELETE AN OBJECT	81
5.6. DELETE A CONTAINER	82
5.7. ERASURE CODING FOR OBJECT STORAGE SERVICE	82
5.8. SET OBJECT STORAGE AS A BACK END FOR THE IMAGE SERVICE	84
CHAPTER 6. CONFIGURE OPENSTACK TO USE AN NFS BACK END	86
6.1. CONFIGURE SELINUX	86
6.2. CONFIGURE THE SHARE	86
6.3. CREATE A NEW BACK END DEFINITION	87
6.4. CREATE A VOLUME TYPE FOR THE NFS BACK END	88
6.5. TEST THE NEW NFS BACK END	89
CHAPTER 7. CONFIGURE CPU PINNING WITH NUMA	91
7.1. COMPUTE NODE CONFIGURATION	92
7.2. SCHEDULER CONFIGURATION	93
7.3. AGGREGATE AND FLAVOR CONFIGURATION	93
APPENDIX A. IMAGE CONFIGURATION PARAMETERS	96

PREFACE

Red Hat Enterprise Linux OpenStack Platform (RHEL OpenStack Platform) provides the foundation to build a private or public Infrastructure-as-a-Service (IaaS) cloud on top of Red Hat Enterprise Linux. It offers a massively scalable, fault-tolerant platform for the development of cloud-enabled workloads.

This guide discusses procedures for creating and managing images, instances, volumes and containers. It also mentions the procedures for configuring the storage for instances and configuring the NFS back end for RHEL OpenStack Platform.

You can manage the cloud using either the OpenStack dashboard or the command-line clients. Most procedures can be carried out using either method; some of the more advanced procedures can only be executed on the command line. This guide provides procedures for the dashboard where possible.



Note

For the complete suite of documentation for RHEL OpenStack Platform, see [Red Hat Enterprise Linux OpenStack Platform Documentation](#)

CHAPTER 1. IMAGE SERVICE

This chapter discusses the steps you can follow to manage images and storage in RHEL OpenStack Platform.

A virtual machine image is a file that contains a virtual disk which has a bootable operating system installed on it. Virtual machine images are supported in different formats. The following are the formats available on RHEL OpenStack Platform:

- ✦ **RAW** - Unstructured disk image format.
- ✦ **QCOW2** - Disk format supported by QEMU emulator.
- ✦ **ISO** - Sector-by-sector copy of the data on a disk, stored in a binary file.
- ✦ **AKI** - Indicates an Amazon Kernel Image.
- ✦ **AMI** - Indicates an Amazon Machine Image.
- ✦ **ARI** - Indicates an Amazon RAMDisk Image.
- ✦ **VDI** - Disk format supported by VirtualBox virtual machine monitor and the QEMU emulator.
- ✦ **VHD** - Common disk format used by virtual machine monitors from VMWare, VirtualBox, and others.
- ✦ **VMDK** - Disk format supported by many common virtual machine monitors.

While we don't normally think of ISO as a virtual machine image format, since ISOs contain bootable filesystems with an installed operating system, you can treat them the same as you treat other virtual machine image files.

To download the official Red Hat Enterprise Linux cloud images, you require a valid Red Hat Enterprise Linux subscription:

- ✦ [Red Hat Enterprise Linux 7 KVM Guest Image](#)
- ✦ [Red Hat Enterprise Linux 6 KVM Guest Image](#)

1.1. IMAGE SERVICE: NEW FEATURES

With the RHEL OpenStack Platform 7 release, the following new features are available for the Image Service:

- ✦ **Image conversion** - Convert images by calling the task API while importing an image (only qcow/raw format conversion is available for the Kilo release).

As part of the import workflow, a plugin provides the image conversion. This plugin can be activated or deactivated based on the deployer configuration. Therefore, the deployer needs to specify the preferred format of images for the deployment.

Internally, Image service receives the bits of the image in a particular format. These bits are stored in a temporary location. The plugin is then triggered to convert the image to the target format, and moved to a final destination. When the task is finished, the temporary location is deleted. As a result, the format uploaded initially is not retained by the Image service.



Note

The conversion can be triggered only when **importing** an image (the old copy-from). It does not run when **uploading** an image. For example:

```
$ glance --os-image-api-version 2 task-create --type import -
-input '{"import_from_format": "qcow2", "import_from":
"http://127.0.0.1:8000/test.qcow2", "image_properties":
{"disk_format": "qcow2", "container_format": "bare"}}'
```

- ✱ **Image Introspection** - Every image format comes with a set of metadata embedded inside the image itself.

For example, a stream optimized **vmdk** would contain the following parameters:

```
$ head -20 so-disk.vmdk

# Disk DescriptorFile
version=1
CID=d5a0bce5
parentCID=ffffffff
createType="streamOptimized"

# Extent description
RDONLY 209714 SPARSE "generated-stream.vmdk"

# The Disk Data Base
#DDB

ddb.adapterType = "buslogic"
ddb.geometry.cylinders = "102"
ddb.geometry.heads = "64"
ddb.geometry.sectors = "32"
ddb.virtualHWVersion = "4"
```

By introspecting this *vmdk*, you can easily know that the *disk_type* is *streamOptimized*, and the *adapter_type* is *buslogic*. By doing this metadata extraction in the Image service, the administrator does not have to care about all of these metadata unless they want to override some of them. These metadata parameters are useful for the consumer of the image. In Compute, the workflow to instantiate a *streamOptimized* disk is totally different than the one to instantiate a *flat* disk. This new feature allows metadata extraction. You can achieve image introspection by calling the task API while importing the image.



Note

For the Kilo release, you can only introspect the *virtual_size* metadata parameter.

1.2. MANAGE IMAGES

The OpenStack Image service (glance) provides discovery, registration, and delivery services for disk and server images. It provides the ability to copy or snapshot a server image, and immediately store it away. Stored images can be used as a template to get new servers up and running quickly

and more consistently, than installing a server operating system and individually configuring additional services.



Note

To be able to see the parent image that a clone came from, `show_image_direct_url` must be set to `True` in `glance-api.conf`. For more information, see [Chapter 8. Image Service](#) in the Configuration Reference.

1.2.1. Create an Image

This section provides you with the steps to manually create OpenStack-compatible images in .qcow2 format using Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7 ISO files.

1.2.1.1. Use a KVM Guest Image With RHEL OpenStack Platform

You can use a ready RHEL KVM guest qcow2 image available at: [RHEL 7 KVM Guest Image](#) or [RHEL 6.6 KVM Guest Image](#). These images are configured with `cloud-init` and must take advantage of ec2-compatible metadata services for provisioning SSH keys in order to function properly.



Note

For the KVM guest images:

- ✦ The `root` account in the image is disabled, but `sudo` access is granted to a special user named `cloud-user`.
- ✦ There is no `root` password set for this image.

The `root` password is locked in `/etc/shadow` by placing `!!` in the second field.

For an OpenStack instance, it is recommended that you generate an ssh keypair from the OpenStack dashboard or command line and use that key combination to perform an SSH public authentication to the instance as root.

When the instance is launched, this public key will be injected to it. You can then authenticate using the private key downloaded while creating the keypair.

If you do not want to use keypairs, you can use the `admin` password that has been set using the [Inject an admin Password Into an Instance](#) procedure.

If you want to create custom Red Hat Enterprise Linux images, see [Create a Red Hat Enterprise Linux 7 Image](#) or [Create a Red Hat Enterprise Linux 6 Image](#).

1.2.1.2. Create Custom Red Hat Enterprise Linux Images

Prerequisites:

- ✦ Linux host machine to create an image. This can be any machine on which you can install and run the Linux packages.

- ✦ libvirt, virt-manager (run command **yum groupinstall -y @virtualization**). This installs all packages necessary for creating a guest operating system.
- ✦ Libguestfs tools (run command **yum install -y libguestfs-tools-c**). This installs a set of tools for accessing and modifying virtual machine images.
- ✦ A Red Hat Enterprise Linux 7 ISO file (see [RHEL 7.0 Binary DVD](#) or [RHEL 6.6 Binary DVD](#)).
- ✦ Text editor, if you want to change the **kickstart** files.



Note

In the following procedures, all commands with the **[user@host]#** prompt should be run on your host machine.

1.2.1.2.1. Create a Red Hat Enterprise Linux 7 Image

This section provides you with the steps to manually create an OpenStack-compatible image in .qcow2 format using a Red Hat Enterprise Linux 7 ISO file.

1. Start the installation using **virt-install** as shown below:

```
[user@host]# qemu-img create -f qcow2 rhel7.qcow2 8G
[user@host]# virt-install --virt-type kvm --name rhel7 --ram 2048 \
\
--cdrom /tmp/rhel-server-7.0-x86_64-dvd.iso \
--disk rhel7.qcow2,format=qcow2 \
--network=bridge:virbr0 --graphics vnc,listen=0.0.0.0 \
--noautoconsole --os-type=linux --os-variant=rhel7
```

This launches an instance and starts the installation process.



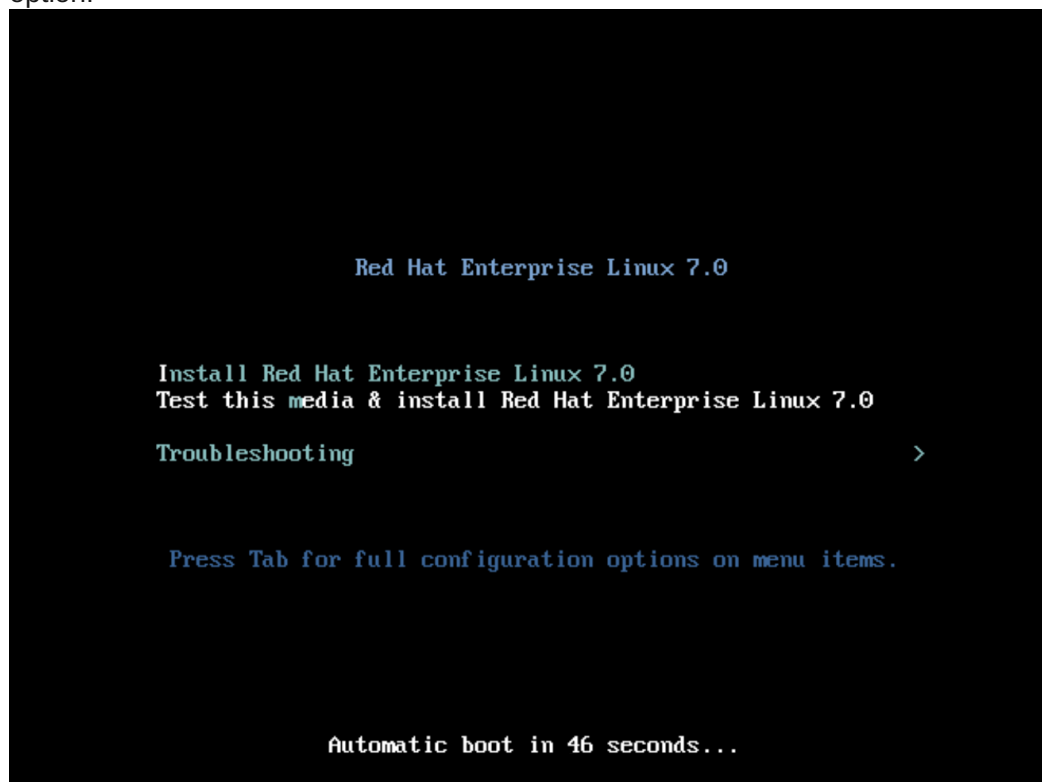
Note

If the instance does not launch automatically, run the following command to view the console:

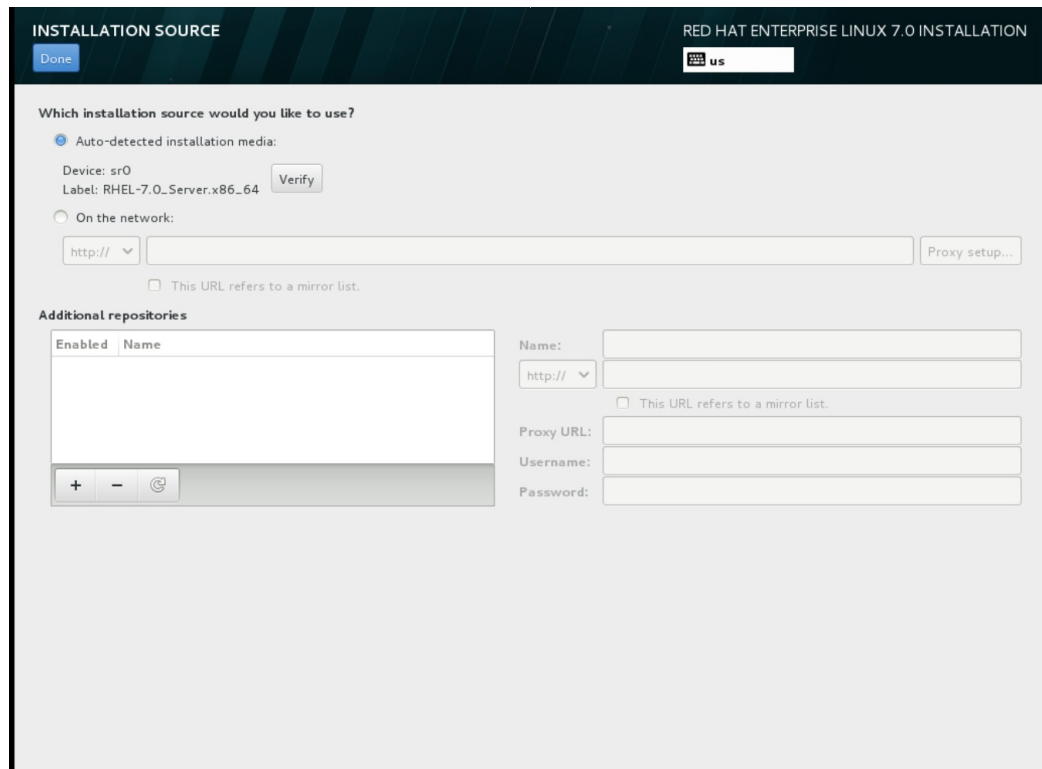
```
[user@host]# virt-viewer rhel7
```

2. Set up the virtual machine as follows:

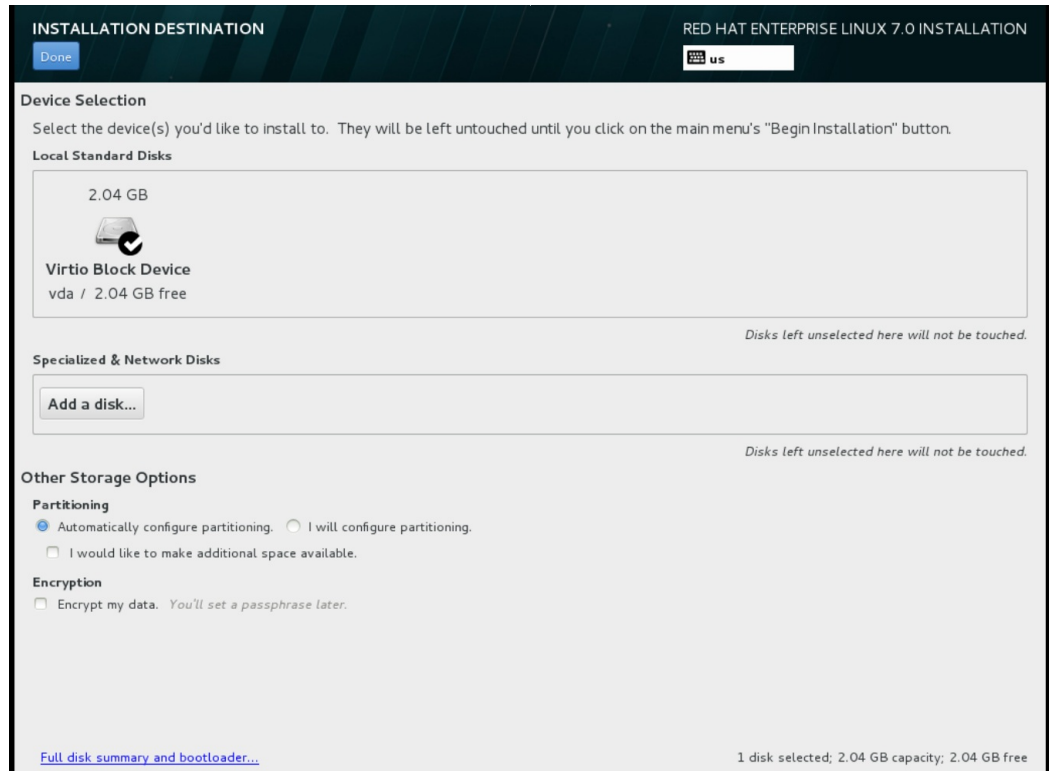
- a. At the initial Installer boot menu, choose the Install Red Hat Enterprise Linux 7.0 option.



- b. Choose the appropriate **Language** and **Keyboard** options.
- c. When prompted about which type of devices your installation uses, choose **Auto-detected installation media**.

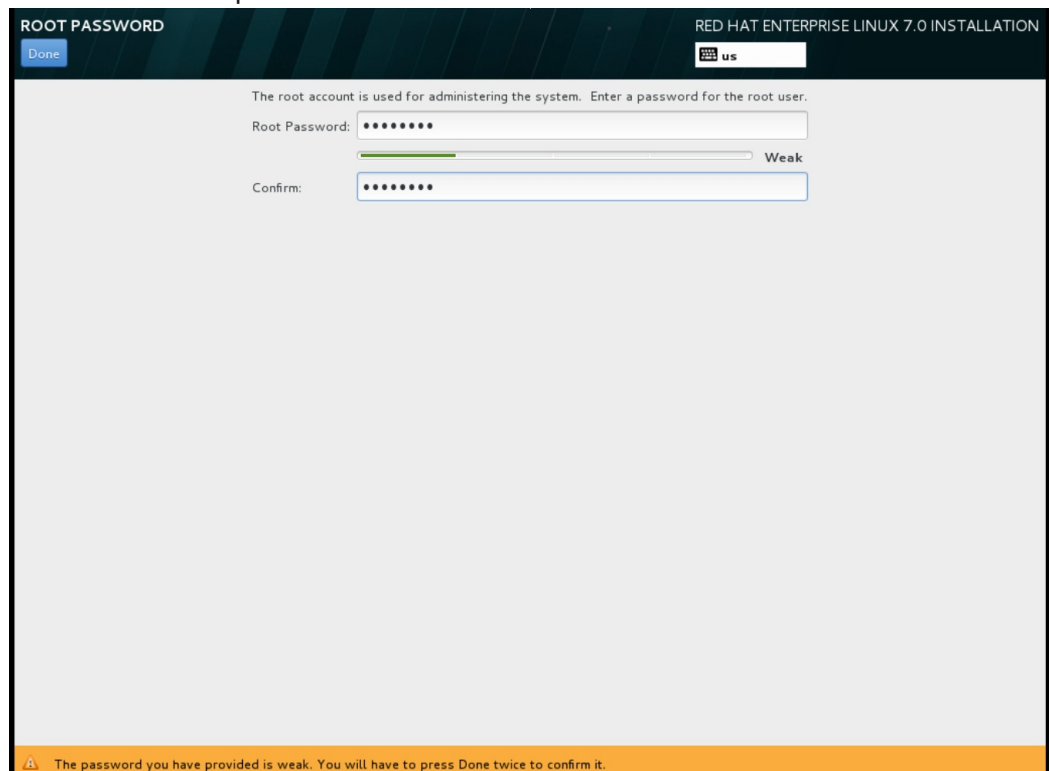


- d. When prompted about which type of installation destination, choose **Local Standard Disks**.



For other storage options, choose **Automatically configure partitioning**.

- e. For software selection, choose **Minimal Install**.
- f. For network and hostname, choose **eth0** for network and choose a **hostname** for your device. The default hostname is **localhost.localdomain**.
- g. Choose the **root** password.



The installation process completes and the **Complete!** screen appears.

3. After the installation is complete, reboot the instance and log in as the root user.

4. Update the `/etc/sysconfig/network-scripts/ifcfg-eth0` file so it only contains the following values:

```
TYPE=Ethernet
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
NM_CONTROLLED=no
```

5. Reboot the machine.
6. Register the machine with the Content Delivery Network:

```
# subscription-manager register
```

- a. Enter your Customer Portal user name and password when prompted:

```
Username: admin@example.com
Password:
```

- b. Find entitlement pools containing the channel:

```
# subscription-manager list --available | grep -A8 "Red Hat
Enterprise Linux Server"
```

- c. Use the pool identifiers located in the previous step to attach the **Red Hat Enterprise Linux Server** entitlement to the system:

```
# subscription-manager attach --pool=pool_id
```

- d. Enable the required channel:

```
# subscription-manager repos --enable=rhel-7-server-rpms
```

For RHEL OpenStack Platform 7, the required channels are **rhel-7-server-openstack-7.0-rpms** and **rhel-7-server-rh-common-rpms**.



Note

For more information, see "Subscribe to the Required Channels" in the *Installation Reference*.

7. Update the system.

```
# yum -y update
```

8. Install the **cloud-init** packages.

```
# yum install -y cloud-utils-growpart cloud-init
```

9. Edit the `/etc/cloud/cloud.cfg` configuration file and under `cloud_init_modules` add:

```
- resolv-conf
```

The `resolv-conf` option automatically configures the `resolv.conf` when an instance boots for the first time. This file contains information related to the instance such as `nameservers`, `domain` and other options.

