Red Hat OpenStack Platform 11

Instances and Images Guide

Managing Instances and Images

Last Updated: 2018-02-19

Orca Security Ltd.
Exhibit 2025
Wiz v. Orca
IPR2024-01109, -01190, -01191
Ex. 2025-001

Red Hat OpenStack Platform 11 Instances and Images Guide

Managing Instances and Images

OpenStack Team
rhos-docs@redhat.com

Ex. 2025-003

Legal Notice

Copyright © 2018 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 of a Red Hat OpenStack Platform environment.

Table of Contents

PREFACE **4**

CHAPTER 1. IMAGE SERVICE **5**
1.1. UNDERSTANDING THE IMAGE SERVICE 5
1.2. MANAGE IMAGES 7
1.2.1. Create an Image 7
1.2.1.1. Use a KVM Guest Image With Red Hat OpenStack Platform 7
1.2.1.2. Create Custom Red Hat Enterprise Linux or Windows Images 8
1.2.1.2.1. Create a Red Hat Enterprise Linux 7 Image 8
1.2.1.2.2. Create a Red Hat Enterprise Linux 6 Image 13
1.2.1.2.3. Create a Windows Image 18
1.2.1.3. Use libosinfo 19
1.2.2. Upload an Image 19
1.2.3. Update an Image 21
1.2.4. Delete an Image 21
1.2.4.1. Purge Deleted Image Data 21

CHAPTER 2. CONFIGURE OPENSTACK COMPUTE STORAGE **22**
2.1. ARCHITECTURE OVERVIEW 22
2.2. CONFIGURATION 23

CHAPTER 3. VIRTUAL MACHINE INSTANCES **26**
3.1. MANAGE INSTANCES 26
3.1.1. Add Components 26
3.1.2. Create an Instance 26
3.1.3. Update an Instance (Actions menu) 28
3.1.4. Resize an Instance 30
3.1.5. Connect to an Instance 30
3.1.5.1. Access an Instance Console using the Dashboard 30
3.1.5.2. Directly Connect to a VNC Console 31
3.1.5.3. Directly Connect to a Serial Console 32
3.1.5.3.1. Install and Configure nova-serialproxy 32
3.1.6. View Instance Usage 33
3.1.7. Delete an Instance 34
3.1.8. Manage Multiple Instances at Once 34
3.2. MANAGE INSTANCE SECURITY 34
3.2.1. Manage Key Pairs 34
3.2.1.1. Create a Key Pair 35
3.2.1.2. Import a Key Pair 35
3.2.1.3. Delete a Key Pair 35
3.2.2. Create a Security Group 35
3.2.3. Create, Assign, and Release Floating IP Addresses 35
3.2.3.1. Allocate a Floating IP to the Project 36
3.2.3.2. Assign a Floating IP 36
3.2.3.3. Release a Floating IP 36
3.2.4. Log in to an Instance 36
3.2.5. Inject an admin Password Into an Instance 37
3.3. MANAGE FLAVORS 38
3.3.1. Update Configuration Permissions 39
3.3.2. Create a Flavor 39
3.3.3. Update General Attributes 40
3.3.4. Update Flavor Metadata 40

3.3.4.1. View Metadata 41
3.3.4.2. Add Metadata 41
3.4. MANAGE HOST AGGREGATES 46
3.4.1. Enable Host Aggregate Scheduling 46
3.4.2. View Availability Zones or Host Aggregates 47
3.4.3. Add a Host Aggregate 47
3.4.4. Update a Host Aggregate 47
3.4.5. Delete a Host Aggregate 48
3.5. SCHEDULE HOSTS 49
3.5.1. Configure Scheduling Filters 49
3.5.2. Configure Scheduling Weights 52
3.5.2.1. Configure Weight Options for Hosts 53
3.6. EVACUATE INSTANCES 54
3.6.1. Evacuate One Instance 55
3.6.2. Evacuate All Instances 55
3.6.3. Configure Shared Storage 56
3.7. MANAGE INSTANCE SNAPSHOTS 57
3.7.1. Create an Instance Snapshot 57
3.7.2. Manage a Snapshot 58
3.7.3. Rebuild an Instance to a State in a Snapshot 59
3.7.4. Consistent Snapshots 59
3.8. USE RESCUE MODE FOR INSTANCES 59
3.8.1. Preparing an Image for a Rescue Mode Instance 59
3.8.1.1. Rescue Image if Using ext4 Filesystem 60
3.8.2. Adding the Rescue Image to the OpenStack Image Service 60
3.8.3. Launching an Instance in Rescue Mode 61
3.8.4. Unrescuing an Instance 61
3.9. SET A CONFIGURATION DRIVE FOR INSTANCES 61
3.9.1. Configuration Drive Options 62
3.9.2. Use a Configuration Drive 62