10. Add the following line to `/etc/sysconfig/network` to avoid problems accessing the EC2 metadata service.

```
NOZEROCONF=yes
```

11. To ensure the console messages appear in the **Log** tab on the dashboard and the `nova console-log` output, add the following boot option to the ```/etc/default/grub`` file:

```
GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
```

Run the following command:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

The output is as follows:

```
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-229.7.2.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-229.7.2.el7.x86_64.img
Found linux image: /boot/vmlinuz-3.10.0-121.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-121.el7.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-
b82a3044fb384a3f9aeacf883474428b
Found initrd image: /boot/initramfs-0-rescue-
b82a3044fb384a3f9aeacf883474428b.img
done
```

12. Un-register the virtual machine so that the resulting image does not contain the same subscription details for every instance cloned based on it.

```
# subscription-manager repos --disable=*
# subscription-manager unregister
# yum clean all
```

13. Power off the instance:

```
# poweroff
```

14. Reset and clean the image using the `virt-sysprep` command so it can be to create instances without issues:

```
[user@host]# virt-sysprep -d rhel7
```

15. Reduce image size using the `virt-sparsify` command. This command converts any free space within the disk image back to free space within the host:

```
[user@host]# virt-sparsify --compress /tmp/rhel7.qcow2 rhel7-  
cloud.qcow2
```

This creates a new **rhel7-cloud.qcow2** file in the location from where the command is run.

The **rhel7-cloud.qcow2** image file is ready to be uploaded to the Image service. For more information on uploading this image to your OpenStack deployment using the dashboard, see [Upload an Image](#).

1.2.1.2.2. Create a Red Hat Enterprise Linux 6 Image

This section provides you with the steps to manually create an OpenStack-compatible image in .qcow2 format using a Red Hat Enterprise Linux 6 ISO file.

1. Start the installation using **virt-install**:

```
[user@host]# qemu-img create -f qcow2 rhel6.qcow2 4G  
[user@host]# virt-install --connect=qemu:///system --  
network=bridge:virbr0 \  
--name=rhel6.6 --os-type linux --os-variant rhel6 \  
--disk path=rhel6.qcow2,format=qcow2,size=10,cache=none \  
--ram 4096 --vcpus=2 --check-cpu --accelerate \  
--hvm --cdrom=rhel-server-6.6-x86_64-dvd.iso
```

This launches an instance and starts the installation process.



Note

If the instance does not launch automatically, run the following command to view the console:

```
[user@host]# virt-viewer rhel6
```

2. Set up the virtual machines as follows:

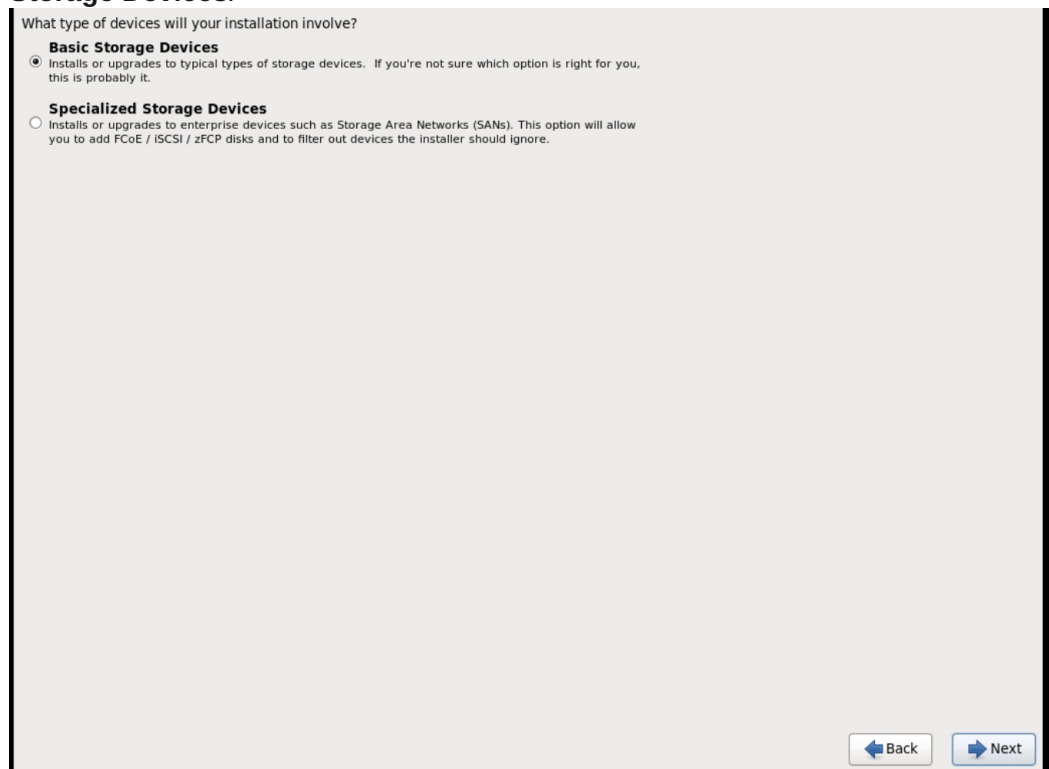
- a. At the initial Installer boot menu, choose the **Install or upgrade an existing system** option.



Step through the installation prompts. Accept the defaults.

The installation checks for disc and performs a **Media Check**. When the check is a **Success**, it ejects the disc.

- b. Choose the appropriate **Language** and **Keyboard** options.
- c. When prompted about which type of devices your installation uses, choose **Basic Storage Devices**.



- d. Choose a **hostname** for your device. The default hostname is **localhost.localdomain**.

- e. Set **timezone** and **root** password.
- f. Based on the space on the disk, choose the type of installation.

Which type of installation would you like?

Use All Space
Removes all partitions on the selected device(s). This includes partitions created by other operating systems.
Tip: This option will remove data from the selected device(s). Make sure you have backups.

Replace Existing Linux System(s)
Removes only Linux partitions (created from a previous Linux installation). This does not remove other partitions you may have on your storage device(s) (such as VFAT or FAT32).
Tip: This option will remove data from the selected device(s). Make sure you have backups.

Shrink Current System
Shrinks existing partitions to create free space for the default layout.

Use Free Space
Retains your current data and partitions and uses only the unpartitioned space on the selected device(s), assuming you have enough free space available.

Create Custom Layout
Manually create your own custom layout on the selected device(s) using our partitioning tool.

Encrypt system
 Review and modify partitioning layout

- g. Choose the **Basic Server** install, which installs an SSH server.

The default installation of Red Hat Enterprise Linux is a basic server install. You can optionally select a different set of software now.

Basic Server
 Database Server
 Web Server
 Identity Management Server
 Virtualization Host
 Desktop
 Software Development Workstation
 Minimal

Please select any additional repositories that you want to use for software installation.

High Availability
 Load Balancer
 Red Hat Enterprise Linux
 Red Hat OpenStack Platform

You can further customize the software selection now, or after install via the software management application.

Customize later Customize now

- h. The installation process completes and **Congratulations, your Red Hat Enterprise Linux installation is complete** screen appears.

3. Reboot the instance and log in as the **root** user.
4. Update the `/etc/sysconfig/network-scripts/ifcfg-eth0` file so it only contains the following values:

```
TYPE=Ethernet
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
NM_CONTROLLED=no
```

5. Reboot the machine.
6. Register the machine with the Content Delivery Network:

```
# subscription-manager register
```

- a. Enter your Customer Portal user name and password when prompted:

```
Username: admin@example.com
Password:
```

- b. Find entitlement pools containing the channel:

```
# subscription-manager list --available | grep -A8 "Red Hat
Enterprise Linux Server"
```

- c. Use the pool identifiers located in the previous step to attach the **Red Hat Enterprise Linux Server** entitlement to the system:

```
# subscription-manager attach --pool=pool_id
```

- d. Enable the required channel:

```
# subscription-manager repos --enable=rhel-6-server-rpms
```

For RHEL OpenStack Platform 7, the required channels are **rhel-7-server-openstack-7.0-rpms** and **rhel-6-server-rh-common-rpms**.



Note

For more information, see "Subscribe to the Required Channels" in the *Installation Reference*.

7. Update the system.

```
# yum -y update
```

8. Install the **cloud-init** packages.

```
# yum install -y cloud-utils-growpart cloud-init
```

9. Edit the `/etc/cloud/cloud.cfg` configuration file and under `cloud_init_modules` add:

```
- resolv-conf
```

-

The **resolv-conf** option automatically configures the **resolv.conf** configuration file when an instance boots for the first time. This file contains information related to the instance such as **nameservers**, **domain**, and other options.

10. To prevent network issues, create **/etc/udev/rules.d/75-persistent-net-generator.rules** file.

```
# echo "#" > /etc/udev/rules.d/75-persistent-net-generator.rules
```

This prevents **/etc/udev/rules.d/70-persistent-net.rules** file from being created. If **/etc/udev/rules.d/70-persistent-net.rules** is created, networking may not function properly when booting from snapshots (the network interface is created as "eth1" rather than "eth0" and IP address is not assigned).

11. Add the following line to **/etc/sysconfig/network** to avoid problems accessing the EC2 metadata service.

```
NOZEROCONF=yes
```

12. To ensure the console messages appear in the **Log** tab on the dashboard and the **nova console-log** output, add the following boot option to the **/etc/grub.conf**:

```
console=tty0 console=ttyS0,115200n8
```

13. Un-register the virtual machine so that the resulting image does not contain the same subscription details for every instance cloned based on it.

```
# subscription-manager repos --disable=*  
# subscription-manager unregister  
# yum clean all
```

14. Power off the instance:

```
# poweroff
```

15. Reset and clean the image using the **virt-sysprep** command so it can be to create instances without issues:

```
[user@host]# virt-sysprep -d rhel6.6
```

16. Reduce image size using the **virt-sparsify** command. This command converts any free space within the disk image back to free space within the host:

```
[user@host]# virt-sparsify - -compress rhel6.qcow2 rhel6-  
cloud.qcow2
```

This creates a new **rhel6-cloud.qcow2** file in the location from where the command is run.



Note

You will need to manually resize the partitions of instances based on the image in accordance with the disk space in the flavor that is applied to the instance.

The `rhe16-cloud.qcow2` image file is ready to be uploaded to the Image service. For more information on uploading this image to your OpenStack deployment using the dashboard, see [Upload an Image](#)

1.2.2. Upload an Image

1. In the dashboard, select **Project > Compute > Images**.
2. Click **Create Image**.
3. Fill out the values, and click **Create Image** when finished.

Field	Notes
Name	Name for the image. The name must be unique within the project.
Description	Brief description to identify the image.
Image Source	Image source: Image Location or Image File . Based on your selection, the next field is displayed.
Image Location or Image File	<ul style="list-style-type: none"> ✦ Select Image Location option to specify the image location URL. ✦ Select Image File option to upload an image from the local disk.
Format	Image format (for example, qcow2).
Architecture	Image architecture. For example, use i686 for a 32-bit architecture or x86_64 for a 64-bit architecture.
Minimum Disk (GB)	Minimum disk size required to boot the image. If this field is not specified, the default value is 0 (no minimum).
Minimum RAM (MB)	Minimum memory size required to boot the image. If this field is not specified, the default value is 0 (no minimum).

Field	Notes
Public	If selected, makes the image public to all users with access to the project.
Protected	If selected, ensures only users with specific permissions can delete this image.



Note

You can also use the **glance image-create** command with the **property** option to create an image. More values are available on the command line. For a complete listing, see [Image Configuration Parameters](#).

1.2.3. Update an Image

1. In the dashboard, select **Project > Compute > Images**.
2. Click **Edit Image** from the dropdown list.



Note

The **Edit Image** option is available only when you log in as an **admin** user. When you log in as a **demo** user, you have the option to **Launch an instance** or **Create Volume**.

3. Update the fields and click **Update Image** when finished. You can update the following values - name, description, kernel ID, ramdisk ID, architecture, format, minimum disk, minimum RAM, public, protected.
4. Click the drop-down menu and select **Update Metadata** option.
5. Specify metadata by adding items from the left column to the right one. In the left column, there are metadata definitions from the Image Service Metadata Catalog. Select **Other** to add metadata with the key of your choice and click **Save** when finished.



Note

You can also use the **glance image-update** command with the **property** option to update an image. More values are available on the command line; for a complete listing, see [Image Configuration Parameters](#).

1.2.4. Delete an Image

1. In the dashboard, select **Project > Compute > Images**.

2. Select the image you want to delete and click **Delete Images**.

CHAPTER 2. CONFIGURE OPENSTACK COMPUTE STORAGE

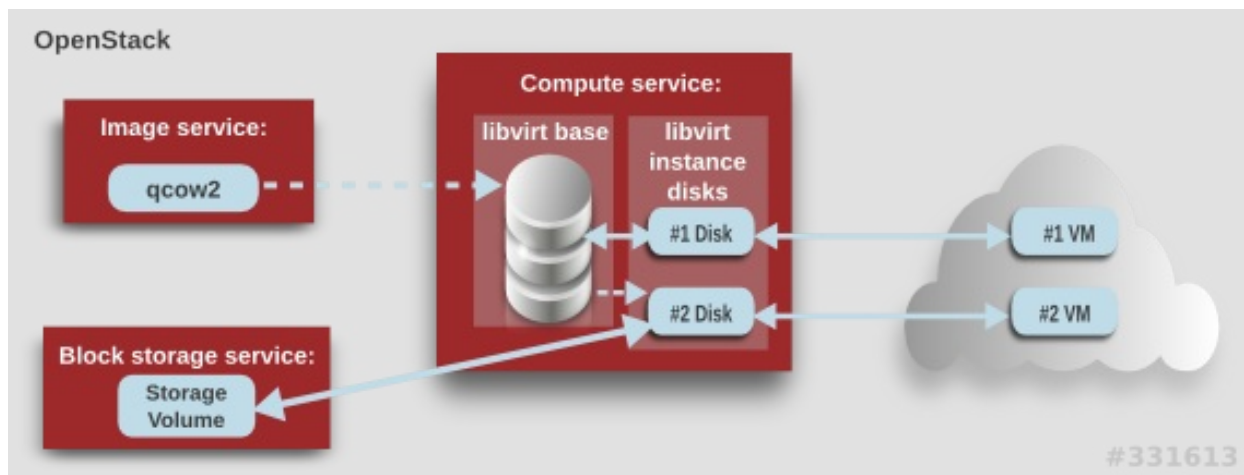
This chapter describes the architecture for the back-end storage of images in OpenStack Compute (nova), and provides basic configuration options.

2.1. ARCHITECTURE OVERVIEW

In Red Hat Enterprise Linux OpenStack Platform, the OpenStack Compute service uses the KVM hypervisor to execute compute workloads. The **libvirt** driver handles all interactions with KVM, and enables the creation of virtual machines.

Two types of **libvirt** storage must be considered for Compute:

- ✧ Base image, which is a cached and formatted copy of the Image service image.
- ✧ Instance disk, which is created using the **libvirt** base and is the back end for the virtual machine instance. Instance disk data can be stored either in Compute's ephemeral storage (using the **libvirt** base) or in persistent storage (for example, using Block Storage).



The steps that Compute takes to create a virtual machine instance are:

1. Cache the Image service's backing image as the **libvirt** base.
2. Convert the base image to the raw format (if configured).
3. Resize the base image to match the VM's flavor specifications.
4. Use the base image to create the libvirt instance disk.

In the diagram above, the #1 instance disk uses ephemeral storage; the #2 disk uses a block-storage volume.

Ephemeral storage is an empty, unformatted, additional disk available to an instance. This storage value is defined by the instance flavor. The value provided by the user must be less than or equal to the ephemeral value defined for the flavor. The default value is **0**, meaning no ephemeral storage is created.

The ephemeral disk appears in the same way as a plugged-in hard drive or thumb drive. It is available as a block device which you can check using the **lsblk** command. You can format it, mount it, and use it however you normally would a block device. There is no way to preserve or reference that disk beyond the instance it is attached to.

Block storage volume is persistent storage available to an instance regardless of the state of the running instance.

2.2. CONFIGURATION

Compute configuration for handling the **libvirt** base and instance disks can determine both performance and security aspects of your environment; parameters are configured in the `/etc/nova/nova.conf` file.

Table 2.1. Compute Image Parameters

Section	Parameter	Description	Default
[DEFAULT]	force_raw_images	<p>Whether to convert a non-raw cached base image to be raw (boolean). If a non-raw image is converted to raw, Compute:</p> <ul style="list-style-type: none"> ✦ Disallows backing files (which might be a security issue). ✦ Removes existing compression (to avoid CPU bottlenecks). <p>Converting the base to raw uses more space for any image that could have been used directly by the hypervisor (for example, a qcow2 image). If you have a system with slower I/O or less available space, you might want to specify 'false', trading the higher CPU requirements of compression for that of minimized input bandwidth.</p> <p>Raw base images are always used with libvirt_images_type=lvm.</p>	true
[DEFAULT]	use_cow_images	<p>Whether to use CoW (Copy on Write) images for libvirt instance disks (boolean):</p> <ul style="list-style-type: none"> ✦ false - The raw format is used. Without CoW, more space is used for common parts of the disk image ✦ true - The cqow2 format is used. With CoW, depending on the backing store and host caching, there may be better concurrency achieved by having each VM operate on its own copy. 	true

Section	Parameter	Description	Default
[DEFAULT]	preallocate_images	<p>Preallocation mode for libvirt instance disks. Value can be:</p> <ul style="list-style-type: none"> ✦ none - No storage is provisioned at instance start. ✦ space - Storage is fully allocated at instance start (using fallocate), which can help with both space guarantees and I/O performance. <p>Even when not using CoW instance disks, the copy each VM gets is sparse and so the VM may fail unexpectedly at run time with ENOSPC. By running fallocate(1) on the instance disk images, Compute immediately and efficiently allocates the space for them in the file system (if supported). Run time performance should also be improved because the file system does not have to dynamically allocate blocks at run time (reducing CPU overhead and more importantly file fragmentation).</p>	none
[DEFAULT]	resize_fs_using_block_device	<p>Whether to enable direct resizing of the base image by accessing the image over a block device (boolean). This is only necessary for images with older versions of cloud-init (that cannot resize themselves).</p> <p>Because this parameter enables the direct mounting of images which might otherwise be disabled for security reasons, it is not enabled by default.</p>	false
[DEFAULT]	default_ephemeral_format	<p>The default format that is used for a new ephemeral volume. Value can be: ext2, ext3, or ext4. The ext4 format provides much faster initialization times than ext3 for new, large disks. You can also override per instance using the guest_format configuration option.</p>	ext4

Section	Parameter	Description	Default
[DEFAULT]	image_cache_manager_interval	Number of seconds to wait between runs of the image cache manager, which impacts base caching on libvirt compute nodes. This period is used in the auto removal of unused cached images (see remove_unused_base_images and remove_unused_original_minimum_age_seconds).	2400
[DEFAULT]	remove_unused_base_images	Whether to enable the automatic removal of unused base images (checked every image_cache_manager_interval seconds). Images are defined as unused if they have not been accessed in remove_unused_original_minimum_age_seconds seconds.	true
[DEFAULT]	remove_unused_original_minimum_age_seconds	How old an unused base image must be before being removed from the libvirt cache (see remove_unused_base_images).	86400
[libvirt]	images_type	Image type to use for libvirt instance disks (deprecates use_cow_images). Value can be: raw , qcow2 , lvm , rbd , or default . If default is specified, the value used for the use_cow_images parameter is used.	default

CHAPTER 3. VIRTUAL MACHINE INSTANCES

OpenStack Compute is the central component that provides virtual machines on demand. Compute interacts with the Identity service for authentication, Image service for images (used to launch instances), and the dashboard service for the user and administrative interface.

RHEL OpenStack Platform allows you to easily manage virtual machine instances in the cloud. The Compute service creates, schedules, and manages instances, and exposes this functionality to other OpenStack components. This chapter discusses these procedures along with procedures to add components like key pairs, security groups, host aggregates and flavors. The term *instance* is used by OpenStack to mean a virtual machine instance.

3.1. MANAGE INSTANCES

Before you can create an instance, you need to ensure certain other OpenStack components (for example, a network, key pair and an image or a volume as the boot source) are available for the instance.

This section discusses the procedures to add these components, create and manage an instance. Managing an instance refers to updating, and logging in to an instance, viewing how the instances are being used, resizing or deleting them.

3.1.1. Add Components

Use the following sections to create a network, key pair and upload an image or volume source. These components are used in the creation of an instance and are not available by default. You will also need to create a new security group to allow SSH access to the user.

1. In the dashboard, select **Project**.
2. Select **Network > Networks**, and ensure there is a private network to which you can attach the new instance (to create a network, see **Add a Network** section in the **Networking Guide** available at [Red Hat Enterprise Linux OpenStack Platform](#)).
3. Select **Compute > Access & Security > Key Pairs**, and ensure there is a key pair (to create a key pair, see [Section 3.2.1.1, “Create a Key Pair”](#)).
4. Ensure that you have either an image or a volume that can be used as a boot source:
 - ✦ To view boot-source images, select the **Images** tab (to create an image, see [Section 1.2.1, “Create an Image”](#)).
 - ✦ To view boot-source volumes, select the **Volumes** tab (to create a volume, see [Section 4.1.1, “Create a Volume”](#)).
5. Select **Compute > Access & Security > Security Groups**, and ensure you have created a security group rule (to create a security group, see **Project Security Management** in the **Users and Identity Management Guide** available at [Red Hat Enterprise Linux OpenStack Platform](#)).

3.1.2. Create an Instance

1. In the dashboard, select **Project > Compute > Instances**.

2. Click **Launch Instance**.

3. Fill out instance fields (those marked with '*' are required), and click **Launch** when finished.

Tab	Field	Notes
Project and User	Project	Select the project from the dropdown list.
	User	Select the user from the dropdown list.
Details	Availability Zone	Zones are logical groupings of cloud resources in which your instance can be placed. If you are unsure, use the default zone (for more information, see Section 3.4, "Manage Host Aggregates").
	Instance Name	A name to identify your instance.
	Flavor	The flavor determines what resources the instance is given (for example, memory). For default flavor allocations and information on creating new flavors, see Section 3.3, "Manage Flavors" .
	Instance Count	The number of instances to create with these parameters. "1" is preselected.
	Instance Boot Source	Depending on the item selected, new fields are displayed allowing you to select the source: <ul style="list-style-type: none"> ✦ Image sources must be compatible with OpenStack (see Section 1.2, "Manage Images"). ✦ If a volume or volume source is selected, the source must be formatted using an image (see Section 4.1, "Basic Volume Usage and Configuration").
Access and Security	Key Pair	The specified key pair is injected into the instance and is used to remotely access the instance using SSH (if neither a direct login information or a static key pair is provided). Usually one key pair per project is created.

Tab	Field	Notes
	Security Groups	Security groups contain firewall rules which filter the type and direction of the instance's network traffic (for more information on configuring groups, see Project Security Management in the Users and Identity Management Guide available at Red Hat Enterprise Linux OpenStack Platform).
Networking	Selected Networks	You must select at least one network. Instances are typically assigned to a private network, and then later given a floating IP address to enable external access.
Post-Creation	Customization Script Source	<p>You can provide either a set of commands or a script file, which will run after the instance is booted (for example, to set the instance hostname or a user password). If 'Direct Input' is selected, write your commands in the Script Data field; otherwise, specify your script file.</p> <div style="display: flex; align-items: flex-start;"> <div style="width: 40px; height: 100px; background: repeating-linear-gradient(45deg, transparent, transparent 2px, #ccc 2px, #ccc 4px); margin-right: 10px;"></div> <div> <p>Note</p> <p>Any script that starts with '#cloud-config' is interpreted as using the cloud-config syntax (for information on the syntax, see http://cloudinit.readthedocs.org/en/latest/topics/examples.html).</p> </div> </div>
Advanced Options	Disk Partition	By default, the instance is built as a single partition and dynamically resized as needed. However, you can choose to manually configure the partitions yourself.
	Configuration Drive	If selected, OpenStack writes metadata to a read-only configuration drive that is attached to the instance when it boots (instead of to Compute's metadata service). After the instance has booted, you can mount this drive to view its contents (enables you to provide files to the instance).

3.1.3. Update an Instance (Actions menu)

You can update an instance by selecting **Project > Compute > Instances**, and selecting an action for that instance in the **Actions** column. Actions allow you to manipulate the instance in a number of ways:

Action	Description
Create Snapshot	Snapshots preserve the disk state of a running instance. You can create a snapshot to migrate the instance, as well as to preserve backup copies.
Associate/Disassociate Floating IP	You must associate an instance with a floating IP (external) address before it can communicate with external networks, or be reached by external users. Because there are a limited number of external addresses in your external subnets, it is recommended that you disassociate any unused addresses.
Edit Instance	Update the instance's name and associated security groups.
Edit Security Groups	Add and remove security groups to or from this instance using the list of available security groups (for more information on configuring groups, see Project Security Management in the Users and Identity Management Guide available at Red Hat Enterprise Linux OpenStack Platform).
Console	View the instance's console in the browser (allows easy access to the instance).
View Log	View the most recent section of the instance's console log. Once opened, you can view the full log by clicking View Full Log.
Pause/Resume Instance	Immediately pause the instance (you are not asked for confirmation); the state of the instance is stored in memory (RAM).
Suspend/Resume Instance	Immediately suspend the instance (you are not asked for confirmation); like hibernation, the state of the instance is kept on disk.
Resize Instance	Bring up the Resize Instance window (see Section 3.1.4, "Resize an Instance").

Action	Description
Soft Reboot	Gracefully stop and restart the instance. A soft reboot attempts to gracefully shut down all processes before restarting the instance.
Hard Reboot	Stop and restart the instance. A hard reboot effectively just shuts down the instance's <i>power</i> and then turns it back on.
Shut Off Instance	Gracefully stop the instance.
Rebuild Instance	Use new image and disk-partition options to rebuild the image (shut down, re-image, and re-boot the instance). If encountering operating system issues, this option is easier to try than terminating the instance and starting over.
Terminate Instance	Permanently destroy the instance (you are asked for confirmation).

You can create and allocate an external IP address, see [Section 3.2.3, “Create, Assign, and Release Floating IP Addresses”](#)

3.1.4. Resize an Instance

To resize an instance (memory or CPU count), you must select a new flavor for the instance that has the right capacity. If you are increasing the size, remember to first ensure that the host has enough space.

1. Ensure communication between hosts by setting up each host with SSH key authentication so that Compute can use SSH to move disks to other hosts (for example, compute nodes can share the same SSH key).

For more information about setting up SSH key authentication, see **Configure SSH Tunneling Between Nodes** in the **Migrating Instances Guide** available at [Red Hat Enterprise Linux OpenStack Platform](#).

2. Enable resizing on the original host by setting the following parameter in the `/etc/nova/nova.conf` file:

```
[DEFAULT] allow_resize_to_same_host = True
```

3. In the dashboard, select **Project > Compute > Instances**.
4. Click the instance's **Actions** arrow, and select **Resize Instance**.

5. Select a new flavor in the **New Flavor** field.
6. If you want to manually partition the instance when it launches (results in a faster build time):
 - a. Select **Advanced Options**.
 - b. In the **Disk Partition** field, select **Manual**.
7. Click **Resize**.

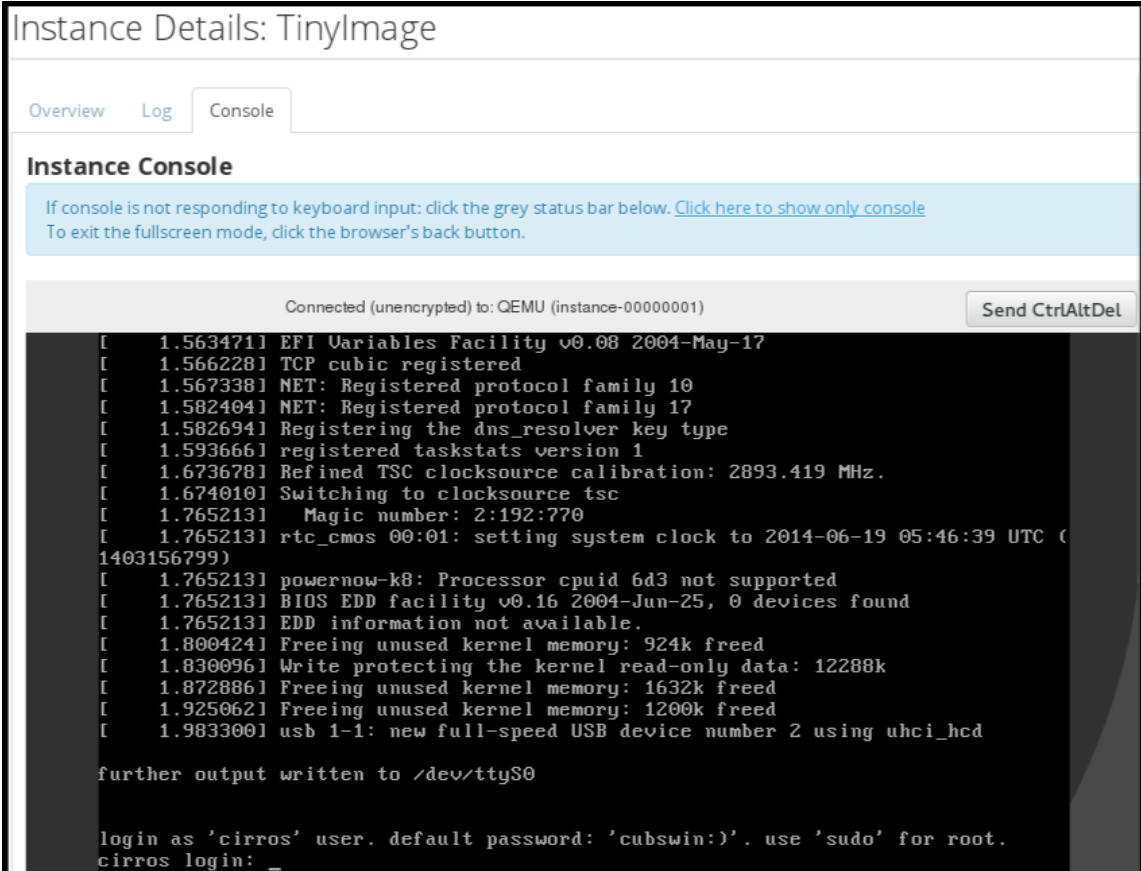
3.1.5. Connect to an Instance

This section discusses the different methods you can use to access an instance console using the dashboard or the command-line interface. You can also directly connect to an instance's serial port allowing you to debug even if the network connection fails.

3.1.5.1. Access an Instance Console using the Dashboard

The console allows you a way to directly access your instance within the dashboard.

1. In the dashboard, select **Compute > Instances**.
2. Click the instance's **More** button and select **Console**.



```

Instance Details: TinyImage
Overview Log Console
Instance Console
If console is not responding to keyboard input: click the grey status bar below. Click here to show only console
To exit the fullscreen mode, click the browser's back button.
Connected (unencrypted) to: QEMU (instance-00000001) Send CtrlAltDel
[ 1.563471] EFI Variables Facility v0.08 2004-May-17
[ 1.566228] TCP cubic registered
[ 1.567338] NET: Registered protocol family 10
[ 1.582404] NET: Registered protocol family 17
[ 1.582694] Registering the dns_resolver key type
[ 1.593666] registered taskstats version 1
[ 1.673678] Refined TSC clocksource calibration: 2893.419 MHz.
[ 1.674010] Switching to clocksource tsc
[ 1.765213] Magic number: 2:192:770
[ 1.765213] rtc_cmos 00:01: setting system clock to 2014-06-19 05:46:39 UTC (
1403156799)
[ 1.765213] powernow-k8: Processor cpuid 6d3 not supported
[ 1.765213] BIOS EDD facility v0.16 2004-Jun-25, 0 devices found
[ 1.765213] EDD information not available.
[ 1.800424] Freeing unused kernel memory: 924k freed
[ 1.830096] Write protecting the kernel read-only data: 12288k
[ 1.872886] Freeing unused kernel memory: 1632k freed
[ 1.925062] Freeing unused kernel memory: 1200k freed
[ 1.983300] usb 1-1: new full-speed USB device number 2 using uhci_hcd
further output written to /dev/ttyS0
login as 'cirros' user. default password: 'cubswin:'. use 'sudo' for root.
cirros login: _
  
```

3. Log in using the image's user name and password (for example, a CirrOS image uses *cirros/cubswin:)*).

3.1.5.2. Directly Connect to a VNC Console

You can directly access an instance's VNC console using a URL returned by `nova get-vnc-console` command.

Browser

To obtain a browser URL, use:

```
$ nova get-vnc-console INSTANCE_ID novnc
```

Java Client

To obtain a Java-client URL, use:

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

Note

`nova-xvpvncviewer` provides a simple example of a Java client. To download the client, use:

```
# git clone https://github.com/cloudbuilders/nova-xvpvncviewer
# cd nova-xvpvncviewer/viewer
# make
```

Run the viewer with the instance's Java-client URL:

```
# java -jar VncViewer.jar _URL_
```

This tool is provided only for customer convenience, and is not officially supported by Red Hat.

3.1.5.3. Directly Connect to a Serial Console

You can directly access an instance's serial port using a websocket client. Serial connections are typically used as a debugging tool (for example, instances can be accessed even if the network configuration fails). To obtain a serial URL for a running instance, use:

```
$ nova get-serial-console INSTANCE_ID
```

Note

novaconsole provides a simple example of a websocket client. To download the client, use:

```
# git clone https://github.com/larsks/novaconsole/
# cd novaconsole
```

Run the client with the instance's serial URL:

```
# python console-client-poll.py
```

This tool is provided only for customer convenience, and is not officially supported by Red Hat.

However, depending on your installation, the administrator may need to first set up the nova-serialproxy service. The proxy service is a websocket proxy that allows connections to OpenStack Compute serial ports.

3.1.5.3.1. Install and Configure nova-serialproxy

1. Install the **nova-serialproxy** service:

```
# yum install openstack-nova-serialproxy
```

2. Update the **serial_console** section in **/etc/nova/nova.conf**:

- a. Enable the **nova-serialproxy** service:

```
$ openstack-config --set /etc/nova/nova.conf serial_console
enabled true
```

- b. Specify the string used to generate URLs provided by the **nova get-serial-console** command.

```
$ openstack-config --set /etc/nova/nova.conf serial_console
base_url ws://PUBLIC_IP:6083/
```

Where **PUBLIC_IP** is the public IP address of the host running the **nova-serialproxy** service.

- c. Specify the IP address on which the instance serial console should listen (string).

```
$ openstack-config --set /etc/nova/nova.conf serial_console
listen 0.0.0.0
```

- d. Specify the address to which proxy clients should connect (string).

```
$ openstack-config --set /etc/nova/nova.conf serial_console
proxycient_address ws://HOST_IP:6083/
```

Where **HOST_IP** is the IP address of your Compute host. For example, an enabled **nova-serialproxy** service is as following:

```
[serial_console]
enabled=true
base_url=ws://192.0.2.0:6083/
listen=0.0.0.0
proxyclient_address=192.0.2.3
```

3. Restart Compute services:

```
# openstack-service restart nova
```

4. Start the **nova-serialproxy** service:

```
# systemctl enable openstack-nova-serialproxy
# systemctl start openstack-nova-serialproxy
```

5. Restart any running instances, to ensure that they are now listening on the right sockets.

6. Open the firewall for serial-console port connections. Serial ports are set using **[serial_console]** `port_range` in `/etc/nova/nova.conf`; by default, the range is 10000:20000. Update iptables with:

```
# iptables -I INPUT 1 -p tcp --dport 10000:20000 -j ACCEPT
```

3.1.6. View Instance Usage

The following usage statistics are available:

✎ Per Project

To view instance usage per project, select **Project > Compute > Overview**. A usage summary is immediately displayed for all project instances.

You can also view statistics for a specific period of time by specifying the date range and clicking **Submit**.

✎ Per Hypervisor

If logged in as an administrator, you can also view information for all projects. Click **Admin > System** and select one of the tabs. For example, the **Resource Usage** tab offers a way to view reports for a distinct time period. You might also click **Hypervisors** to view your current vCPU, memory, or disk statistics.



Note

The **vCPU Usage** value (**x of y**) reflects the number of total vCPUs of all virtual machines (x) and the total number of hypervisor cores (y).

3.1.7. Delete an Instance

1. In the dashboard, select **Project > Compute > Instances**, and select your instance.

2. Click **Terminate Instance**.



Note

Deleting an instance does not delete its attached volumes; you must do this separately (see [Section 4.1.4, “Delete a Volume”](#)).

3.1.8. Manage Multiple Instances at Once

If you need to start multiple instances at the same time (for example, those that were down for compute or controller maintenance) you can do so easily at **Project > Compute > Instances**:

1. Click the check boxes in the first column for the instances that you want to start. If you want to select all of the instances, click the check box in the first row in the table.
2. Click **More Actions** above the table and select **Start Instances**.

Similarly, you can shut off or soft reboot multiple instances by selecting the respective actions.

3.2. MANAGE INSTANCE SECURITY

You can manage access to an instance by assigning it the correct security group (set of firewall rules) and key pair (enables SSH user access). Further, you can assign a floating IP address to an instance to enable external network access. The sections below outline how to create and manage key pairs, security groups, floating IP addresses and logging in to an instance using SSH. There is also a procedure for injecting an **admin** password in to an instance.

For information on managing security groups, see **Project Security Management** in the **Users and Identity Management Guide** available at [Red Hat Enterprise Linux OpenStack Platform](#).

3.2.1. Manage Key Pairs

Key pairs provide SSH access to the instances. Each time a key pair is generated, its certificate is downloaded to the local machine and can be distributed to users. Typically, one key pair is created for each project (and used for multiple instances).

You can also import an existing key pair into OpenStack.

3.2.1.1. Create a Key Pair

1. In the dashboard, select **Project > Compute > Access & Security**.
2. On the **Key Pairs** tab, click **Create Key Pair**.
3. Specify a name in the **Key Pair Name** field, and click **Create Key Pair**.

When the key pair is created, a key pair file is automatically downloaded through the browser. Save this file for later connections from external machines. For command-line SSH connections, you can load this file into SSH by executing:

```
# ssh-add ~/.ssh/os-key.pem
```

3.2.1.2. Import a Key Pair

1. In the dashboard, select **Project > Compute > Access & Security**.
2. On the **Key Pairs** tab, click **Import Key Pair**.
3. Specify a name in the **Key Pair Name** field, and copy and paste the contents of your public key into the **Public Key** field.
4. Click **Import Key Pair**.

3.2.1.3. Delete a Key Pair

1. In the dashboard, select **Project > Compute > Access & Security**.
2. On the **Key Pairs** tab, click the key's **Delete Key Pair** button.

3.2.2. Create a Security Group

Security groups are sets of IP filter rules that can be assigned to project instances, and which define networking access to the instance. Security groups are project specific; project members can edit the default rules for their security group and add new rule sets.

1. In the dashboard, select the **Project** tab, and click **Compute > Access & Security**.
2. On the **Security Groups** tab, click **+ Create Security Group**.
3. Provide a name and description for the group, and click **Create Security Group**.

For more information on managing project security, see **Project Security Management** in the **Users and Identity Management Guide** available at [Red Hat Enterprise Linux OpenStack Platform](#).

3.2.3. Create, Assign, and Release Floating IP Addresses

By default, an instance is given an internal IP address when it is first created. However, you can enable access through the public network by creating and assigning a floating IP address (external address). You can change an instance's associated IP address regardless of the instance's state.

Projects have a limited range of floating IP addresses that can be used (by default, the limit is 50), so you should release these addresses for reuse when they are no longer needed. Floating IP addresses can only be allocated from an existing floating IP pool, see **Create Floating IP Pools** in the **Networking Guide** available at [Red Hat Enterprise Linux OpenStack Platform](#).

3.2.3.1. Allocate a Floating IP to the Project

1. In the dashboard, select **Project > Compute > Access & Security**.
2. On the **Floating IPs** tab, click **Allocate IP to Project**.
3. Select a network from which to allocate the IP address in the **Pool** field.
4. Click **Allocate IP**.

3.2.3.2. Assign a Floating IP

1. In the dashboard, select **Project > Compute > Access & Security**.
2. On the **Floating IPs** tab, click the address' **Associate** button.
3. Select the address to be assigned in the IP address field.



Note

If no addresses are available, you can click the + button to create a new address.

4. Select the instance to be associated in the **Port to be Associated** field. An instance can only be associated with one floating IP address.
5. Click **Associate**.

3.2.3.3. Release a Floating IP

1. In the dashboard, select **Project > Compute > Access & Security**.
2. On the **Floating IPs** tab, click the address' menu arrow (next to the **Associate/Disassociate** button).
3. Select **Release Floating IP**.

3.2.4. Log in to an Instance

Prerequisites:

- ✦ Ensure that the instance's security group has an SSH rule (see **Project Security Management** in the **Users and Identity Management Guide** available at [Red Hat Enterprise Linux OpenStack Platform](#)).
- ✦ Ensure the instance has a floating IP address (external address) assigned to it (see [Create, Assign, and Release Floating IP Addresses](#)).
- ✦ Obtain the instance's key-pair certificate. The certificate is downloaded when the key pair is created; if you did not create the key pair yourself, ask your administrator (see [Section 3.2.1, "Manage Key Pairs"](#)).

To first load the key pair file into SSH, and then use ssh without naming it:

1. Change the permissions of the generated key-pair certificate.

```
$ chmod 600 os-key.pem
```

2. Check whether **ssh-agent** is already running:

```
# ps -ef | grep ssh-agent
```

3. If not already running, start it up with:

■

```
# eval `ssh-agent`
```

4. On your local machine, load the key-pair certificate into SSH. For example:

```
$ ssh-add ~/.ssh/os-key.pem
```

5. You can now SSH into the file with the user supplied by the image.

The following example command shows how to SSH into the Red Hat Enterprise Linux guest image with the user **cloud-user**:

```
$ ssh cloud-user@192.0.2.24
```



Note

You can also use the certificate directly. For example:

```
$ ssh -i /myDir/os-key.pem cloud-user@192.0.2.24
```

3.2.5. Inject an admin Password Into an Instance

You can inject an **admin (root)** password into an instance using the following procedure.

1. In the `/etc/openstack-dashboard/local_settings` file, set the **change_set_password** parameter value to **True**.

```
can_set_password: True
```

2. In the `/etc/nova/nova.conf` file, set the **inject_password** parameter to **True**.

```
inject_password=true
```

3. Restart the Compute service.

```
# service nova-compute restart
```

When you use the **nova boot** command to launch a new instance, the output of the command displays an **adminPass** parameter. You can use this password to log into the instance as the **root** user.

The Compute service overwrites the password value in the `/etc/shadow` file for the **root** user. This procedure can also be used to activate the **root** account for the KVM guest images. For more information on how to use KVM guest images, see [Section 1.2.1.1, “Use a KVM Guest Image With RHEL OpenStack Platform”](#)

You can also set a custom password from the dashboard. To enable this, run the following command after you have set **can_set_password** parameter to **true**.

```
# systemctl restart httpd.service
```

The newly added **admin** password fields are as follows:

Launch Instance ✕

Project & User *
Details *
Access & Security
Networking *
Post-Creation

[Advanced Options](#)

Key Pair ⓘ

▼
+

Control access to your instance via key pairs, security groups, and other mechanisms.

Admin Password

👁

Confirm Admin Password

👁

Security Groups ⓘ

default

Cancel

Launch

These fields can be used when you launch or rebuild an instance.

3.3. MANAGE FLAVORS

Each created instance is given a flavor (resource template), which determines the instance's size and capacity. Flavors can also specify secondary ephemeral storage, swap disk, metadata to restrict usage, or special project access (none of the default flavors have these additional attributes defined).

Table 3.1. Default Flavors

Name	vCPUs	RAM	Root Disk Size
m1.tiny	1	512 MB	1 GB
m1.small	1	2048 MB	20 GB
m1.medium	2	4096 MB	40 GB
m1.large	4	8192 MB	80 GB

Name	vCPUs	RAM	Root Disk Size
m1.xlarge	8	16384 MB	160 GB

The majority of end users will be able to use the default flavors. However, you can create and manage specialized flavors. For example, you can:

- ✧ Change default memory and capacity to suit the underlying hardware needs.
- ✧ Add metadata to force a specific I/O rate for the instance or to match a host aggregate.