CHAPTER 4. CONFIGURE CPU PINNING WITH NUMA 63
4.1. COMPUTE NODE CONFIGURATION 64
4.2. SCHEDULER CONFIGURATION 65
4.3. AGGREGATE AND FLAVOR CONFIGURATION 66

APPENDIX A. IMAGE CONFIGURATION PARAMETERS 68

Ex. 2025-006

PREFACE

Red Hat OpenStack Platform (Red Hat 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, and instances. It also mentions the procedure for configuring the storage for instances for Red Hat 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 Red Hat OpenStack Platform, see Red Hat OpenStack Platform Documentation Suite.

Ex. 2025-008

CHAPTER 1. IMAGE SERVICE

This chapter discusses the steps you can follow to manage images and storage in Red Hat 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 Red Hat 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 **`ISO`** is not normally considered 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, your account must have a valid Red Hat Enterprise Linux subscription:

- Red Hat Enterprise Linux 7 KVM Guest Image
- Red Hat Enterprise Linux 6 KVM Guest Image

You will be prompted to enter your Red Hat account credentials if you are not logged in to the Customer Portal.

1.1. UNDERSTANDING THE IMAGE SERVICE

The following notable Image service features are available:

- **Image signing and verification** - Protect image integrity and authenticity by enabling deployers to sign images and save the signatures and public key certificates as image properties.
  By taking advantage of this feature, you can:
  - sign an image using your private key and upload the image, the signature, and a reference to your public key certificate (the verification metadata); the Image service then verifies that the signature is valid

  - create an image in the Compute service, have the Compute service sign the image, and upload the image and its verification metadata; the Image service again verifies that the signature is valid
  - request a signed image in the Compute service; the Image service provides the image and its verification metadata, allowing the Compute service to validate the image before booting it

- **Image conversion** - Convert images by calling the task API while importing an image.
  As part of the import workflow, a plug-in provides the image conversion. This plug-in 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 plug-in 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 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.

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.

1.2.1. Create an Image

This section provides you with the steps to manually create OpenStack-compatible images in the QCOW2 format using Red Hat Enterprise Linux 7 ISO files, Red Hat Enterprise Linux 6 ISO files, or Windows ISO files.

1.2.1.1. Use a KVM Guest Image With Red Hat OpenStack Platform

You can use a ready RHEL KVM guest QCOW2 image:

- RHEL 7.2 KVM Guest Image
- RHEL 6.8 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.

Ready Windows KVM guest QCOW2 images are not available.

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 or Windows images, see Create a Red Hat Enterprise Linux 7 Image, Create a Red Hat Enterprise Linux 6 Image, or Create a Windows Image.

1.2.1.2. Create Custom Red Hat Enterprise Linux or Windows 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 or 6 ISO file (see RHEL 7.2 Binary DVD or RHEL 6.8 Binary DVD) or a Windows ISO file. If you do not have a Windows ISO file, visit the Microsoft TechNet Evaluation Center and download an evaluation image.
- Text editor, if you want to change the **`kickstart`** files (RHEL only).

NOTE

In the following procedures, all commands with the **`[root@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 the QCOW2 format using a Red Hat Enterprise Linux 7 ISO file.