Note

Behavior set using image properties overrides behavior set using flavors (for more information, see [Section 1.2, “Manage Images”](#)).

3.3.1. Update Configuration Permissions

By default, only administrators can create flavors or view the complete flavor list (select Admin > System > Flavors). To allow all users to configure flavors, specify the following in the `/etc/nova/policy.json` file (nova-api server):

```
"compute_extension:flavormanage": "",
```

3.3.2. Create a Flavor

1. As an admin user in the dashboard, select **Admin > System > Flavors**.
2. Click **Create Flavor**, and specify the following fields:

Tab	Field	Description
Flavor Information	Name	Unique name.
	ID	Unique ID. The default value, auto , generates a UUID4 value, but you can also manually specify an integer or UUID4 value.
	VCPUs	Number of virtual CPUs.

Tab	Field	Description
	RAM (MB)	Memory (in megabytes).
	Root Disk (GB)	Ephemeral disk size (in gigabytes); to use the native image size, specify 0 . This disk is not used if Instance Boot Source=Boot from Volume .
	Ephemeral Disk (GB)	Secondary ephemeral disk size (in gigabytes) available to an instance. This disk is destroyed when an instance is deleted. The default value is 0 , which implies that no ephemeral disk is created.
	Swap Disk (MB)	Swap disk size (in megabytes).
Flavor Access	Selected Projects	Projects which can use the flavor. If no projects are selected, all projects have access (Public=Yes).

3. Click Create Flavor.

3.3.3. Update General Attributes

1. As an admin user in the dashboard, select **Admin > System > Flavors**.
2. Click the flavor's **Edit Flavor** button.
3. Update the values, and click **Save**.

3.3.4. Update Flavor Metadata

In addition to editing general attributes, you can add metadata to a flavor (**extra_specs**), which can help fine-tune instance usage. For example, you might want to set the maximum-allowed bandwidth or disk writes.

- ✳ Pre-defined keys determine hardware support or quotas. Pre-defined keys are limited by the hypervisor you are using (for libvirt, see [Table 3.2, “Libvirt Metadata”](#)).
- ✳ Both pre-defined and user-defined keys can determine instance scheduling. For example, you might specify **SpecialComp=True**; any instance with this flavor can then only run in a host aggregate with the same key-value combination in its metadata (see [Section 3.4, “Manage Host Aggregates”](#)).

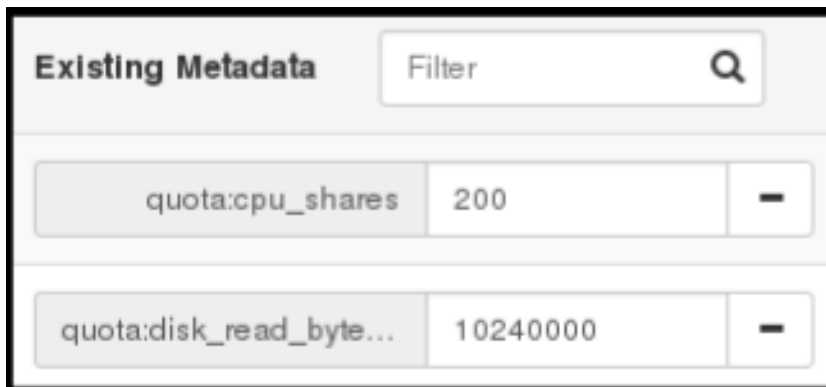
3.3.4.1. View Metadata

1. As an admin user in the dashboard, select **Admin > System > Flavors**.
2. Click the flavor’s **Metadata** link (**Yes** or **No**). All current values are listed on the right-hand side under **Existing Metadata**.

3.3.4.2. Add Metadata

You specify a flavor’s metadata using a **key/value** pair.

1. As an admin user in the dashboard, select **Admin > System > Flavors**.
2. Click the flavor’s **Metadata** link (**Yes** or **No**). All current values are listed on the right-hand side under **Existing Metadata**.
3. Under **Available Metadata**, click on the **Other** field, and specify the key you want to add (see [Table 3.2, “Libvirt Metadata”](#)).
4. Click the + button; you can now view the new key under **Existing Metadata**.
5. Fill in the key’s value in its right-hand field.




6. When finished with adding key-value pairs, click **Save**.

Table 3.2. Libvirt Metadata

Key	Description
-----	-------------

Key	Description
hw:action	<p>Action that configures support limits per instance. Valid actions are:</p> <ul style="list-style-type: none">✦ <code>cpu_max_sockets</code> - Maximum supported CPU sockets.✦ <code>cpu_max_cores</code> - Maximum supported CPU cores.✦ <code>cpu_max_threads</code> - Maximum supported CPU threads.✦ <code>cpu_sockets</code> - Preferred number of CPU sockets.✦ <code>cpu_cores</code> - Preferred number of CPU cores.✦ <code>cpu_threads</code> - Preferred number of CPU threads.✦ <code>serial_port_count</code> - Maximum serial ports per instance. <p>Example: hw:cpu_max_sockets=2</p>

Key	Description
hw:NUMA_def	<p>Definition of NUMA topology for the instance. For flavors whose RAM and vCPU allocations are larger than the size of NUMA nodes in the compute hosts, defining NUMA topology enables hosts to better utilize NUMA and improve performance of the guest OS. NUMA definitions defined through the flavor override image definitions. Valid definitions are:</p> <ul style="list-style-type: none"> ✦ numa_nodes - Number of NUMA nodes to expose to the instance. Specify '1' to ensure image NUMA settings are overridden. ✦ numa_mempolicy - Memory allocation policy. Valid policies are: <ul style="list-style-type: none"> ■ strict - Mandatory for the instance's RAM allocations to come from the NUMA nodes to which it is bound (default if numa_nodes is specified). ■ preferred - The kernel can fall back to using an alternative node. Useful when the numa_nodes is set to '1'. ✦ numa_cpus.0 - Mapping of vCPUs N-M to NUMA node 0 (comma-separated list). ✦ numa_cpus.1 - Mapping of vCPUs N-M to NUMA node 1 (comma-separated list). ✦ numa_mem.0 - Mapping N GB of RAM to NUMA node 0. ✦ numa_mem.1 - Mapping N GB of RAM to NUMA node 1. ✦ numa_cpu.N and numa_mem.N are only valid if numa_nodes is set. Additionally, they are only required if the instance's NUMA nodes have an asymmetrical allocation of CPUs and RAM (important for some NFV workloads).
	<div style="display: flex; align-items: flex-start;">  <div> <p>Note</p> <p>If the values of numa_cpu or numa_mem.N specify more than that available, an exception is raised.</p> </div> </div>
	<p>Example when the instance has 8 vCPUs and 4GB RAM:</p> <ul style="list-style-type: none"> ✦ hw:numa_nodes=2 ✦ hw:numa_cpus.0=0,1,2,3,4,5 ✦ hw:numa_cpus.1=6,7 ✦ hw:numa_mem.0=3 ✦ hw:numa_mem.1=1 <p>The scheduler looks for a host with 2 NUMA nodes with the ability to run 6 CPUs + 3 GB of RAM on one node, and 2 CPUS + 1 GB of RAM on another node. If a host has a single NUMA node with capability to run 8 CPUs and 4 GB of RAM, it will not be considered a valid match. The same logic is applied in the scheduler regardless of the numa_mempolicy setting.</p>

Key	Description
hw:watchdog_action	<p>An instance watchdog device can be used to trigger an action if the instance somehow fails (or hangs). Valid actions are:</p> <ul style="list-style-type: none"> ✦ disabled - The device is not attached (default value). ✦ pause - Pause the instance. ✦ poweroff - Forcefully shut down the instance. ✦ reset - Forcefully reset the instance. ✦ none - Enable the watchdog, but do nothing if the instance fails. <p>Example: hw:watchdog_action=poweroff</p>
hw_rng:action	<p>A random-number generator device can be added to an instance using its image properties (see hw_rng_model in the "Command-Line Interface Reference" in RHEL OpenStack Platform documentation).</p> <p>If the device has been added, valid actions are:</p> <ul style="list-style-type: none"> ✦ allowed - If True, the device is enabled; if False, disabled. By default, the device is disabled. ✦ rate_bytes - Maximum number of bytes the instance's kernel can read from the host to fill its entropy pool every rate_period (integer). ✦ rate_period - Duration of the read period in seconds (integer). <p>Example: hw_rng:allowed=True.</p>
hw_video:ram_max_mb	<p>Maximum permitted RAM to be allowed for video devices (in MB).</p> <p>Example: hw:ram_max_mb=64</p>

Key	Description
quota:option	<p data-bbox="544 235 1098 264">Enforcing limit for the instance. Valid options are:</p> <ul style="list-style-type: none"> <li data-bbox="544 293 1362 412">✦ <code>cpu_period</code> - Time period for enforcing <code>cpu_quota</code> (in microseconds). Within the specified <code>cpu_period</code>, each vCPU cannot consume more than <code>cpu_quota</code> of runtime. The value must be in range [1000, 1000000]; '0' means 'no value'. <li data-bbox="544 441 1398 584">✦ <code>cpu_quota</code> - Maximum allowed bandwidth (in microseconds) for the vCPU in each <code>cpu_period</code>. The value must be in range [1000, 18446744073709551]. '0' means 'no value'; a negative value means that the vCPU is not controlled. <code>cpu_quota</code> and <code>cpu_period</code> can be used to ensure that all vCPUs run at the same speed. <li data-bbox="544 613 1353 732">✦ <code>cpu_shares</code> - Share of CPU time for the domain. The value only has meaning when weighted against other machine values in the same domain. That is, an instance with a flavor with '200' will get twice as much machine time as an instance with '100'. <li data-bbox="544 761 1318 790">✦ <code>disk_read_bytes_sec</code> - Maximum disk reads in bytes per second. <li data-bbox="544 819 1310 848">✦ <code>disk_read_iops_sec</code> - Maximum read I/O operations per second. <li data-bbox="544 878 1326 907">✦ <code>disk_write_bytes_sec</code> - Maximum disk writes in bytes per second. <li data-bbox="544 936 1318 965">✦ <code>disk_write_iops_sec</code> - Maximum write I/O operations per second. <li data-bbox="544 994 1334 1032">✦ <code>disk_total_bytes_sec</code> - Maximum total throughput limit in bytes per second. <li data-bbox="544 1061 1305 1090">✦ <code>disk_total_iops_sec</code> - Maximum total I/O operations per second. <li data-bbox="544 1120 1246 1149">✦ <code>vif_inbound_average</code> - Desired average of incoming traffic. <li data-bbox="544 1178 1378 1216">✦ <code>vif_inbound_burst</code> - Maximum amount of traffic that can be received at <code>vif_inbound_peak</code> speed. <li data-bbox="544 1245 1334 1305">✦ <code>vif_inbound_peak</code> - Maximum rate at which incoming traffic can be received. <li data-bbox="544 1335 1257 1364">✦ <code>vif_outbound_average</code> - Desired average of outgoing traffic. <li data-bbox="544 1393 1347 1453">✦ <code>vif_outbound_burst</code> - Maximum amount of traffic that can be sent at <code>vif_outbound_peak</code> speed. <li data-bbox="544 1482 1347 1543">✦ <code>vif_outbound_peak</code> - Maximum rate at which outgoing traffic can be sent. <p data-bbox="544 1559 1182 1588">Example: quota:vif_inbound_average=10240</p>

3.4. MANAGE HOST AGGREGATES

A single Compute deployment can be partitioned into logical groups for performance or administrative purposes. OpenStack uses the following terms:

- ✦ *Host aggregates* - A host aggregate creates logical units in a OpenStack deployment by grouping together hosts. Aggregates are assigned Compute hosts and associated metadata; a host can be in more than one host aggregate. Only administrators can see or create host aggregates.

An aggregate's metadata is commonly used to provide information for use with the Compute

scheduler (for example, limiting specific flavors or images to a subset of hosts). Metadata specified in a host aggregate will limit the use of that host to any instance that has the same metadata specified in its flavor.

Administrators can use host aggregates to handle load balancing, enforce physical isolation (or redundancy), group servers with common attributes, or separate out classes of hardware. When you create an aggregate, a zone name must be specified, and it is this name which is presented to the end user.

- ✦ *Availability zones* - An availability zone is the end-user view of a host aggregate. An end user cannot view which hosts make up the zone, nor see the zone's metadata; the user can only see the zone's name.

End users can be directed to use specific zones which have been configured with certain capabilities or within certain areas.

3.4.1. Enable Host Aggregate Scheduling

By default, host-aggregate metadata is not used to filter instance usage; you must update the Compute scheduler's configuration to enable metadata usage:

1. Edit the `/etc/nova/nova.conf` file (you must have either root or nova user permissions).
2. Ensure that the `scheduler_default_filters` parameter contains:

- ✦ **AggregateInstanceExtraSpecsFilter** for host aggregate metadata. For example:

```
scheduler_default_filters=AggregateInstanceExtraSpecsFilter,RetryFilter,RamFilter,ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,CoreFilter
```

- ✦ **AvailabilityZoneFilter** for availability zone host specification when launching an instance. For example:

```
scheduler_default_filters=AvailabilityZoneFilter,RetryFilter,RamFilter,ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,CoreFilter
```

3. Save the configuration file.

3.4.2. View Availability Zones or Host Aggregates

As an admin user in the dashboard, select **Admin > System > Host Aggregates**. All currently defined aggregates are listed in the **Host Aggregates** section; all zones are in the **Availability Zones** section.

3.4.3. Add a Host Aggregate

1. As an admin user in the dashboard, select **Admin > System > Host Aggregates**. All currently defined aggregates are listed in the **Host Aggregates** section.
2. Click **Create Host Aggregate**.

3. Add a name for the aggregate in the **Name** field, and a name by which the end user should see it in the **Availability Zone** field.
4. Click **Manage Hosts within Aggregate**.
5. Select a host for use by clicking its + icon.
6. Click **Create Host Aggregate**.

3.4.4. Update a Host Aggregate

1. As an admin user in the dashboard, select **Admin > System > Host Aggregates**. All currently defined aggregates are listed in the **Host Aggregates** section.
2. To update the instance's **Name** or **Availability zone**:
 - ✦ Click the aggregate's **Edit Host Aggregate** button.
 - ✦ Update the **Name** or **Availability Zone** field, and click **Save**.
3. To update the instance's **Assigned hosts**:
 - ✦ Click the aggregate's arrow icon under **Actions**.
 - ✦ Click **Manage Hosts**.
 - ✦ Change a host's assignment by clicking its + or - icon.
 - ✦ When finished, click **Save**.
4. To update the instance's **Metadata**:
 - ✦ Click the aggregate's arrow icon under **Actions**.
 - ✦ Click the **Update Metadata** button. All current values are listed on the right-hand side under **Existing Metadata**.
 - ✦ Under **Available Metadata**, click on the **Other** field, and specify the key you want to add. Use predefined keys (see [Table 3.3, "Host Aggregate Metadata"](#)) or add your own (which will only be valid if exactly the same key is set in an instance's flavor).
 - ✦ Click the + button; you can now view the new key under **Existing Metadata**.



Note

Remove a key by clicking its - icon.

- ✦ Click **Save**.

Table 3.3. Host Aggregate Metadata

Key	Description
cpu_allocation_ratio	Sets allocation ratio of virtual CPU to physical CPU. Depends on the AggregateCoreFilter filter being set for the Compute scheduler.
disk_allocation_ratio	Sets allocation ratio of Virtual disk to physical disk. Depends on the AggregateDiskFilter filter being set for the Compute scheduler.
filter_tenant_id	If specified, the aggregate only hosts this tenant (project). Depends on the AggregateMultiTenancyIsolation filter being set for the Compute scheduler.
ram_allocation_ratio	Sets allocation ratio of virtual RAM to physical RAM. Depends on the AggregateRamFilter filter being set for the Compute scheduler.

3.4.5. Delete a Host Aggregate

1. As an admin user in the dashboard, select **Admin > System > Host Aggregates**. All currently defined aggregates are listed in the **Host Aggregates** section.
2. Remove all assigned hosts from the aggregate:
 - a. Click the aggregate's arrow icon under **Actions**.
 - b. Click **Manage Hosts**.
 - c. Remove all hosts by clicking their - icon.
 - d. When finished, click **Save**.
3. Click the aggregate's arrow icon under **Actions**.
4. Click **Delete Host Aggregate** in this and the next dialog screen.

3.5. SCHEDULE HOSTS AND CELLS

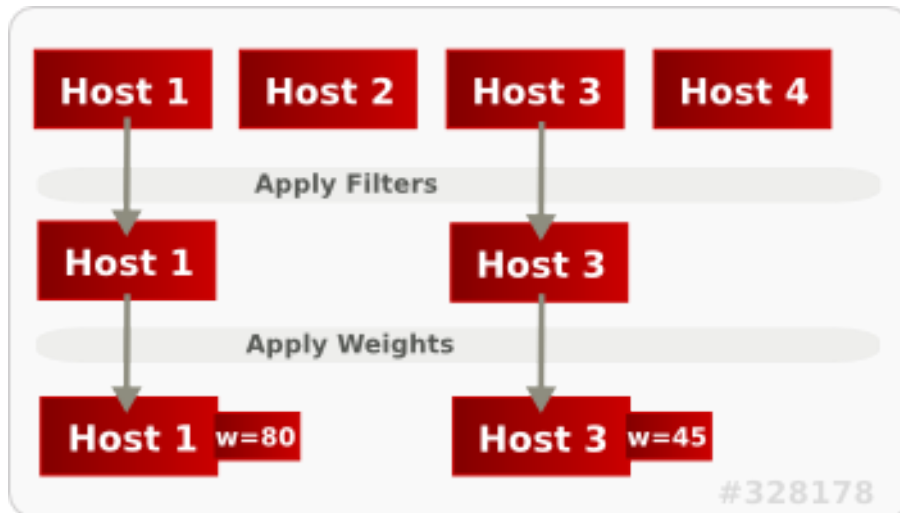
The Compute scheduling service determines on which cell or host (or host aggregate), an instance will be placed. As an administrator, you can influence where the scheduler will place an instance. For example, you might want to limit scheduling to hosts in a certain group or with the right RAM.

You can configure the following components:

- ✎ Filters - Determine the initial set of hosts on which an instance might be placed (see [Section 3.5.1, "Configure Scheduling Filters"](#)).

- ✦ Weights - When filtering is complete, the resulting set of hosts are prioritized using the weighting system. The highest weight has the highest priority (see [Section 3.5.2, "Configure Scheduling Weights"](#)).
- ✦ Scheduler service - There are a number of configuration options in the `/etc/nova/nova.conf` file (on the scheduler host), which determine how the scheduler executes its tasks, and handles weights and filters. There is both a host and a cell scheduler. For a list of these options, refer to the "Configuration Reference" ([RHEL OpenStack Platform Documentation](#)).

In the following diagram, both host 1 and 3 are eligible after filtering. Host 1 has the highest weight and therefore has the highest priority for scheduling.



3.5.1. Configure Scheduling Filters

You define which filters you would like the scheduler to use in the `scheduler_default_filters` option (`/etc/nova/nova.conf` file; you must have either root or nova user permissions). Filters can be added or removed.

By default, the following filters are configured to run in the scheduler:


```
scheduler_default_filters=RetryFilter,AvailabilityZoneFilter,RamFilter,
ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,ServerGroupAntiAffinityFilter,ServerGroupAffinityFilter
```

Some filters use information in parameters passed to the instance in:

- ✦ The `nova boot` command, see the "Command-Line Interface Reference" in [RHEL OpenStack Platform Documentation](#).
- ✦ The instance's flavor (see [Section 3.3.4, "Update Flavor Metadata"](#))
- ✦ The instance's image (see [Appendix A, Image Configuration Parameters](#)).

All available filters are listed in the following table.

Table 3.4. Scheduling Filters

Filter	Description
AggregateCoreFilter	<p>Uses the host-aggregate metadata key <code>cpu_allocation_ratio</code> to filter out hosts exceeding the over-commit ratio (virtual CPU to physical CPU allocation ratio); only valid if a host aggregate is specified for the instance.</p> <p>If this ratio is not set, the filter uses the <code>cpu_allocation_ratio</code> value in the <code>/etc/nova/nova.conf</code> file. The default value is 16.0 (16 virtual CPU can be allocated per physical CPU).</p>
AggregateDiskFilter	<p>Uses the host-aggregate metadata key <code>disk_allocation_ratio</code> to filter out hosts exceeding the over-commit ratio (virtual disk to physical disk allocation ratio); only valid if a host aggregate is specified for the instance.</p> <p>If this ratio is not set, the filter uses the <code>disk_allocation_ratio</code> value in the <code>/etc/nova/nova.conf</code> file. The default value is 1.0 (one virtual disk can be allocated for each physical disk).</p>
AggregateImageProperties Isolation	<p>Only passes hosts in host aggregates whose metadata matches the instance's image metadata; only valid if a host aggregate is specified for the instance. For more information, see Section 1.2.1, "Create an Image".</p>
AggregateInstanceExtraSpecsFilter	<p>Metadata in the host aggregate must match the host's flavor metadata. For more information, see Section 3.3.4, "Update Flavor Metadata".</p>
AggregateMultiTenancyIsolation	<p>A host with the specified <code>filter_tenant_id</code> can only contain instances from that tenant (project).</p> <div style="display: flex; align-items: flex-start; margin-top: 10px;"> <div style="flex: 1;">  </div> <div style="flex: 2;"> <p>Note</p> <p>The tenant can still place instances on other hosts.</p> </div> </div>
AggregateRamFilter	<p>Uses the host-aggregate metadata key <code>ram_allocation_ratio</code> to filter out hosts exceeding the over commit ratio (virtual RAM to physical RAM allocation ratio); only valid if a host aggregate is specified for the instance.</p> <p>If this ratio is not set, the filter uses the <code>ram_allocation_ratio</code> value in the <code>/etc/nova/nova.conf</code> file. The default value is 1.5 (1.5 RAM can be allocated for each physical RAM).</p>

Filter	Description
AllHostsFilter	Passes all available hosts (however, does not disable other filters).
AvailabilityZoneFilter	Filters using the instance's specified availability zone.
ComputeCapabilitiesFilter	Ensures Compute metadata is read correctly. Anything before the <code>:</code> is read as a namespace. For example, quota:cpu_period uses quota as the namespace and cpu_period as the key.
ComputeFilter	Passes only hosts that are operational and enabled.
CoreFilter	Uses the <code>cpu_allocation_ratio</code> in the <code>/etc/nova/nova.conf</code> file to filter out hosts exceeding the over commit ratio(virtual CPU to physical CPU allocation ratio). The default value is 16.0 (16 virtual CPU can be allocated per physical CPU).
DifferentHostFilter	Enables an instance to build on a host that is different from one or more specified hosts. Specify different hosts using the nova boot option --different_host option.
DiskFilter	Uses <code>disk_allocation_ratio</code> in the <code>/etc/nova/nova.conf</code> file to filter out hosts exceeding the over commit ratio(virtual disk to physical disk allocation ratio). The default value is 1.0 (one virtual disk can be allocated for each physical disk).
ImagePropertiesFilter	Only passes hosts that match the instance's image properties. For more information, see Section 1.2.1, "Create an Image" .
IsolatedHostsFilter	Passes only isolated hosts running isolated images that are specified in the <code>/etc/nova/nova.conf</code> file using isolated_hosts and isolated_images (comma-separated values).
JsonFilter	Recognises and uses an instance's custom JSON filters: <ul style="list-style-type: none"> ✦ Valid operators are: <code>=</code>, <code><</code>, <code>></code>, <code>in</code>, <code>∈</code>, <code>>=</code>, <code>not</code>, <code>or</code>, and ✦ Recognised variables are: <code>\$free_ram_mb</code>, <code>\$free_disk_mb</code>, <code>\$total_usable_ram_mb</code>, <code>\$vcpus_total</code>, <code>\$vcpus_used</code>

Filter	Description
	<p>The filter is specified as a query hint in the nova boot command. For example:</p> <pre>--hint query='[>=', '\$free_disk_mb', 200 * 1024]'</pre>
MetricFilter	Filters out hosts with unavailable metrics.
NUMATopologyFilter	Filters out hosts based on its NUMA topology; if the instance has no topology defined, any host can be used. The filter tries to match the exact NUMA topology of the instance to those of the host (it does not attempt to pack the instance onto the host). The filter also looks at the standard over-subscription limits for each NUMA node, and provides limits to the compute host accordingly.
RamFilter	Uses <code>ram_allocation_ratio</code> in the <code>/etc/nova/nova.conf</code> file to filter out hosts exceeding the over commit ratio (virtual RAM to physical RAM allocation ratio). The default value is 1.5 (1.5 RAM can be allocated for each physical RAM).
RetryFilter	Filters out hosts that have failed a scheduling attempt; valid if <code>scheduler_max_attempts</code> is greater than zero (by default, scheduler_max_attempts=3).
SameHostFilter	Passes one or more specified hosts; specify hosts for the instance using the --hint same_host option for nova boot .
ServerGroupAffinityFilter	<p>Only passes hosts for a specific server group:</p> <ul style="list-style-type: none"> ✦ Give the server group the affinity policy (<code>nova server-group-create --policy affinity groupName</code>). ✦ Build the instance with that group (nova boot option --hint group=UUID)
ServerGroupAntiAffinityFilter	<p>Only passes hosts in a server group that do not already host an instance:</p> <ul style="list-style-type: none"> ✦ Give the server group the anti-affinity policy (nova server-group-create --policy anti-affinity groupName). ✦ Build the instance with that group (nova boot option --hint group=UUID).

Filter	Description
SimpleCIDRAffinityFilter	<p>Only passes hosts on the specified IP subnet range specified by the instance's cidr and build_new_host_ip hints. Example:</p> <pre>--hint build_near_host_ip=192.0.2.0 --hint cidr=/24</pre>

3.5.2. Configure Scheduling Weights

Both cells and hosts can be weighted for scheduling; the host or cell with the largest weight (after filtering) is selected. All weighers are given a multiplier that is applied after normalising the node's weight. A node's weight is calculated as:

$$w1_multiplier * norm(w1) + w2_multiplier * norm(w2) + \dots$$

You can configure weight options in the scheduler host's `/etc/nova/nova.conf` file (must have either root or nova user permissions).

3.5.2.1. Configure Weight Options for Hosts

You can define the host weighers you would like the scheduler to use in the `[DEFAULT] scheduler_weight_classes` option. Valid weighers are:

- ✦ `nova.scheduler.weights.ram` - Weighs the host's available RAM.
- ✦ `nova.scheduler.weights.metrics` - Weighs the host's metrics.
- ✦ `nova.scheduler.weights.all_weighers` - Uses all host weighers (default).

Table 3.5. Host Weight Options

Weigher	Option	Description
All	<code>[DEFAULT] scheduler_host_subset_size</code>	Defines the subset size from which a host is selected (integer); must be at least 1 . A value of 1 selects the first host returned by the weighing functions. Any value less than 1 is ignored and 1 is used instead (integer value).

Weigher	Option	Description
metrics	[metrics] required	Specifies how to handle metrics in [metrics] weight_setting that are unavailable: <ul style="list-style-type: none"> ✦ True- Metrics are required; if unavailable, an exception is raised. To avoid the exception, use the MetricFilter filter in the scheduler_default_filters option. ✦ False - The unavailable metric is treated as a negative factor in the weighing process; the returned value is set by weight_of_unavailable.
metrics	[metrics] weight_of_unavailable	Used as the weight if any metric in [metrics] weight_setting is unavailable; valid if required=False .
metrics	[metrics] weight_multiplier	Multplier used for weighing metrics. By default, weight_multiplier=1.0 and spreads instances across possible hosts. If this value is negative, the host with lower metrics is prioritized, and instances are stacked in hosts.
metrics	[metrics] weight_setting	Specifies metrics and the ratio with which they are weighed; use a comma-separated list of metric=ratio pairs. Valid metric names are: <ul style="list-style-type: none"> ✦ cpu.frequency - Current CPU frequency ✦ cpu.user.time - CPU user mode time ✦ cpu.kernel.time - CPU kernel time ✦ cpu.idle.time - CPU idle time ✦ cpu.iowait.time - CPU I/O wait time ✦ cpu.user.percent - CPU user mode percentage ✦ cpu.kernel.percent - CPU kernel percentage ✦ cpu.idle.percent - CPU idle percentage ✦ cpu.iowait.percent - CPU I/O wait percentage ✦ cpu.percent - Generic CPU utilization <p>Example: weight_setting=cpu.user.time=1.0</p>
ram	[DEFAULT] ram_weight_multiplier	Multplier for RAM (floating point). By default, ram_weight_multiplier=1.0 and spreads instances across possible hosts. If this value is negative, the host with less RAM is prioritized, and instances are stacked in hosts.

3.5.2.2. Configure Weight Options for Cells

You define which cell weighers you would like the scheduler to use in the `[cells]` `scheduler_weight_classes` option (`/etc/nova/nova.conf` file; you must have either **root** or **nova** user permissions).



Note

The use of cells is available in this release as a *Technology Preview*, and therefore is not fully supported by Red Hat. It should only be used for testing, and should not be deployed in a production environment. For more information about Technology Preview features, see [Scope of Coverage Details](#).

Valid weighers are:

- ✦ `nova.cells.weights.all_weighers` - Uses all cell weighers(default).
- ✦ `nova.cells.weights.mute_child` - Weighs whether a child cell has not sent capacity or capability updates for some time.
- ✦ `nova.cells.weights.ram_by_instance_type` - Weighs the cell's available RAM.
- ✦ `nova.cells.weights.weight_offset` - Evaluates a cell's weight offset.



Note

A cell's weight offset is specified using `--woffset` **in the** `nova-manage cell create` command.

Table 3.6. Cell Weight Options

Weighers	Option	Description
<code>mute_child</code>	<code>[cells]</code> <code>mute_weight_multiplier</code>	Multiplier for hosts which have been silent for some time (negative floating point). By default, this value is -10.0 .
<code>mute_child</code>	<code>[cells]</code> <code>mute_weight_value</code>	Weight value given to silent hosts (positive floating point). By default, this value is 1000.0 .

Weighers	Option	Description
<code>ram_by_instance_type</code>	[cells] <code>ram_weight_multiplier</code>	Multiplier for weighing RAM (floating point). By default, this value is 1.0 , and spreads instances across possible cells. If this value is negative, the cell with fewer RAM is prioritized, and instances are stacked in cells.
<code>weight_offset</code>	[cells] <code>offset_weight_multiplier</code>	Multiplier for weighing cells (floating point). Enables the instance to specify a preferred cell (floating point) by setting its weight offset to 9999999999999999 (highest weight is prioritized). By default, this value is 1.0 .

3.6. EVACUATE INSTANCES

If you want to move an instance from a dead or shut-down compute node to a new host server in the same environment (for example, because the server needs to be swapped out), you can evacuate it using **nova evacuate**.

- ✦ An evacuation is only useful if the instance disks are on shared storage or if the instance disks are Block Storage volumes. Otherwise, the disks will not be accessible and cannot be accessed by the new compute node.
- ✦ An instance can only be evacuated from a server if the server is shut down; if the server is not shut down, the **evacuate** command will fail.

Note

If you have a functioning compute node, and you want to:

- ✦ Make a static copy (not running) of an instance for backup purposes or to copy the instance to a different environment, make a snapshot using **nova image-create** (see [Migrate a Static Instance](#)).
- ✦ Move an instance in a static state (not running) to a host in the same environment (shared storage not needed), migrate it using **nova migrate** (see [Migrate a Static Instance](#)).
- ✦ Move an instance in a live state (running) to a host in the same environment, migrate it using **nova live-migration** (see [Migrate a Live \(running\) Instance](#)).

3.6.1. Evacuate One Instance

1. Evacuate an instance using:

```
# nova evacuate [--password pass] [--on-shared-storage]
instance_name [target_host]
```

Where:

- ✦ **--password** - Admin password to set for the evacuated instance (cannot be used if **--on-shared-storage** is specified). If a password is not specified, a random password is generated and output when evacuation is complete.
- ✦ **--on-shared-storage** - Indicates that all instance files are on shared storage.
- ✦ **instance_name** - Name of the instance to be evacuated.
- ✦ **target_host** - Host to which the instance is evacuated; if you do not specify the host, the Compute scheduler selects one for you. You can find possible hosts using:

```
# nova host-list | grep compute
```

For example:

```
# nova evacuate myDemoInstance Compute2_OnEL7.myDomain
```

3.6.2. Evacuate All Instances

1. Evacuate all instances on a specified host using:

```
# nova host-evacuate instance_name [--target target_host] [--on-
shared-storage] source_host
```

Where:

- ✦ **--target** - Host to which the instance is evacuated; if you do not specify the host, the Compute scheduler selects one for you. You can find possible hosts using:

```
# nova host-list | grep compute
```

- ✦ **--on-shared-storage** - Indicates that all instance files are on shared storage.
- ✦ **source_host** - Name of the host to be evacuated.

For example:

```
# nova host-evacuate --target Compute2_OnEL7.localdomain
myDemoHost.localdomain
```

3.6.3. Configure Shared Storage

If you are using shared storage, this procedure exports the instances directory for the Compute

service to the two nodes, and ensures the nodes have access. The directory path is set in the **state_path** and **instances_path** parameters in the `/etc/nova/nova.conf` file. This procedure uses the default value, which is `/var/lib/nova/instances`. Only users with root access can set up shared storage.

1. On the controller host:

- a. Ensure the `/var/lib/nova/instances` directory has read-write access by the Compute service user (this user must be the same across controller and nodes). For example:

```
drwxr-xr-x. 9 nova nova 4096 Nov  5 20:37 instances
```

- b. Add the following lines to the `/etc/exports` file; switch out `node1_IP` and `node2_IP` for the IP addresses of the two compute nodes:

```
/var/lib/nova/instances (rw, sync, fsid=0, no_root_squash)
/var/lib/nova/instances (rw, sync, fsid=0, no_root_squash)
```

- c. Export the `/var/lib/nova/instances` directory to the compute nodes.

```
# exportfs -avr
```

- d. Restart the NFS server:

```
# systemctl restart nfs-server
```

2. On each compute node:

- a. Ensure the `/var/lib/nova/instances` directory exists locally.
- b. Add the following line to the `/etc/fstab` file:

```
:/var/lib/nova/instances /var/lib/nova/instances nfs4
defaults 0 0
```

- c. Mount the controller's instance directory (all devices listed in `/etc/fstab`):

```
# mount -a -v
```

- d. Ensure `qemu` can access the directory's images:

```
# ls -ld /var/lib/nova/instances
drwxr-xr-x. 9 nova nova 4096 Nov  5 20:37
/var/lib/nova/instances
```

- e. Ensure that the node can see the instances directory with:

```
drwxr-xr-x. 9 nova nova 4096 Nov  5 20:37
/var/lib/nova/instances
```

**Note**

You can also run the following to view all mounted devices:

```
# df -k
```

3.7. MANAGE INSTANCE SNAPSHOTS

An instance snapshot allows you to create a new image from an instance. This is very convenient for upgrading base images or for taking a published image and customizing it for local use.

The difference between an image that you upload directly to the Image Service and an image that you create by snapshot is that an image created by snapshot has additional properties in the Image Service database. These properties are found in the **image_properties** table and include the following parameters:

Name	Value
image_type	snapshot
instance_uuid	<uuid of instance that was snapshotted>
base_image_ref	<uuid of original image of instance that was snapshotted>
image_location	snapshot

Snapshots allow you to create new instances based on that snapshot, and potentially restore an instance to that state. Moreover, this can be performed while the instance is running.

By default, a snapshot is accessible to the users and projects that were selected while launching an instance that the snapshot is based on.

3.7.1. Create an Instance Snapshot

1. In the dashboard, select **Project > Compute > Instances**.
2. Select the instance from which you want to create a snapshot.
3. In the **Actions** column, click **Create Snapshot**.
4. In the **Create Snapshot** dialog, enter a name for the snapshot and click **Create Snapshot**.

The **Images** category now shows the instance snapshot.

To launch an instance from a snapshot, select the snapshot and click **Launch**.

3.7.2. Manage a Snapshot

1. In the dashboard, select **Project > Images**.
2. All snapshots you created, appear under the **Project** option.
3. For every snapshot you create, you can perform the following functions, using the dropdown list:
 - a. Use the **Create Volume** option to create a volume and entering the values for volume name, description, image source, volume type, size and availability zone. For more information, see [Create a Volume](#).
 - b. Use the **Edit Image** option to update the snapshot image by updating the values for name, description, Kernel ID, Ramdisk ID, Architecture, Format, Minimum Disk (GB), Minimum RAM (MB), public or private. For more information, see [Update an Image](#).
 - c. Use the **Delete Image** option to delete the snapshot.

3.7.3. Rebuild an Instance to a State in a Snapshot

In an event that you delete an instance on which a snapshot is based, the snapshot still stores the instance ID. You can check this information using the `nova image-list` command and use the snapshot to restore the instance.

1. In the dashboard, select **Project > Compute > Images**.
2. Select the snapshot from which you want to restore the instance.
3. In the **Actions** column, click **Launch Instance**.
4. In the **Launch Instance** dialog, enter a name and the other details for the instance and click **Launch**.

For more information on launching an instance, see [Create an Instance](#).

3.7.4. Consistent Snapshots

Previously, file systems had to be quiesced manually (`fsfreeze`) before taking a snapshot of active instances for consistent backups.

With the RHEL OpenStack Platform 7 release, Compute's `libvirt` driver now automatically requests the *QEMU Guest Agent* to freeze the file systems (and applications if `fsfreeze-hook` is installed) during an image snapshot. Support for quiescing file systems enables scheduled, automatic snapshots at the block device level.

This feature is only valid if the QEMU Guest Agent is installed (`qemu-ga`) and the image metadata enables the agent (`hw_qemu_guest_agent=yes`)



Note

Snapshots should not be considered a substitute for an actual system backup.

CHAPTER 4. MANAGE VOLUMES

A volume is a block storage device that provides persistent storage to OpenStack instances.

4.1. BASIC VOLUME USAGE AND CONFIGURATION

The following procedures describe how to perform basic end-user volume management. These procedures do not require administrative privileges.

4.1.1. Create a Volume

1. In the dashboard, select **Project > Compute > Volumes**.
2. Click **Create Volume**, and edit the following fields:

Field	Description
Volume name	Name of the volume.
Description	Optional, short description of the volume.
Type	Optional volume type (see Section 4.2.3, “Group Volume Settings with Volume Types”). If you have multiple Block Storage back ends, you can use this to select a specific back end. See Section 4.1.2, “Specify Back End for Volume Creation” for details.
Size (GB)	Volume size (in gigabytes).
Availability Zone	Availability zones (logical server groups), along with host aggregates, are a common method for segregating resources within OpenStack. Availability zones are defined during installation. For more information on availability zones and host aggregates, see Section 3.4, “Manage Host Aggregates” .

3. Specify a **Volume Source**:

Source	Description
--------	-------------

Source	Description
No source, empty volume	The volume will be empty, and will not contain a file system or partition table.
Snapshot	Use an existing snapshot as a volume source. If you select this option, a new Use snapshot as a source list appears; you can then choose a snapshot from the list. For more information about volume snapshots, refer to Section 4.1.8, “Create, Clone, or Delete Volume Snapshots” .
Image	Use an existing image as a volume source. If you select this option, a new Use image as a source lists appears; you can then choose an image from the list.
Volume	Use an existing volume as a volume source. If you select this option, a new Use volume as a source list appears; you can then choose a volume from the list.

4. Click **Create Volume**. After the volume is created, its name appears in the **Volumes** table.

4.1.2. Specify Back End for Volume Creation

You can configure the Block Storage service to use multiple back ends. For example, [Configure OpenStack to Use an NFS Back End](#) provides step-by-step instructions on how to configure the Block Storage service to use an NFS share alongside the default back end.

Whenever multiple Block Storage back ends are configured, you will also need to create a volume type for each back end. You can then use the type to specify which back end should be used for a created volume. For more information about volume types, see [Section 4.2.3, “Group Volume Settings with Volume Types”](#).

To specify a back end when creating a volume, select its corresponding volume type from the Type drop-down list (see [Section 4.1.1, “Create a Volume”](#)).

If you do not specify a back end during volume creation, the Block Storage service will automatically choose one for you. By default, the service will choose the back end with the most available free space. You can also configure the Block Storage service to choose randomly among all available back ends instead; for more information, see [Section 4.2.6, “Configure How Volumes are Allocated to Multiple Back Ends”](#).

4.1.3. Edit a Volume’s Name or Description

1. In the dashboard, select **Project > Compute > Volumes**.

2. Select the volume's **Edit Volume** button.
3. Edit the volume name or description as required.
4. Click **Edit Volume** to save your changes.

**Note**

To create an encrypted volume, you must first have a volume type configured specifically for volume encryption. In addition, both Compute and Block Storage services must be configured to use the same static key. For information on how to set up the requirements for volume encryption, refer to [Section 4.2.5, “Encrypt Volumes with Static Keys”](#).

4.1.4. Delete a Volume

1. In the dashboard, select **Project > Compute > Volumes**.
2. In the **Volumes** table, select the volume to delete.
3. Click **Delete Volumes**.

**Note**

A volume cannot be deleted if it has existing snapshots. For instructions on how to delete snapshots, see [Section 4.1.8, “Create, Clone, or Delete Volume Snapshots”](#).

4.1.5. Attach and Detach a Volume to an Instance

Instances can use a volume for persistent storage. A volume can only be attached to one instance at a time. For more information on instances, see [Section 3.1, “Manage Instances”](#).

4.1.5.1. Attach a Volume to an Instance

1. In the dashboard, select **Project > Compute > Volumes**.
2. Select the volume's **Edit Attachments** action. If the volume is not attached to an instance, the Attach To Instance drop-down list is visible.
3. From the **Attach To Instance** list, select the instance to which you wish to attach the volume.
4. Click **Attach Volume**.

4.1.5.2. Detach a Volume From an Instance

1. In the dashboard, select **Project > Compute > Volumes**.
2. Select the volume's **Manage Attachments** action. If the volume is attached to an instance, the instance's name is displayed in the **Attachments** table.
3. Click **Detach Volume** in this and the next dialog screen.

4.1.6. Set a Volume to Read-Only

You can give multiple users shared access to a single volume without allowing them to edit its contents. To do so, set the volume to **read-only** using the following command:

```
# cinder readonly-mode-update VOLUME true
```

Replace **VOLUME** with the **ID** of the target volume.

To set a read-only volume back to read-write, run:

```
# cinder readonly-mode-update VOLUME false
```

4.1.7. Change a Volume's Owner

To change a volume's owner, you will have to perform a volume transfer. A volume transfer is initiated by the volume's owner, and the volume's change in ownership is complete after the transfer is accepted by the volume's new owner.

4.1.7.1. Transfer a Volume from the Command Line

1. Log in as the volume's current owner.
2. List the available volumes:

```
# cinder list
```

3. Initiate the volume transfer:

```
# cinder transfer-create VOLUME
```

Where **VOLUME** is the name or **ID** of the volume you wish to transfer. For example,

```
+-----+-----+
| Property |          Value          |
+-----+-----+
| auth_key |          f03bf51ce7ead189          |
| created_at |          2014-12-08T03:46:31.884066          |
| id       |          3f5dc551-c675-4205-a13a-d30f88527490          |
| name     |          None          |
| volume_id |          bcf7d015-4843-464c-880d-7376851ca728          |
+-----+-----+
```

The **cinder transfer-create** command clears the ownership of the volume and creates an **id** and **auth_key** for the transfer. These values can be given to, and used by, another user to accept the transfer and become the new owner of the volume.