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

```
[root@host]# qemu-img create -f qcow2 rhel7.qcow2 8G
[root@host]# virt-install --virt-type kvm --name rhel7 --ram 2048 \
--cdrom /tmp/rhel-server-7.2-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.

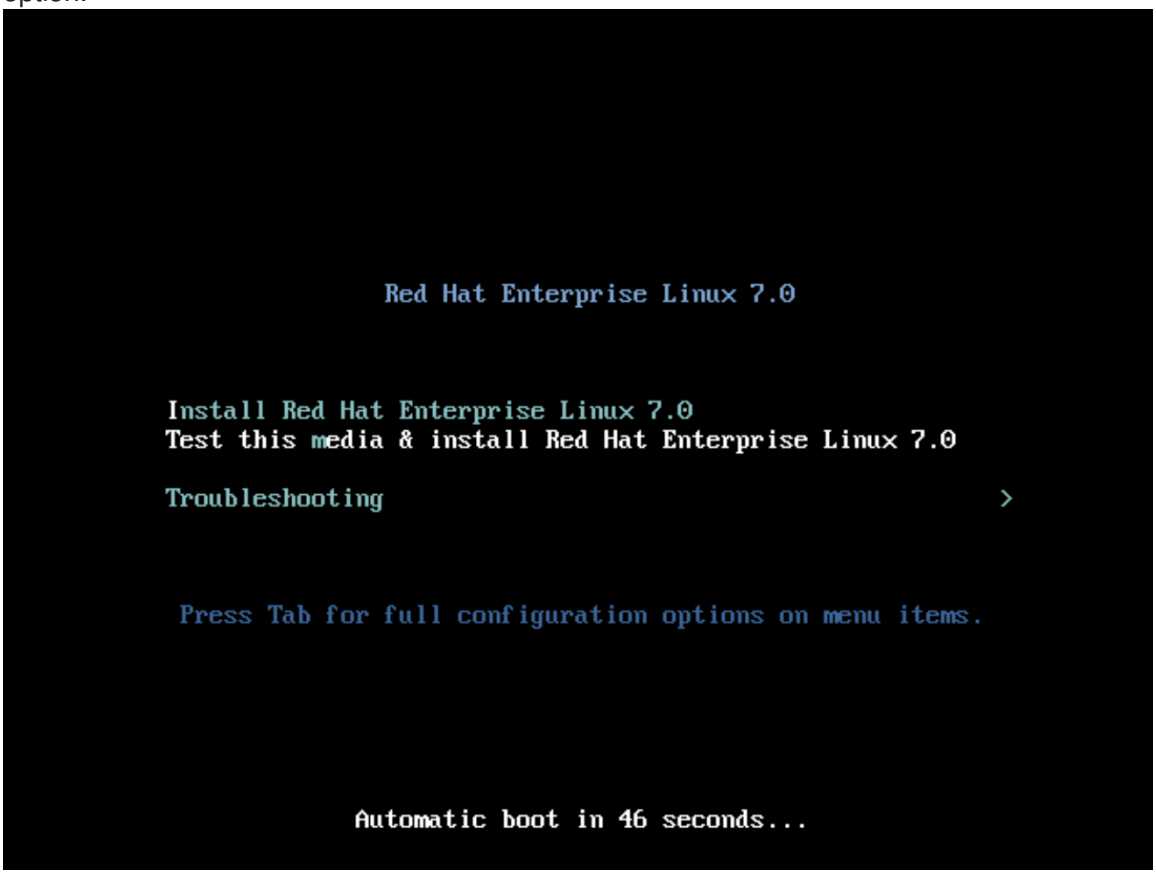
   NOTE

   If the instance does not launch automatically, run the **`virt-viewer`** command to view the console:

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