4. The new user can now claim ownership of the volume. To do so, the user should first log in from the command line and run:

```
# cinder transfer-accept TRANSFERID TRANSFERKEY
```

Where **TRANSFERID** and **TRANSFERKEY** are the **id** and **auth_key** values returned by the

`cinder transfer-create` command, respectively. For example,

```
# cinder transfer-accept 3f5dc551-c675-4205-a13a-d30f88527490
f03bf51ce7ead189
```



Note

You can view all available volume transfers using:

```
# cinder transfer-list
```

4.1.7.2. Transfer a Volume Using the Dashboard

Create a volume transfer from the dashboard

1. As the volume owner in the dashboard, select **Projects > Volumes**.
2. In the **Actions** column of the volume to transfer, select **Create Transfer**.
3. In the **Create Transfer** dialog box, enter a name for the transfer and click **Create Volume Transfer**.

The volume transfer is created and in the **Volume Transfer** screen you can capture the **transfer ID** and the **authorization key** to send to the recipient project.



Note

The authorization key is available only in the **Volume Transfer** screen. If you lose the authorization key, you must cancel the transfer and create another transfer to generate a new authorization key.

4. Close the **Volume Transfer** screen to return to the volume list.

The volume status changes to **awaiting-transfer** until the recipient project accepts the transfer

Accept a volume transfer from the dashboard

1. As the recipient project owner in the dashboard, select **Projects > Volumes**.
2. Click **Accept Transfer**.
3. In the **Accept Volume Transfer** dialog box, enter the **transfer ID** and the **authorization key** that you received from the volume owner and click **Accept Volume Transfer**.

The volume now appears in the volume list for the active project.

4.1.8. Create, Clone, or Delete Volume Snapshots

You can preserve a volume's state at a specific point in time by creating a volume snapshot. You can then use the snapshot to clone new volumes.

Warning

Creating a snapshot of a volume that is attached to an instance may corrupt the snapshot. For instructions on how to detach a volume from an instance, see [Section 4.1.5.2, "Detach a Volume From an Instance"](#).

Note

Volume backups are different from snapshots. Backups preserve the data contained in the volume, whereas snapshots preserve the state of a volume at a specific point in time. In addition, you cannot delete a volume if it has existing snapshots. Volume backups are used to prevent data loss, whereas snapshots are used to facilitate cloning.

For this reason, snapshot back ends are typically co-located with volume back ends in order to minimize latency during cloning. By contrast, a backup repository is usually located in a different location (eg. different node, physical storage, or even geographical location) in a typical enterprise deployment. This is to protect the backup repository from any damage that might occur to the volume back end.

For more information about volume backups, refer to [Section 4.2.1, "Back Up and Restore a Volume"](#)

To create a volume snapshot:

1. In the dashboard, select **Project > Compute > Volumes**.
2. Select the target volume's **Create Snapshot** action.
3. Provide a **Snapshot Name** for the snapshot and click **Create a Volume Snapshot**. The **Volume Snapshots** tab displays all snapshots.

You can clone new volumes from a snapshot once it appears in the **Volume Snapshots** table. To do so, select the snapshot's **Create Volume** action. For more information about volume creation, see [Section 4.1.1, "Create a Volume"](#).

To delete a snapshot, select its **Delete Volume Snapshot** action.

If your OpenStack deployment uses a Red Hat Ceph back end, see [Section 4.1.8.1, "Protected and Unprotected Snapshots in a Red Hat Ceph Back End"](#) for more information on snapshot security and troubleshooting.

4.1.8.1. Protected and Unprotected Snapshots in a Red Hat Ceph Back End

When using Red Hat Ceph as a back end for your OpenStack deployment, you can set a snapshot to *protected* in the back end. Attempting to delete protected snapshots through OpenStack (as in, through the dashboard or the `cinder snapshot-delete` command) will fail.

When this occurs, set the snapshot to *unprotected* in the Red Hat Ceph back end first. Afterwards, you should be able to delete the snapshot through OpenStack as normal.

For related instructions, see [Protecting a Snapshot](#) and [Unprotecting a Snapshot](#).

4.1.9. Upload a Volume to the Image Service

You can upload an existing volume as an image to the Image service directly. To do so:

1. In the dashboard, select **Project > Compute > Volumes**.
2. Select the target volume's **Upload to Image** action.
3. Provide an **Image Name** for the volume and select a **Disk Format** from the list.
4. Click **Upload**. The QEMU disk image utility uploads a new image of the chosen format using the name you provided.

To view the uploaded image, select **Project > Compute > Images**. The new image appears in the **Images** table. For information on how to use and configure images, see [Section 1.2, "Manage Images"](#).

4.2. ADVANCED VOLUME CONFIGURATION

The following procedures describe how to perform advanced volume management. These procedures require administrative privileges.

4.2.1. Back Up and Restore a Volume

A volume backup is a persistent copy of a volume's contents. Volume backups are typically created as object stores, and are managed through the Object Storage service by default. You can, however, set up a different repository for your backups; OpenStack supports Ceph, GlusterFS, and NFS as alternative back ends for backups.

When creating a volume backup, all of the backup's metadata is stored in the Block Storage service's database. The **cinder** utility uses this metadata when restoring a volume from the backup. As such, when recovering from a catastrophic database loss, you must restore the Block Storage service's database first before restoring any volumes from backups. This also presumes that the Block Storage service database is being restored with all the original volume backup metadata intact.

If you wish to configure only a subset of volume backups to survive a catastrophic database loss, you can also export the backup's metadata. In doing so, you can then re-import the metadata to the Block Storage database later on, and restore the volume backup as normal.

Note

Volume backups are different from snapshots. Backups preserve the data contained in the volume, whereas snapshots preserve the state of a volume at a specific point in time. In addition, you cannot delete a volume if it has existing snapshots. Volume backups are used to prevent data loss, whereas snapshots are used to facilitate cloning.

For this reason, snapshot back ends are typically co-located with volume back ends in order to minimize latency during cloning. By contrast, a backup repository is usually located in a different location (eg. different node, physical storage, or even geographical location) in a typical enterprise deployment. This is to protect the backup repository from any damage that might occur to the volume back end.

For more information about volume snapshots, refer to [Section 4.1.8, “Create, Clone, or Delete Volume Snapshots”](#).

4.2.1.1. Create a Full Volume Backup

To back up a volume, use the **cinder backup-create** command. By default, this command will create a full backup of the volume. If the volume has existing backups, you can choose to create an **incremental** backup instead (see [Section 4.2.1.2, “Create an Incremental Volume Backup”](#) for details.)

You can create backups of volumes you have access to. This means that users with administrative privileges can back up any volume, regardless of owner. For more information, see [Section 4.2.1.1.1, “Create a Volume Backup as an Admin”](#).

1. View the **ID** or **Display Name** of the volume you wish to back up:

```
# cinder list
```

2. Back up the volume:

```
# cinder backup-create VOLUME
```

Replace *VOLUME* with the **ID** or **Display Name** of the volume you want to back up. For example:

```
+-----+-----+
| Property |          Value          |
+-----+-----+
|    id    | e9d15fc7-eeae-4ca4-aa72-d52536dc551d |
|   name   |                None      |
| volume_id | 5f75430a-abff-4cc7-b74e-f808234fa6c5 |
+-----+-----+
```

Note

The **volume_id** of the resulting backup is identical to the **ID** of the source volume.

3. Verify that the volume backup creation is complete:

```
# cinder backup-list
```

The volume backup creation is complete when the **Status** of the backup entry is **available**.

At this point, you can also export and store the volume backup's metadata. This allows you to restore the volume backup, even if the Block Storage database suffers a catastrophic loss. To do so, run:

```
# cinder --os-volume-api-version 2 backup-export BACKUPID
```

Where *BACKUPID* is the ID or name of the volume backup. For example,

```
+-----+-----+
| Property | Value |
+-----+-----+
| backup_service | cinder.backup.drivers.swift |
| backup_url | eyJzdGF0dXMiOiAiYXZhaWxhYmxlIiwgIm9iam... |
| | ...4NS02ZmY4MzBhZWYwNWUiLCAic2l6ZSI6IDF9 |
+-----+-----+
```

The volume backup metadata consists of the **backup_service** and **backup_url** values.

4.2.1.1.1. Create a Volume Backup as an Admin

Users with administrative privileges (such as the default **admin** account) can back up any volume managed by OpenStack. When an admin backs up a volume owned by a non-admin user, the backup is hidden from the volume owner by default.

As an admin, you can still back up a volume **and** make the backup available to a specific tenant. To do so, run:

```
# cinder --os-auth-url KEYSTONEURL --os-tenant-name TENANTNAME --os-username USERNAME --os-password PASSWD backup-create VOLUME
```

Where:

- ✦ *TENANTNAME* is the name of the tenant where you want to make the backup available.
- ✦ *USERNAME* and *PASSWD* are the username/password credentials of a user within *TENANTNAME*.
- ✦ *VOLUME* is the name or ID of the volume you want to back up.
- ✦ *KEYSTONEURL* is the URL endpoint of the Identity service (typically `http://IP:5000/v2`, where *IP* is the IP address of the Identity service host).

When performing this operation, the resulting backup's size will count against the quota of *TENANTNAME* rather than the admin's tenant.

4.2.1.1.2. Create an Incremental Volume Backup

By default, the **cinder backup-create** command will create a full backup of a volume. However, if the volume has existing backups, you can choose to create an **incremental** backup.

An incremental backup captures any changes to the volume since the last backup (full or incremental). Performing numerous, regular, full back ups of a volume can become resource-intensive as the volume's size increases over time. In this regard, incremental backups allow you to capture periodic changes to volumes while minimizing resource usage.

To create an incremental volume backup, use the **--incremental** option. As in:

```
# cinder backup-create VOLUME --incremental
```

Replace *VOLUME* with the **ID** or **Display Name** of the volume you want to back up.



Note

You cannot delete a full backup if it already has an incremental backup. In addition, if a full backup has multiple incremental backups, you can only delete the latest one.

Incremental backups are fully supported on NFS and Object Storage backup repositories. Ceph backup repositories also support incremental backups, but only for volumes that are also stored on a Ceph back end.

4.2.1.3. Restore a Volume After a Block Storage Database Loss

Typically, a Block Storage database loss prevents you from restoring a volume backup. This is because the Block Storage database contains metadata required by the volume backup service (**openstack-cinder-backup**). This metadata consists of **backup_service** and **backup_url** values, which you can export after creating the volume backup (as shown in [Section 4.2.1.1, "Create a Full Volume Backup"](#)).

If you exported and stored this metadata, then you can import it to a new Block Storage database (thereby allowing you to restore the volume backup).

1. As a user with administrative privileges, run:

```
# cinder --os-volume-api-version 2 backup-import backup_service
backup_url
```

Where *backup_service* and *backup_url* are from the metadata you exported. For example, using the exported metadata from [Section 4.2.1.1, "Create a Full Volume Backup"](#):

```
# cinder --os-volume-api-version 2 backup-import
cinder.backup.drivers.swift eyJzdGF0dXMi...c2l6ZSI6IDF9
+-----+-----+
| Property | Value |
+-----+-----+
| id       | 77951e2f-4aff-4365-8c64-f833802eaa43 |
| name     | None |
+-----+-----+
```

2. After the metadata is imported into the Block Storage service database, you can restore the volume as normal (see [Section 4.2.1.4, "Restore a Volume from a Backup"](#)).

4.2.1.4. Restore a Volume from a Backup

1. Find the **ID** of the volume backup you wish to use:

```
# cinder backup-list
```

The **Volume ID** should match the ID of the volume you wish to restore.

2. Restore the volume backup:

```
# cinder backup-restore BACKUP_ID
```

Where *BACKUP_ID* is the ID of the volume backup you wish to use.

3. If you no longer need the backup, delete it:

```
# cinder backup-delete BACKUP_ID
```

4.2.1.5. View and Modify a Tenant's Backup Quota

Unlike most tenant storage quotas (number of volumes, volume storage, snapshots, etc.), backup quotas cannot be modified through the dashboard yet.

Backup quotas can only be modified through the command-line interface; namely, through the **cinder quota-update** command.

To view the storage quotas of a specific tenant (*TENANTNAME*), run:

```
# cinder quota-show TENANTNAME
```

To update the maximum number of backups (*MAXNUM*) that can be created in a specific tenant, run:

```
# cinder quota-update --backups MAXNUM TENANTNAME
```

To update the maximum total size of all backups (*MAXGB*) within a specific tenant, run:

```
# cinder quota-update --backup-gigabytes MAXGB TENANTNAME
```

To view the storage quota usage of a specific tenant, run:

```
# cinder quota-usage TENANTNAME
```

4.2.1.6. Enable Volume Backup Management Through the Dashboard

You can now create, view, delete, and restore volume backups through the dashboard. To perform any of these functions, go to the **Project > Compute > Volumes > Volume Backups** tab.

However, the **Volume Backups** tab is not enabled by default. To enable it, configure the dashboard accordingly:

1. Open `/etc/openstack-dashboard/local_settings`.

2. Search for the following setting:

```
OPENSTACK_CINDER_FEATURES = {
    'enable_backup': False,
}
```

Change this setting to:

```
OPENSTACK_CINDER_FEATURES = {
    'enable_backup': True,
}
```

3. Restart the dashboard by restarting the **httpd** service:

```
# systemctl restart httpd.service
```

4.2.1.7. Set an NFS Share as a Backup Repository

By default, the Block Storage service uses the Object Storage service as a repository for backups. You can configure the Block Storage service to use an existing NFS share as a backup repository instead. To do so:

1. Log in to the node hosting the backup service (**openstack-cinder-backup**) as a user with administrative privileges.
2. Configure the Block Storage service to use the NFS backup driver (**cinder.backup.drivers.nfs**):

```
# openstack-config --set /etc/cinder/cinder.conf DEFAULT
backup_driver cinder.backup.drivers.nfs
```

3. Set the details of the NFS share that you want to use as a backup repository:

```
# openstack-config --set /etc/cinder/cinder.conf DEFAULT
backup_share NFSHOST:PATH
```

Where:

- ✧ *NFSHOST* is the IP address or hostname of the NFS server.
- ✧ *PATH* is the absolute path of the NFS share on *NFSHOST*.

4. If you want to set any optional mount settings for the NFS share, run:

```
# openstack-config --set /etc/cinder/cinder.conf DEFAULT
backup_mount_options NFSMOUNTOPTS
```

Where *NFSMOUNTOPTS* is a comma-separated list of NFS mount options (for example, **rw, sync**). For more information on supported mount options, see the **man** pages for **nfs** and **mount**.

5. Restart the Block Storage backup service to apply your changes:

```
# systemctl restart openstack-cinder-backup.service
```

4.2.1.7.1. Set a Different Backup File Size

The backup service limits backup files sizes to a maximum **backup file size**. If you are backing up a volume that exceeds this size, the resulting backup will be split into multiple chunks. The default backup file size is 1.8GB.

To set a different backup file size, run:

```
# openstack-config --set /etc/cinder/cinder.conf DEFAULT
  backup_file_size SIZE
```

Replace *SIZE* with the file size you want, in bytes. Restart the Block Storage backup service to apply your changes:

```
# systemctl restart openstack-cinder-backup.service
```

4.2.2. Migrate a Volume

Only an administrator can migrate volumes; volumes to be migrated cannot be in use nor can they have any snapshots.

1. As an administrative user, list all available volumes:

```
# cinder list
```

2. List the available back ends (hosts) and their respective availability zones:

```
# cinder-manage host list
```

3. Initiate the migration:

```
# cinder migrate VOLUME BACKEND
```

Where:

- ✧ **VOLUME** is the **ID** of the volume to be migrated.
- ✧ **BACKEND** is the back end to where the volume should be migrated.

4. View the current status of the volume to be migrated:

```
# cinder show VOLUME
```

For example,

```
# cinder show 45a85c3c-3715-484d-ab5d-745da0e0bd5a
+-----+-----+
|                Property                |          Value          |
+-----+-----+
|                ...                |          ...            |
|      os-vol-host-attr:host              |          server1        |
|      os-vol-mig-status-attr:migstat     |          None           |
|                ...                |          ...            |
+-----+-----+
```

During migration, note the following attributes:

os-vol-host-attr:host

The volume's current back end. Once the migration completes, this displays the target back end (namely, BACKEND).

os-vol-mig-status-attr:migstat

The status of the migration. A status of **None** means a migration is no longer in progress.

4.2.3. Group Volume Settings with Volume Types

OpenStack allows you to create volume types, which allows you apply the type's associated settings when creating a volume (Section 4.1.1, "Create a Volume"). For example, you can associate:

- ✦ Whether or not a volume is encrypted (Section 4.2.5.2, "Configure Volume Type Encryption")
- ✦ Which back end a volume should use (Section 4.1.2, "Specify Back End for Volume Creation")
- ✦ Quality-of-Service (QoS) Specs

Settings are associated with volume types using key-value pairs called Extra Specs. When you specify a volume type during volume creation, the Block Storage scheduler applies these key/value pairs as settings. You can associate multiple key/value pairs to the same volume type.

Volume types provide the capability to provide different users with storage tiers. By associating specific performance, resilience, and other settings as key/value pairs to a volume type, you can map tier-specific settings to different volume types. You can then apply tier settings when creating a volume by specifying the corresponding volume type.



Note

Available and supported Extra Specs vary per volume driver. Consult your volume driver's documentation for a list of valid Extra Specs.

4.2.3.1. Create and Configure a Volume Type

1. As an admin user in the dashboard, select **Admin > Volumes > Volume Types**.
2. Click **Create Volume Type**.
3. Enter the volume type name in the **Name** field.
4. Click **Create Volume Type**. The new type appears in the **Volume Types** table.
5. Select the volume type's **View Extra Specs** action.
6. Click **Create**, and specify the **Key** and **Value**. The key/value pair must be valid; otherwise, specifying the volume type during volume creation will result in an error.
7. Click **Create**. The associated setting (key/value pair) now appears in the **Extra Specs** table.

By default, all volume types are accessible to all OpenStack tenants. If you need to create volume types with restricted access, you will need to do so through the CLI. For instructions, see Section 4.2.3.4, "Create and Configure Private Volume Types".



Note

You can also associate a QOS Spec to the volume type. For details, refer to [Section 4.2.4.2, “Associate a QOS Spec with a Volume Type”](#).

4.2.3.2. Edit a Volume Type

1. As an admin user in the dashboard, select **Admin > Volumes > Volume Types**.
2. In the **Volume Types** table, select the volume type's **View Extra Specs** action.
3. On the **Extra Specs** table of this page, you can:
 - ✦ Add a new setting to the volume type. To do this, click **Create**, and specify the key/value pair of the new setting you want to associate to the volume type.
 - ✦ Edit an existing setting associated with the volume type. To do this, select the setting's **Edit** action.
 - ✦ Delete existing settings associated with the volume type. To do this, select the extra specs' check box and click **Delete Extra Specs** in this and the next dialog screen.

4.2.3.3. Delete a Volume Type

To delete a volume type, select its corresponding check boxes from the **Volume Types** table and click **Delete Volume Types**.

4.2.3.4. Create and Configure Private Volume Types

By default, all volume types are visible to all tenants. You can override this during volume type creation and set it to **private**. To do so, you will need to set the type's **Is_Public** flag to **False**.

Private volume types are useful for restricting access to certain volume settings. Typically, these are settings that should only be usable by specific tenants; examples include new back ends or ultra-high performance configurations that are being tested.

To create a private volume type, run:

```
# cinder --os-volume-api-version 2 type-create --is-public false
_VTYPE_
```

+ Replace *VTYPE* with the name of the private volume type.

By default, private volume types are only accessible to their creators. However, admin users can find and view private volume types using the following command:

```
# cinder --os-volume-api-version 2 type-list --all
```

This command will list both public and private volume types, and will also include the name and ID of each one. You will need the volume type's ID to provide access to it.

Access to a private volume type is granted at the tenant level. To grant a tenant access to a private volume type, run:

```
# cinder --os-volume-api-version 2 type-access-add --volume-type
_VTYPEID_ --project-id _TENANTID_
```

Where:

- ✦ *VTYPEID* is the ID of the private volume type.
- ✦ *TENANTID* is the ID of the project/tenant you are granting access to *VTYPEID*.

To view which tenants have access to a private volume type, run:

```
# cinder --os-volume-api-version 2 type-access-list --volume-type
_VTYPE_
```

To remove a tenant from the access list of a private volume type, run:

```
# cinder --os-volume-api-version 2 type-access-remove --volume-type
_VTYPE_ --project-id _TENANTID_
```



Note

By default, only users with administrative privileges can create, view, or configure access for private volume types.

4.2.4. Use Quality-of-Service Specifications

You can map multiple performance settings to a single Quality-of-Service specification (QOS Specs). Doing so allows you to provide performance tiers for different user types.

Performance settings are mapped as key/value pairs to QOS Specs, similar to the way volume settings are associated to a volume type. However, QOS Specs are different from volume types in the following respects:

- ✦ QOS Specs are used to apply performance settings, which include limiting read/write operations to disks. Available and supported performance settings vary per storage driver.

To determine which QOS Specs are supported by your back end, consult the documentation of your back end device's volume driver.

- ✦ Volume types are directly applied to volumes, whereas QOS Specs are not. Rather, QOS Specs are associated to volume types. During volume creation, specifying a volume type also applies the performance settings mapped to the volume type's associated QOS Specs.

4.2.4.1. Create and Configure a QOS Spec

As an administrator, you can create and configure a QOS Spec through the QOS Specs table. You can associate more than one key/value pair to the same QOS Spec.

1. As an admin user in the dashboard, select **Admin > Volumes > Volume Types**.
2. On the **QOS Specs** table, click **Create QOS Spec**.
3. Enter a name for the **QOS Spec**.

- In the **Consumer** field, specify where the QOS policy should be enforced:

Table 4.1. Consumer Types

Type	Description
back-end	QOS policy will be applied to the Block Storage back end.
front-end	QOS policy will be applied to Compute.
both	QOS policy will be applied to both Block Storage and Compute.

- Click **Create**. The new QOS Spec should now appear in the **QOS Specs** table.
- In the **QOS Specs** table, select the new spec's **Manage Specs** action.
- Click **Create**, and specify the **Key** and **Value**. The key/value pair must be valid; otherwise, specifying a volume type associated with this QOS Spec during volume creation will fail.
- Click **Create**. The associated setting (key/value pair) now appears in the **Key-Value Pairs** table.

4.2.4.2. Associate a QOS Spec with a Volume Type

As an administrator, you can associate a QOS Spec to an existing volume type using the **Volume Types** table.

- As an administrator in the dashboard, select **Admin > Volumes > Volume Types**.
- In the **Volume Types** table, select the type's **Manage QOS Spec Association** action.
- Select a QOS Spec from the **QOS Spec to be associated** list.
- Click **Associate**. The selected QOS Spec now appears in the **Associated QOS Spec** column of the edited volume type.

4.2.4.3. Disassociate a QOS Spec from a Volume Type

- As an administrator in the dashboard, select **Admin > Volumes > Volume Types**.
- In the **Volume Types** table, select the type's **Manage QOS Spec Association** action.
- Select **None** from the QOS Spec to be associated list.
- Click **Associate**. The selected QOS Spec is no longer in the **Associated QOS Spec** column of the edited volume type.

4.2.5. Encrypt Volumes with Static Keys

Volume encryption helps provide basic data protection in case the volume back-end is either compromised or outright stolen. The contents of an encrypted volume can only be read with the use of a specific key; both Compute and Block Storage services must be configured to use the same key in order for instances to use encrypted volumes. This section describes how to configure an OpenStack deployment to use a single key for encrypting volumes.



Important

At present, volume encryption is only supported on volumes backed by block devices. Encryption of network-attached volumes (such as RBD) or file-based volumes (such as NFS) is still unsupported.

4.2.5.1. Configure a Static Key

The first step in implementing basic volume encryption is to set a *static key*. This key must be a hex string, which will be used by the Block Storage volume service (namely, **openstack-cinder-volume**) and all Compute services (**openstack-nova-compute**). To configure both services to use this key, set the key as the **fixed_key** value in the **[keymgr]** section of both service's respective configuration files.

1. From the command line, log in as **root** to the node hosting **openstack-cinder-volume**.
2. Set the static key:

```
# openstack-config --set /etc/cinder/cinder.conf keymgr fixed_key
HEX_KEY
```

Replace **HEX_KEY** with a 16-digit alphanumeric hex key (for example, **00**).

3. Restart the Block Storage volume service:

```
# openstack-service restart cinder-volume
```

4. Log in to the node hosting **openstack-nova-compute**, and set the same static key:

```
# openstack-config --set /etc/nova/nova.conf keymgr fixed_key
HEX_KEY
```



Note

If you have multiple Compute nodes (multiple nodes hosting **openstack-nova-compute**), then you need to set the same static key in **/etc/nova/nova.conf** of each node.

5. Restart the Compute service:

```
# openstack-service restart nova-compute
```



Note

Likewise, if you set the static key on multiple Compute nodes, you need to restart the **openstack-nova-compute** service on each node as well.

At this point, both Compute and Block Storage volume services can now use the same static key to encrypt/decrypt volumes. That is, new instances will be able to use volumes encrypted with the static key (**HEX_KEY**).

4.2.5.2. Configure Volume Type Encryption

To create volumes encrypted with the static key from [Section 4.2.5.1, “Configure a Static Key”](#), you need an *encrypted volume type*. Configuring a volume type as encrypted involves setting what provider class, cipher, and key size it should use. To do so, run:

```
# cinder encryption-type-create --cipher aes-xts-plain64 --key_size
BITSIZE --control_location front-end VOLTYPE
nova.volume.encryptors.luks.LuksEncryptor
```

Where:

- ✦ **BITSIZE** is the key size (for example, **512** for a 512-bit key).
- ✦ **VOLTYPE** is the name of the volume type you want to encrypt.

This command sets the **nova.volume.encryptors.luks.LuksEncryptor** provider class and **aes-xts-plain64** cipher. As of this release, this is the only supported class/cipher configuration for volume encryption.

Once you have an encrypted volume type, you can invoke it to automatically create encrypted volumes. Specifically, select the encrypted volume type from the Type drop-down list in the **Create Volume** window (see to [Section 4.1, “Basic Volume Usage and Configuration”](#)).

4.2.6. Configure How Volumes are Allocated to Multiple Back Ends

If the Block Storage service is configured to use multiple back ends, you can use configured volume types to specify where a volume should be created. For details, see [Section 4.1.2, “Specify Back End for Volume Creation”](#).

The Block Storage service will automatically choose a back end if you do not specify one during volume creation. Block Storage sets the first defined back end as a default; this back end will be used until it runs out of space. At that point, Block Storage will set the second defined back end as a default, and so on.

If this is not suitable for your needs, you can use the filter scheduler to control how Block Storage should select back ends. This scheduler can use different filters to triage suitable back ends, such as:

AvailabilityZoneFilter

Filters out all back ends that do not meet the availability zone requirements of the requested volume

CapacityFilter

Selects only back ends with enough space to accommodate the volume

CapabilitiesFilter

Selects only back ends that can support any specified settings in the volume

To configure the filter scheduler:

1. Enable the **FilterScheduler**.

```
# openstack-config --set /etc/cinder/cinder.conf DEFAULT
scheduler_driver
cinder.scheduler.filter_scheduler.FilterScheduler
```

2. Set which filters should be active:

```
# openstack-config --set /etc/cinder/cinder.conf DEFAULT
scheduler_default_filters
AvailabilityZoneFilter, CapacityFilter, CapabilitiesFilter
```

3. Configure how the scheduler should select a suitable back end. If you want the scheduler:

- ✦ To always choose the back end with the most available free space, run:

```
# openstack-config --set /etc/cinder/cinder.conf DEFAULT
scheduler_default_weighters AllocatedCapacityWeigher
# openstack-config --set /etc/cinder/cinder.conf DEFAULT
allocated_capacity_weight_multiplier -1.0
```

- ✦ To choose randomly among all suitable back ends, run:

```
# openstack-config --set /etc/cinder/cinder.conf DEFAULT
scheduler_default_weighters ChanceWeigher
```

4. Restart the Block Storage scheduler to apply your settings:

```
# openstack-service restart openstack-cinder-scheduler
```

CHAPTER 5. MANAGE CONTAINERS

OpenStack Object Storage (swift) stores its objects (data) in containers, which are similar to directories in a file system although they cannot be nested. Containers provide an easy way for users to store any kind of unstructured data; for example, objects might include photos, text files, or images. Stored objects are not encrypted nor are they compressed.

To help with organization, pseudo-folders are logical devices that can contain objects (and can be nested). For example, you might create an *Images* folder in which to store pictures and a *Media* folder in which to store videos.

You can create one or more containers in each project, and one or more objects or pseudo-folders in each container.

5.1. CREATE A CONTAINER

1. In the dashboard, select **Project > Object Store > Containers**
2. Click **Create Container**.
3. Specify the **Container Name**, and select one of the following in the **Container Access** field.

Type	Description
Private	Limits access to a user in the current project.
Public	Permits API access to anyone with the public URL. However, in the dashboard, project users cannot see public containers and data from other projects.

4. Click **Create Container**.

5.2. CREATE PSEUDO FOLDER FOR CONTAINER

1. In the dashboard, select **Project > Object Store > Containers**
2. Click the name of the container to which you want to add the pseudo-folder.
3. Click **Create Pseudo-folder**.
4. Specify the name in the **Pseudo-folder Name** field, and click **Create**.

5.3. UPLOAD AN OBJECT

If you do not upload an actual file, the object is still created (as placeholder) and can later be used to upload the file.

1. In the dashboard, select **Project > Object Store > Containers**
2. Click the name of the container in which the uploaded object will be placed; if a pseudo-folder already exists in the container, you can click its name.
3. Browse for your file, and click **Upload Object**.
4. Specify a name in the **Object Name** field:
 - ✦ Pseudo-folders can be specified in the name using a / character (for example, *Images/myImage.jpg*). If the specified folder does not already exist, it is created when the object is uploaded.
 - ✦ A name that is not unique to the location (that is, the object already exists) overwrites the object's contents.
5. Click **Upload Object**.

5.4. COPY AN OBJECT

1. In the dashboard, select **Project > Object Store > Containers**
2. Click the name of the object's container or folder (to display the object).
3. Click **Upload Object**.
4. Browse for the file to be copied, and select **Copy** in its arrow menu.
5. Specify the following:

Field	Description
Destination container	Target container for the new object.
Path	Pseudo-folder in the destination container; if the folder does not already exist, it is created.
Destination object name	New object's name. If you use a name that is not unique to the location (that is, the object already exists), it overwrites the object's previous contents.

6. Click **Copy Object**.

5.5. DELETE AN OBJECT

1. In the dashboard, select **Project > Object Store > Containers**
2. Browse for the object, and select **Delete Object** in its arrow menu.

3. Click **Delete Object** to confirm the object's removal.

5.6. DELETE A CONTAINER

1. In the dashboard, select **Project > Object Store > Containers**
2. Browse for the container in the **Containers** section, and ensure all objects have been deleted (see [Section 5.5, "Delete an Object"](#)).
3. Select **Delete Container** in the container's arrow menu.
4. Click **Delete Container** to confirm the container's removal.

5.7. ERASURE CODING FOR OBJECT STORAGE SERVICE

Erasure coding (EC) is a method of data protection in which the data is broken into fragments, expanded and encoded with redundant data pieces and stored across a set of different locations or storage media. It uses a smaller volume of storage to attain the required durability than traditional replication. When compared to replication factor of 3, savings of 50% may be attained with careful deployment. However, depending on the workload, erasure coding may incur a performance penalty.

With the RHEL OpenStack Platform 7 release, erasure coding support is available as a technology preview for Object Storage service. For more information on the support scope for features marked as technology previews, refer to <https://access.redhat.com/support/offerings/techpreview/>.

Erasure coding is supported for Object Storage service as a Storage Policy. A Storage Policy allows segmenting the cluster for various purposes through the creation of multiple object rings. Red Hat recommends you split off devices used by erasure coding and replication Storage Policies. This way behavior of the cluster is easier to analyze.

The direction you choose depends on why the erasure coding policy is being deployed. Some of the main considerations are:

- ✦ Layout of existing infrastructure.
- ✦ Cost of adding dedicated erasure coding nodes (or just dedicated erasure coding devices).
- ✦ Intended usage model(s).

5.7.1. Configure Erasure Coding

To use an erasure coding policy, define an erasure coding policy in **swift.conf** file and create, configure the associated object ring. An example of how an erasure coding policy can be setup is shown below:

```
[storage-policy:2]
name = ec104
policy_type = erasure_coding
ec_type = jerasure_rs_vand
ec_num_data_fragments = 10
ec_num_parity_fragments = 4
ec_object_segment_size = 1048576
```

The following table describes the terms in the storage policy:

name	This is a standard storage policy parameter.
policy_type	Set this to <code>erasure_coding</code> to indicate that this is an erasure coding policy.
ec_type	Set this value according to the available options in the selected PyECLib back-end. This specifies the erasure coding scheme that is to be used. For example, the option shown here selects Vandermonde Reed-Solomon encoding while an option of <code>flat_xor_hd_3</code> would select Flat-XOR based HD combination codes. See the PyECLib page for full details.
ec_num_data_fragments	The total number of fragments that will be comprised of data.
ec_num_parity_fragments	The total number of fragments that will be comprised of parity.
ec_object_segment_size	The amount of data that will be buffered up before feeding a segment into the encoder/decoder. The default value is 1048576.

When PyECLib encodes an object, it breaks it into N fragments. It is important during configuration to know how many of those fragments are data and how many are parity. So in the example above, PyECLib will break an object in 14 different fragments, 10 of them will be made up of actual object data and 4 of them will be made of parity data (calculations depending on `ec_type`). With such a configuration, the system can sustain 4 disk failures before the data is lost. Other commonly used configurations are 4+2 (with 4 data fragments and 2 parity fragments) or 8+3 (with 8 data fragments and 3 parity fragments).



Note

It is important to note that once you have deployed a policy and have created objects with that policy, these configuration options cannot be changed. In case a change in the configuration is desired, you must create a new policy and migrate the data to a new container. However, once defined, policy indices cannot be discarded. If policies are to be retired, they may be disabled, but not be removed. There is essentially no performance penalty for having old policies around, but a minor administrative overhead.

5.7.2. Configure an Object Storage Ring

Object Storage uses a data structure called the **Ring** to distribute a partition space across the cluster. This partition space is core to the replication system in Object Storage service. It allows the Object Storage service to quickly and easily synchronize each partition across the cluster. When

any component in Swift needs to interact with data, a quick lookup is done locally in the Ring to determine the possible partitions for each replica.

Object Storage service already has three rings to store different types of data. There is one for account information, another for containers (so that it's convenient to organize objects under an account) and another for the object replicas. To support erasure codes, there will be an additional ring that is created to store erasure code chunks.

To create a typical replication ring, for example, you can use the following command:

```
swift-ring-builder object-1.builder create 10 3 1
```

where 3 is the number of replicas.

In order to create an erasure coding object ring, you need to use the number of fragments in place of the number of replicas, for example:

```
swift-ring-builder object-1.builder create 10 14 1
```

where 14 is for a 10+4 configuration with 10 data fragments and 4 parity fragments.

Consider the performance impacts when deciding which devices to use in the erasure coding policy's object ring. We recommend that you run some performance benchmarking in a test environment for the configuration before deployment. After you have configured your erasure coding policy in the **swift.conf** and created your object ring, your application is ready to start using erasure coding by creating a container with the specified policy name and interacting as usual.

5.8. SET OBJECT STORAGE AS A BACK END FOR THE IMAGE SERVICE

The OpenStack Image service, by default, saves images and instance snapshots to the local filesystem in **/var/lib/glance/images/**. Alternatively, you can configure the Image service to save images and snapshots to the Object Storage service (when available).

To do so, perform the following procedure:

1. Log into the node running the Image service (the controller node also running Identity) as root and source your OpenStack credentials (this is typically a file named **openrc**).

```
# source ~/openrc
```

2. Verify that the Image service is part of the tenant **service** with role **admin**.

```
# keystone user-role-list --user glance --tenant service
```

One of the roles returned should be **admin**.

3. Open the **/etc/glance/glance.conf** file and comment out the following lines:

```
##### DEFAULT OPTIONS #####
#default_store = file
#filesystem_store_datadir = /var/lib/glance/images/
```

4. In the same file, add the following lines to the **DEFAULT OPTIONS** section.


```
default_store = swift
swift_store_auth_address = http://KEYSTONEIP:35357/v2.0/
swift_store_user = service:glance
swift_store_key = ADMINPW
swift_store_create_container_on_put = True
```

Where:

- » **KEYSTONEIP** is the IP address of the Identity service, and
- » **ADMINPW** is the value of admin password attribute in the `/etc/glance/glance-api.conf` file.

5. Apply the changes by restarting the Image service:

```
# systemctl restart openstack-glance-api
# systemctl restart openstack-glance-registry
```

From this point onwards, images uploaded to the Image service (whether through the Dashboard or **glance**) should now be saved to an Object Storage container named **glance**. This container exists in the service account.

To verify whether newly-created images are saved to the Image service, run:

```
# ls /var/lib/glance/images
```

Once the Dashboard or the **glance image-list** reports the image is active, you can verify whether it is in Object Storage by running the following command:

```
# swift --os-auth-url http://KEYSTONEIP:5000/v2.0 --os-tenant-name
service --os-username glance --os-password ADMINPW list glance
```

CHAPTER 6. CONFIGURE OPENSTACK TO USE AN NFS BACK END

This chapter describes how to configure the OpenStack volume service (**openstack-cinder-volume**) to use an existing NFS server as an additional back end. In addition, this chapter also describes how to create a volume type that you can use to invoke to create volumes backed by the NFS share.

Prerequisites:

- ✦ The NFS share that you will be using as a back end should already be properly configured.
- ✦ The node hosting the OpenStack volume service should have read/write access to the NFS share.
- ✦ You have **root** access to the node hosting the OpenStack volume service.

Assumptions:

- ✦ Your OpenStack deployment was not provisioned through the Red Hat Enterprise Linux OpenStack Platform Installer.
- ✦ Your OpenStack Block Storage service uses the default back end (which uses the back end name **lvm**, as deployed by Packstack).

6.1. CONFIGURE SELINUX

If a client has SELinux enabled, you should also enable the `virt_use_nfs` Boolean if the client requires access to NFS volumes on an instance. To enable this Boolean (and make it persistent through reboots), run the following command as **root**:

```
# setsebool -P virt_use_nfs on
```

Run this command on all client hosts that require access to NFS volumes on an instance. This includes all Compute nodes.

6.2. CONFIGURE THE SHARE

The first step in adding an NFS back end is defining the NFS share that the OpenStack volume service should use. To do so:

1. Log in as root to the node hosting the OpenStack volume service.
2. Create a new text file named `nfs_share` in the `/etc/cinder/` directory:

```
/etc/cinder/nfs_share
```

3. Define the NFS share in `/etc/cinder/nfs_share` using the following format:

```
HOST : SHARE
```

Where:

- ✧ HOST is the IP address or hostname of the NFS server.
 - ✧ SHARE is the absolute path to the NFS shares exported on HOST.
4. Set the root user and cinder group as the owner of `/etc/cinder/nfs_share`:

```
# chown root:cinder /etc/cinder/nfs_share
```

5. Finally, configure `/etc/cinder/nfs_share` to be readable by members of the cinder group:

```
# chmod 0640 /etc/cinder/nfs_share
```

6.3. CREATE A NEW BACK END DEFINITION

By default, Packstack creates a back end definition for LVM in `/etc/cinder/cinder.conf`:

```
[lvm]
iscsi_helper=lioadm
volume_group=cinder-volumes
iscsi_ip_address=
volume_driver=cinder.volume.drivers.lvm.LVMISCSIDriver
volume_backend_name=lvm
```

After defining the NFS share in `/etc/cinder/cinder.conf`, you can now configure an additional back end definition for it. To do so:

1. Log in as **root** to the node hosting the OpenStack volume service.
2. Create a new definition for the NFS back end and set the volume service to use the file defining the NFS share (namely, `/etc/cinder/nfs_share`):

```
# openstack-config --set /etc/cinder/cinder.conf nfs
nfs_shares_config /etc/cinder/nfs_shares
```

Here, we use the name **nfsbackend** as a definition name.

3. Configure the volume service to use the NFS volume driver, namely **cinder.volume.drivers.nfs.NfsDriver**:

```
# openstack-config --set /etc/cinder/cinder.conf nfs
volume_driver cinder.volume.drivers.nfs.NfsDriver
```

4. Define a volume back end name for the NFS back end (the following command uses the name **nfs**):

```
# openstack-config --set /etc/cinder/cinder.conf nfs
volume_backend_name nfsbackend
```

5. Add any mount options (**MOUNTOPTIONS**) you need to the `nfs_mount_options` configuration key:

```
# openstack-config --set /etc/cinder/cinder.conf nfs
nfs_mount_options _MOUNTOPTIONS_
```

At this point, the following section should now appear in `/etc/cinder/cinder.conf`:

```
[nfs]
nfs_shares_config = /etc/cinder/nfs_shares
volume_driver = cinder.volume.drivers.nfs.NfsDriver
volume_backend_name = nfsbackend
nfs_mount_options =
```

You can now enable the NFS back end. Back ends are enabled through the `enabled_backends` configuration key of `/etc/cinder/cinder.conf`. The default back end created by Packstack should already be listed there:

```
enabled_backends=lvm
```

Add the new NFS back end definition to this list, as in:

```
enabled_backends=lvm,nfs
```

Once the NFS back end is enabled, restart the OpenStack volume service:

```
# openstack-service restart cinder-volume
```

6.4. CREATE A VOLUME TYPE FOR THE NFS BACK END

The new NFS back end is now available, but cannot be used yet when creating new volumes. To configure new volumes to use this NFS back end, you need to first create a *volume type* for it.

1. View the existing volume types. By default, a volume type should already exist for the lvm back end (namely, `iscsi`):

```
+-----+-----+
|                ID                | Name |
+-----+-----+
| f8d31dc8-a20e-410c-81bf-6b0a971c61a0 | iscsi |
+-----+-----+
```

2. Create a new volume type named `nfstype` for the NFS back end:

```
# cinder type-create nfstype
```

3. Configure the `nfstype` volume type to use the NFS back end through the back end's name (namely, `nfsbackend`):

```
# cinder type-key nfstype set volume_backend_name=nfsbackend
```

4. Verify that the new type was created and configured correctly:

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

```

|          ID          | Name |
+-----+-----+
| bfff44b5-52b1-43d6-beb4-83aa2d20bc59 | nfstype |
| f8d31dc8-a20e-410c-81bf-6b0a971c61a0 | iscsi |
+-----+-----+
+-----+-----+
|          ID          | Name |          extra_specs
|
+-----+-----+
+-----+
|bfff44b5-~-83aa2d20bc59|nfstype|{u'volume_backend_name':
u'nfsbackend'}|
|f8d31dc8-~-6b0a971c61a0| iscsi |    {u'volume_backend_name':
u'lvn'} |
+-----+-----+
+-----+

```



Note

You can also create and configure volume types through the dashboard. For more information, see [Section 4.2.3, “Group Volume Settings with Volume Types”](#).

6.5. TEST THE NEW NFS BACK END

To test the new NFS back end, create a new volume named **nfsvolume** while invoking the volume type **nfstype**:

```

+-----+-----+
|          Property          |          Value          |
+-----+-----+
|      attachments          |          []              |
|  availability_zone        |          nova            |
|      bootable             |          false           |
|      created_at           |          2015-01-06T05:14:09.271114 |
|  display_description      |          None            |
|      display_name         |          nfsvolume       |
|      encrypted            |          False           |
|      id                    |          0cd7ac45-622a-47b0-9503-7025bbcdc8ed |
|      metadata              |          {}              |
|      size                  |          1                |
|      snapshot_id          |          None            |
|      source_volid         |          None            |
|      status                |          creating        |
|      volume_type          |          nfstype         |
+-----+-----+

```

Once the volume is successfully created, check the NFS share (on the NFS server). A corresponding volume (whose name contains the ID of the newly-created volume) should appear there:

```
drwxrwxrwx. 2 root      root      4.0K Jan  6 15:14 .
drwxr-xr-x. 18 root      root      4.0K Jan  5 04:03 ..
-rw-rw-rw-. 1 nfsnobody nfsnobody 1.0G Jan  6 15:14+ +volume-
0cd7ac45-622a-47b0-9503-7025bbedc8ed
```

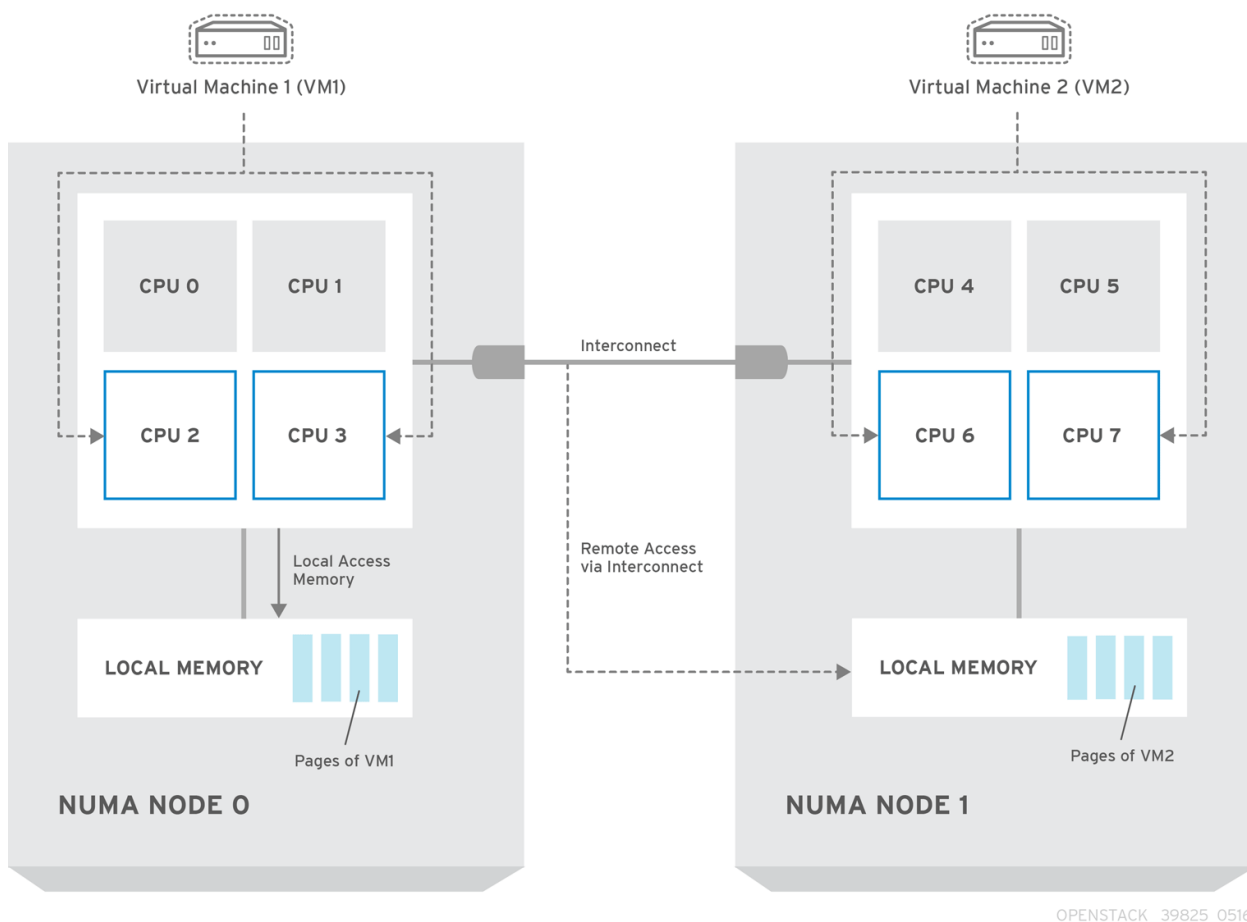
CHAPTER 7. CONFIGURE CPU PINNING WITH NUMA

This chapter concerns NUMA topology awareness and the configuration of an OpenStack environment on systems supporting this technology. With this setup, virtual machine instances are pinned to dedicated CPU cores, which enables smarter scheduling and therefore improves guest performance.

Tip

Background information about NUMA is available in the following article: [What is NUMA and how does it work on Linux ?](#).

The following diagram provides an example of a two-node NUMA system and the way the CPU cores and memory pages are made available:



OPENSTACK_39825_0516

Note

Remote memory available via Interconnect is accessed **only** if VM1 from NUMA node 0 has a CPU core in NUMA node 1. In this case, the memory of NUMA node 1 will act as local for the third CPU core of VM1 (for example, if VM1 is allocated with CPU 4 in the diagram above), but at the same time, it will act as remote memory for the other CPU cores of the same VM.

For more details on NUMA tuning with libvirt, see the [Virtualization Tuning and Optimization Guide](#).

Warning

At present, it is impossible to migrate an instance which has been configured to use CPU pinning. For more information about this issue, see the following solution: [Instance migration fails when using cpu-pinning from a numa-cell and flavor-property "hw:cpu_policy=dedicated"](#).

7.1. COMPUTE NODE CONFIGURATION

The exact configuration depends on the NUMA topology of your host system; however, you must reserve some CPU cores across all the NUMA nodes for host processes and let the rest of the CPU cores handle your guest virtual machine instances. For example, with eight CPU cores evenly spread across two NUMA nodes, the layout can be illustrated as follows:

Table 7.1. Example of NUMA Topology

	Node 0		Node 1	
Host processes	Core 0	Core 1	Core 4	Core 5
Guest processes	Core 2	Core 3	Core 6	Core 7

**Note**

The number of cores to reserve for host processes should be determined by observing the performance of the host under typical workloads.

The configuration of the Compute nodes consists of the following steps:

1. Set the **vcpu_pin_set** option in the `/etc/nova/nova.conf` file to the list of CPU cores reserved for guest processes. Using the example above, you would set:

```
vcpu_pin_set=2,3,6,7
```

The **vcpu_pin_set** option will also ensure that a **cpuset** attribute similar to the following will be added to the XML configuration file for libvirt:

```
<vcpu placement='static' cpuset='2-3,6-7'>1</vcpu>
```

This will pin the guest vCPUs to the listed physical CPU cores and allow the scheduler to see only these cores.

2. Set the **reserved_host_memory_mb** option in the same file to the amount of RAM to reserve for host processes. If you want to reserve 512 MB, use:

```
reserved_host_memory_mb=512
```

3. Restart the Compute service on the Compute nodes by running the following command:

```
systemctl restart openstack-nova-compute.service
```

4. Ensure that host processes do not run on the CPU cores reserved for guest processes by adding the **isolcpus** argument to the system's boot configuration. Use the list of CPU cores reserved for guest processes as a parameter of this argument. Using the topology from the example above, you would run the following command:

```
grubby --update-kernel=ALL --args="isolcpus=2,3,6,7"
```



Note

The **cpuset** option along with the **isolcpus** kernel argument will ensure that the underlying compute node will not be able to use the corresponding pCPUs for itself. The pCPUs will be dedicated to instances.

5. Update the boot record for this change to take effect:

```
grub2-install /dev/device
```

Replace *device* with the name of the device that contains the boot record, usually *sda*.

6. Reboot the system.

7.2. SCHEDULER CONFIGURATION

1. Edit the **/etc/nova/nova.conf** file on each system running the OpenStack Compute Scheduler. Find the **scheduler_default_filters** option, uncomment it if commented out, and add **AggregateInstanceExtraSpecFilter** and **NUMATopologyFilter** to the list of filters. The whole line can look like this:

```
scheduler_default_filters=RetryFilter,AvailabilityZoneFilter,RamFilter,
ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,CoreFilter,
NUMATopologyFilter,AggregateInstanceExtraSpecsFilter
```

2. Restart the `openstack-nova-scheduler` service:

```
systemctl restart openstack-nova-scheduler.service
```

7.3. AGGREGATE AND FLAVOR CONFIGURATION

Prepare your OpenStack environment for running virtual machine instances pinned to specific

resources by completing the following steps on a system with the Compute command-line interface:

1. Load the **admin** credentials:

```
source ~/keystonerc_admin
```

2. Create an aggregate for the hosts that will receive pinning requests:

```
nova aggregate-create name
```

Replace *name* with a suitable name, such as *performance* or *cpu_pinning*.

3. Enable the pinning by editing the metadata for the aggregate:

```
nova aggregate-set-metadata 1 pinned=true
```

In this command, number *1* matches the ID of the aggregate created in the previous step.

4. Create an aggregate for other hosts:

```
nova aggregate-create name
```

Replace *name* with another suitable name, such as *normal*.

5. Edit the metadata for this aggregate accordingly:

```
nova aggregate-set-metadata 2 pinned=false
```

Here, number *2* is used because it comes after *1*, which is the ID of the first aggregate.

6. Change your existing flavors' specifications to this one:

```
for i in $(nova flavor-list | cut -f 2 -d ' ' | grep -o '[0-9]*'); do nova flavor-key $i set "aggregate_instance_extra_specs:pinned="false"; done
```

7. Create a flavor for the hosts that will receive pinning requests:

```
nova flavor-create name ID RAM disk vCPUs
```

Replace *name* with an appropriate name, such as *m1.small.performance* or *pinned.small*, *ID* with the identifier for the new flavor (**6** if you have five standard flavors, or **auto** if you want **nova** to generate a UUID), *RAM* with the desired amount of RAM in MB, *disk* with the desired disk size in GB, and *vCPUs* with the number of virtual CPUs that you want to reserve.

8. Set the **hw:cpu_policy** specification of this flavor to **dedicated** so as to require dedicated resources, which enables CPU pinning:

```
nova flavor-key ID set hw:cpu_policy=dedicated
```

Replace *ID* with the ID of the flavor created in the previous step.

- Set the `aggregate_instance_extra_specs:pinned` specification to `true` so as to ensure that instances based on this flavor have this specification in their aggregate metadata:

```
nova flavor-key ID set aggregate_instance_extra_specs:pinned=true
```

Again, replace *ID* with the ID of the flavor.

- Add some hosts to the the new aggregates:

```
nova aggregate-add-host ID_1 host_1
```

Replace *ID_1* with the ID of the first ("performance"/"pinning") aggregate and *host_1* with the host name of the host that you want to add to the aggregate.

```
nova aggregate-add-host ID_2 host_2
```

Replace *ID_2* with the ID of the second ("normal") aggregate and *host_2* with the host name of the host that you want to add to it.

You can now boot an instance using the new flavor:

```
nova boot --image image --flavor flavor server_name
```

Replace *image* with a saved VM image name (see `nova image-list`), *flavor* with the name of the flavor (*m1.small.performance*, *pinned.small*, or any other name that you used), and *server_name* with the name for the new server.

To verify that the new server has been placed correctly, run the following command and check for `OS-EXT-SRV-ATTR:hypervisor_hostname` in the output:

```
nova show server_name
```

APPENDIX A. IMAGE CONFIGURATION PARAMETERS

The following keys can be used with the **property** option for both the **glance image-update** and **glance image-create** commands.

```
$ glance image-update IMG-UUID --property architecture=x86_64
```



Note

Behavior set using image properties overrides behavior set using flavors. For more information, see [Manage Flavors](#).

Table A.1. Property keys

Specific to	Key	Description	Supported values
-------------	-----	-------------	------------------

Specific to	Key	Description	Supported values
All	architecture	<p>The CPU architecture that must be supported by the hypervisor. For example, x86_64, arm, or ppc64. Run uname -m to get the architecture of a machine. We strongly recommend using the architecture data vocabulary defined by the libosinfo project for this purpose.</p>	<ul style="list-style-type: none"> ✦ alpha-DEC 64-bit RISC ✦ armv7l-ARM Cortex-A7 MPCore ✦ cris-Ethernet, Token Ring, AXis-Code Reduced Instruction Set ✦ i686-Intel sixth-generation x86 (P6 micro architecture) ✦ ia64-Itanium ✦ lm32-Lattice Micro32 ✦ m68k-Motorola 68000 ✦ microblaze-Xilinx 32-bit FPGA (Big Endian) ✦ microblazeel-Xilinx 32-bit FPGA (Little Endian) ✦ mips-MIPS 32-bit RISC (Big Endian) ✦ mipsel-MIPS 32-bit RISC (Little Endian) ✦ mips64-MIPS 64-bit RISC (Big Endian) ✦ mips64el-MIPS 64-bit RISC (Little Endian) ✦ openrisc-OpenCores RISC ✦ parisc-HP Precision Architecture RISC ✦ parisc64-HP Precision Architecture 64-bit RISC ✦ ppc-PowerPC 32-bit ✦ ppc64-PowerPC 64-bit ✦ ppcemb-PowerPC (Embedded 32-bit) ✦ s390-IBM Enterprise Systems Architecture/390 ✦ s390x-S/390 64-bit ✦ sh4-SuperH SH-4 (Little Endian) ✦ sh4eb-SuperH SH-4 (Big Endian) ✦ sparc-Scalable Processor Architecture, 32-bit ✦ sparc64-Scalable Processor Architecture, 64-bit ✦ unicore32-Microprocessor Research and Development Center RISC Unicore32 ✦ x86_64-64-bit extension of IA-32 ✦ xtensa-Tensilica Xtensa configurable microprocessor core ✦ xtensaeb-Tensilica Xtensa configurable microprocessor core (Big Endian)

Specific to	Key	Description	Supported values
All	hypervisor_type	The hypervisor type.	kvm, vmware
All	instance_uuid	For snapshot images, this is the UUID of the server used to create this image.	Valid server UUID
All	kernel_id	The ID of an image stored in the Image Service that should be used as the kernel when booting an AMI-style image.	Valid image ID

Specific to	Key	Description	Supported values
All	os_distro	<p>The common name of the operating system distribution in lowercase (uses the same data vocabulary as the libosinfo project). Specify only a recognized value for this field.</p> <p>Deprecated values are listed to assist you in searching for the recognized value.</p>	<ul style="list-style-type: none"> ✦ arch-Arch Linux. Do not use archlinux or org.archlinux ✦ centos-Community Enterprise Operating System. Do not use org.centos or CentOS ✦ debian-Debian. Do not use Debian or org.debian ✦ fedora-Fedora. Do not use Fedora, org.fedora, or org.fedoraproject ✦ freebsd-FreeBSD. Do not use org.freebsd, freeBSD, or FreeBSD ✦ gentoo-Gentoo Linux. Do not use Gentoo or org.gentoo ✦ mandrake-Mandrakelinux (MandrakeSoft) distribution. Do not use mandrakelinux or MandrakeLinux ✦ mandriva-Mandriva Linux. Do not use mandrivalinux ✦ mes-Mandriva Enterprise Server. Do not use mandrivaent or mandrivaES ✦ msdos-Microsoft Disc Operating System. Do not use ms-dos ✦ netbsd-NetBSD. Do not use NetBSD or org.netbsd ✦ netware-Novell NetWare. Do not use novell or NetWare ✦ openbsd-OpenBSD. Do not use OpenBSD or org.openbsd ✦ opensolaris-OpenSolaris. Do not use OpenSolaris or org.opensolaris ✦ opensuse-openSUSE. Do not use suse, SuSE, or org.opensuse ✦ rhel-Red Hat Enterprise Linux. Do not use redhat, RedHat, or com.redhat ✦ sled-SUSE Linux Enterprise Desktop. Do not use com.suse ✦ ubuntu-Ubuntu. Do not use Ubuntu, com.ubuntu, org.ubuntu, or canonical ✦ windows-Microsoft Windows. Do not use com.microsoft.server
All	os_version	The operating system version as specified by the distributor.	Version number (for example, "11.10")

Specific to	Key	Description	Supported values
All	ramdisk_id	The ID of image stored in the Image Service that should be used as the ramdisk when booting an AMI-style image.	Valid image ID
All	vm_mode	The virtual machine mode. This represents the host/guest ABI (application binary interface) used for the virtual machine.	hvm -Fully virtualized. This is the mode used by QEMU and KVM.
libvirt API driver	hw_disk_bus	Specifies the type of disk controller to attach disk devices to.	scsi, virtio, ide, or usb.
libvirt API driver	hw_numa_nodes	Number of NUMA nodes to expose to the instance (does not override flavor definition).	Integer. For a detailed example of NUMA-topology definition, refer to the hw:NUMA_def key in Add Metadata .
libvirt API driver	hw_numa_memory_policy	NUMA memory allocation policy (does not override flavor definition).	strict - Mandatory for the instance's RAM allocations to come from the NUMA nodes to which it is bound (default if numa_nodes is specified). preferred - The kernel can fall back to using an alternative node. Useful when the 'hw:numa_nodes' parameter is set to '1'.

Specific to	Key	Description	Supported values
libvirt API driver	hw_numa_cpus .0	Mapping of vCPUs N-M to NUMA node 0 (does not override flavor definition).	Comma-separated list of integers.
libvirt API driver	hw_numa_cpus .1	Mapping of vCPUs N-M to NUMA node 1 (does not override flavor definition).	Comma-separated list of integers.
libvirt API driver	hw_numa_mem .0	Mapping N GB of RAM to NUMA node 0 (does not override flavor definition).	Integer
libvirt API driver	hw_numa_mem .1	Mapping N GB of RAM to NUMA node 1 (does not override flavor definition).	Integer
libvirt API driver	hw_qemu_guest_agent	Guest agent support. If set to yes , and if qemu-ga is also installed, file systems can be quiesced (frozen) and snapshots created automatically.	yes / no

Specific to	Key	Description	Supported values
libvirt API driver	hw_rng_model	<p>Adds a random-number generator device to the image's instances. The cloud administrator can enable and control device behavior by configuring the instance's flavor. By default:</p> <ul style="list-style-type: none"> ✦ The generator device is disabled. ✦ /dev/random is used as the default entropy source. To specify a physical HW RNG device, use the following option in the nova.conf file: <pre>rng_dev_path=/dev/hwrng</pre> 	virtio , or other supported device.

Specific to	Key	Description	Supported values
libvirt API driver	hw_scsi_model	Enables the use of VirtIO SCSI (virtio-scsi) to provide block device access for compute instances; by default, instances use VirtIO Block (virtio-blk). VirtIO SCSI is a para-virtualized SCSI controller device that provides improved scalability and performance, and supports advanced SCSI hardware.	virtio-scsi
libvirt API driver	hw_video_model	The video image driver used.	vga, cirrus, vmvga, xen, or qxl
libvirt API driver	hw_video_ram	Maximum RAM for the video image. Used only if a hw_video:ram_max_mb value has been set in the flavor's extra_specs and that value is higher than the value set in hw_video_ram .	Integer in MB (for example, '64')

Specific to	Key	Description	Supported values
libvirt API driver	hw_watchdog_action	Enables a virtual hardware watchdog device that carries out the specified action if the server hangs. The watchdog uses the i6300esb device (emulating a PCI Intel 6300ESB). If hw_watchdog_action is not specified, the watchdog is disabled.	<ul style="list-style-type: none"> ✦ disabled-The device is not attached. Allows the user to disable the watchdog for the image, even if it has been enabled using the image's flavor. The default value for this parameter is disabled. ✦ reset-Forcefully reset the guest. ✦ poweroff-Forcefully power off the guest. ✦ pause-Pause the guest. ✦ none-Only enable the watchdog; do nothing if the server hangs.
libvirt API driver	os_command_line	The kernel command line to be used by the libvirt driver, instead of the default. For Linux Containers (LXC), the value is used as arguments for initialization. This key is valid only for Amazon kernel, ramdisk, or machine images (aki, ari, or ami).	

Specific to	Key	Description	Supported values
libvirt API driver and VMware API driver	hw_vif_model	Specifies the model of virtual network interface device to use.	The valid options depend on the configured hypervisor. <ul style="list-style-type: none"> ✦ KVM and QEMU: e1000, ne2k_pci, pcnet, rtl8139, and virtio. ✦ VMware: e1000, e1000e, VirtualE1000, VirtualE1000e, VirtualPCNet32, VirtualSriovEthernetCard, and VirtualVmxnet. ✦ Xen: e1000, netfront, ne2k_pci, pcnet, and rtl8139.
VMware API driver	vmware_adaptype	The virtual SCSI or IDE controller used by the hypervisor.	lsiLogic , busLogic , or ide
VMware API driver	vmware_ostype	A VMware GuestID which describes the operating system installed in the image. This value is passed to the hypervisor when creating a virtual machine. If not specified, the key defaults to otherGuest .	See thinkvirt.com .
VMware API driver	vmware_image_version	Currently unused.	1

Specific to	Key	Description	Supported values
XenAPI driver	auto_disk_config	If true, the root partition on the disk is automatically resized before the instance boots. This value is only taken into account by the Compute service when using a Xen-based hypervisor with the XenAPI driver. The Compute service will only attempt to resize if there is a single partition on the image, and only if the partition is in ext3 or ext4 format.	true / false

Specific to	Key	Description	Supported values
XenAPI driver	os_type	The operating system installed on the image. The XenAPI driver contains logic that takes different actions depending on the value of the os_type parameter of the image. For example, for os_type=windows images, it creates a FAT32-based swap partition instead of a Linux swap partition, and it limits the injected host name to less than 16 characters.	linux or windows