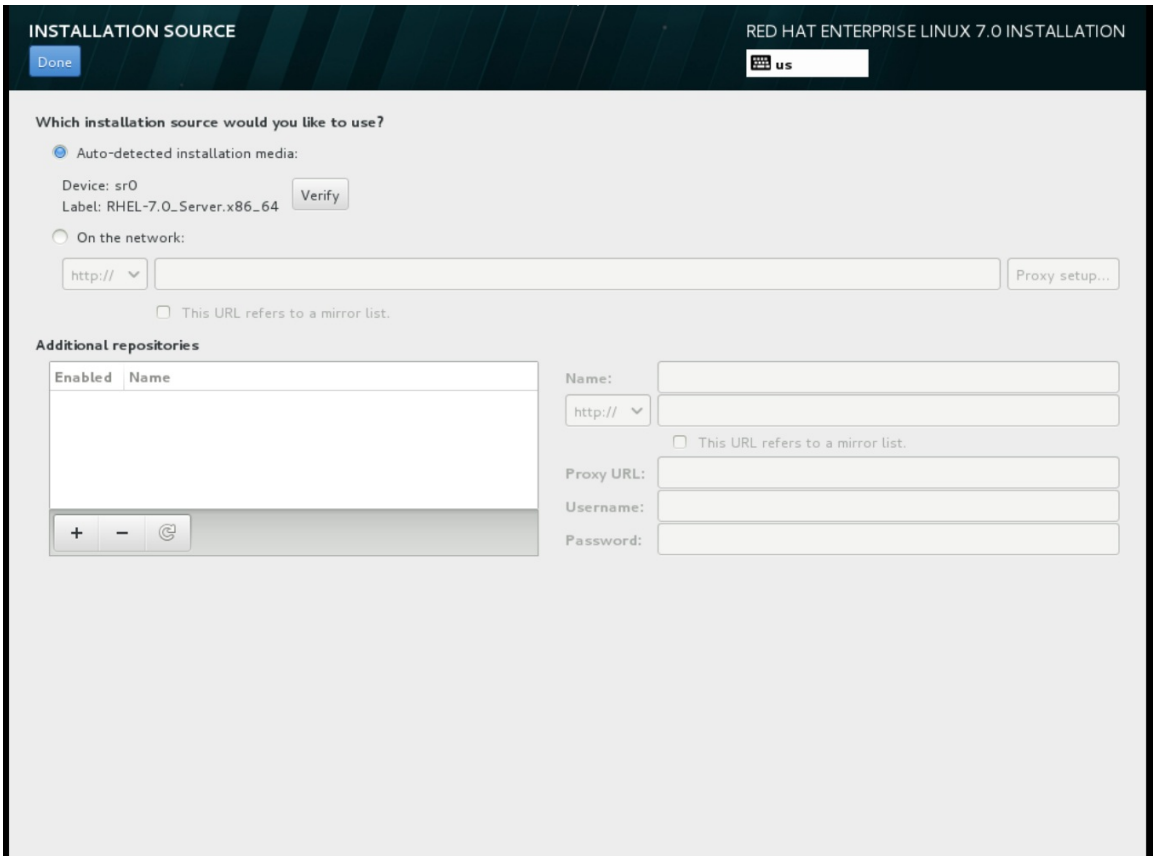
2. Set up the virtual machine as follows:

a. At the initial Installer boot menu, choose the **`Install Red Hat Enterprise Linux 7`**.*X* option.

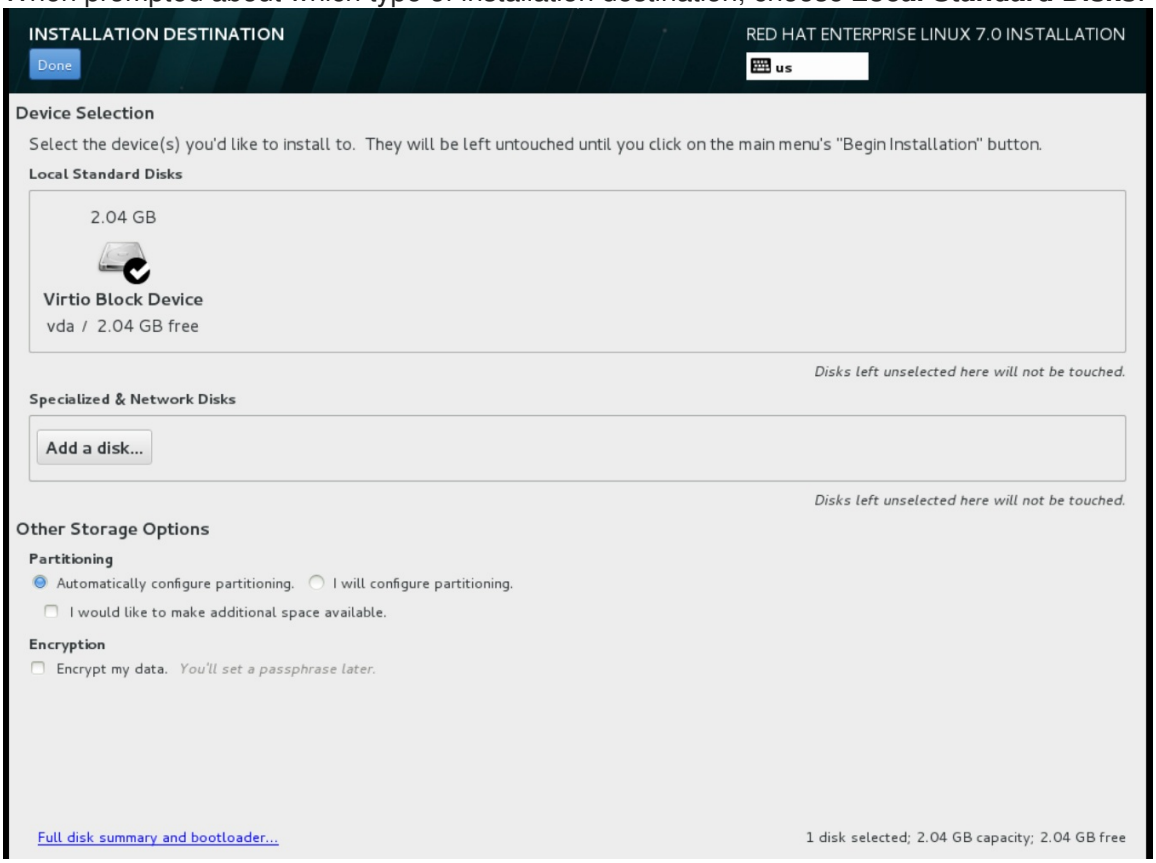


b. Choose the appropriate **Language** and **Keyboard** options.

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**.

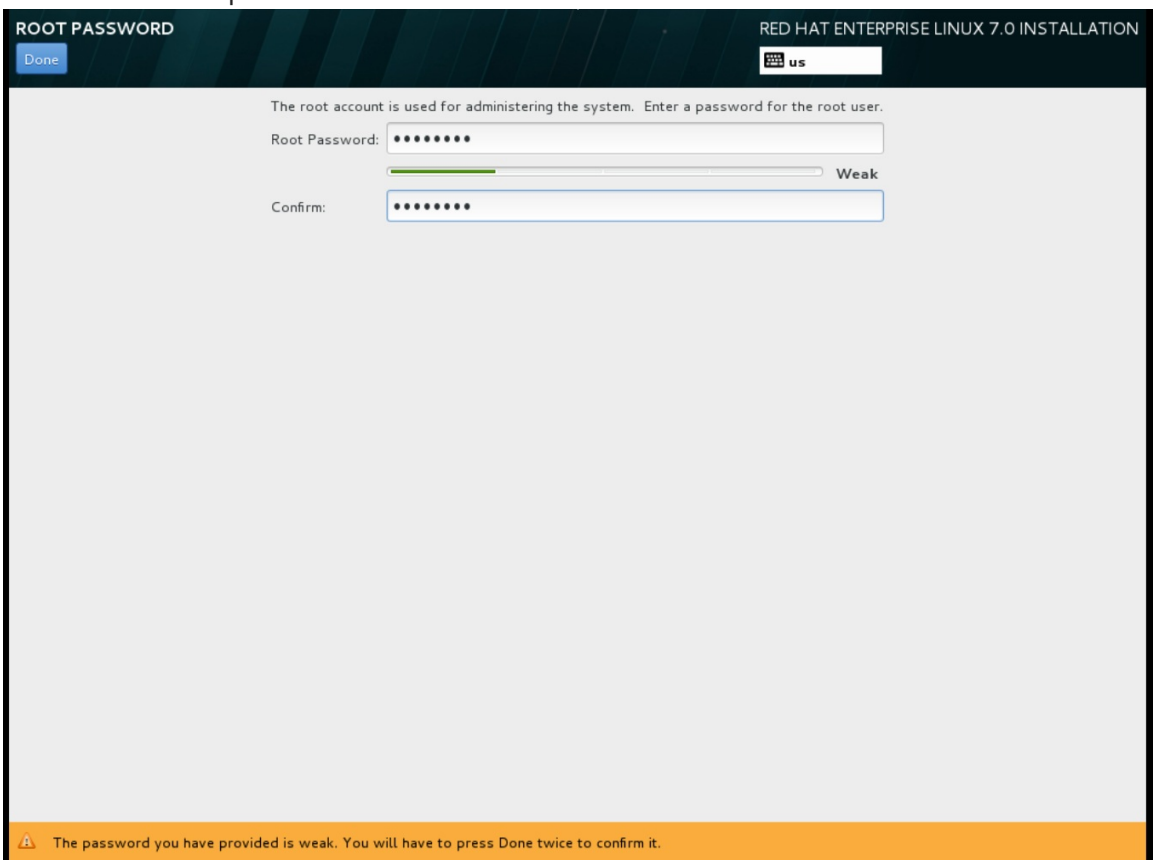


For other storage options, choose **Automatically configure partitioning**.

e. For software selection, choose **Minimal Install**.

f. For network and host name, choose **`eth0`** for network and choose a **`hostname`** for your device. The default host name 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. For details, see Subscribe to the Required Channels in *Manual Installation Procedures*.

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 **grub2-mkconfig** 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 used to create instances without issues:

```
[root@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:

```
[root@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 the QCOW2 format using a Red Hat Enterprise Linux 6 ISO file.

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

```
[root@host]# qemu-img create -f qcow2 rhel6.qcow2 4G
[root@host]# virt-install --connect=qemu:///system --
network=bridge:virbr0 \
--name=rhel6 --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-6.8-x86_64-dvd.iso
```

   This launches an instance and starts the installation process.

   NOTE

   If the instance does not launch automatically, run the **virt-viewer** command to view the console:

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

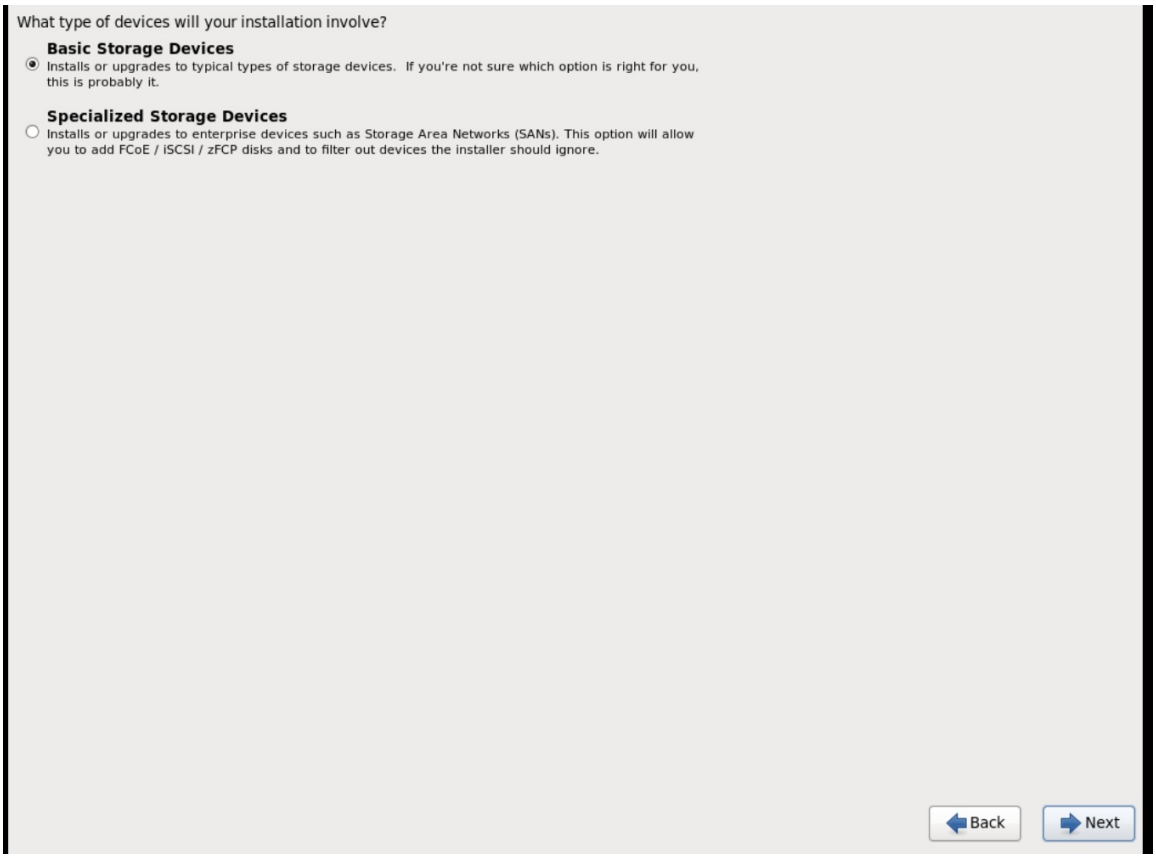
2. Set up the virtual machines as follows:

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 installer checks for the disc and lets you decide whether you want to test your installation media before installation. Select **OK** to run the test or **Skip** to proceed without testing.

b. Choose the appropriate **Language** and **Keyboard** options.

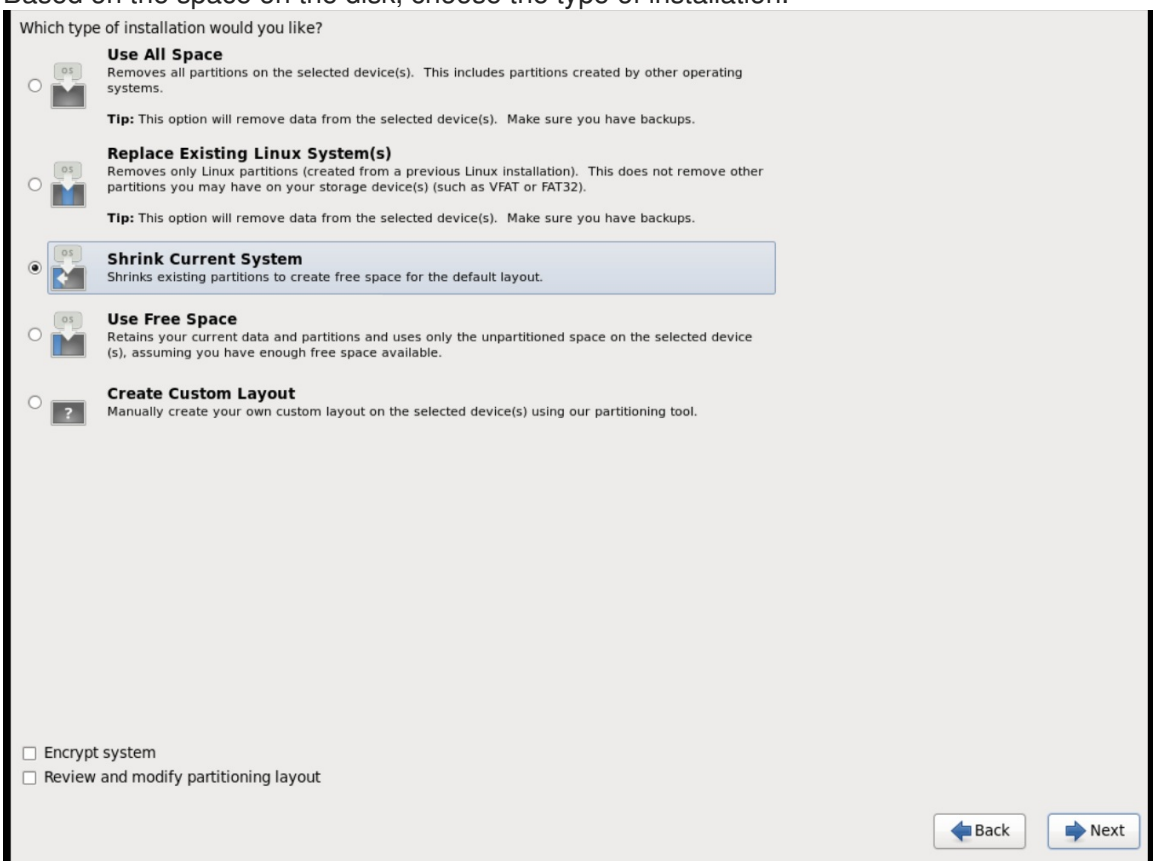
c. When prompted about which type of devices your installation uses, choose **Basic Storage Devices**.



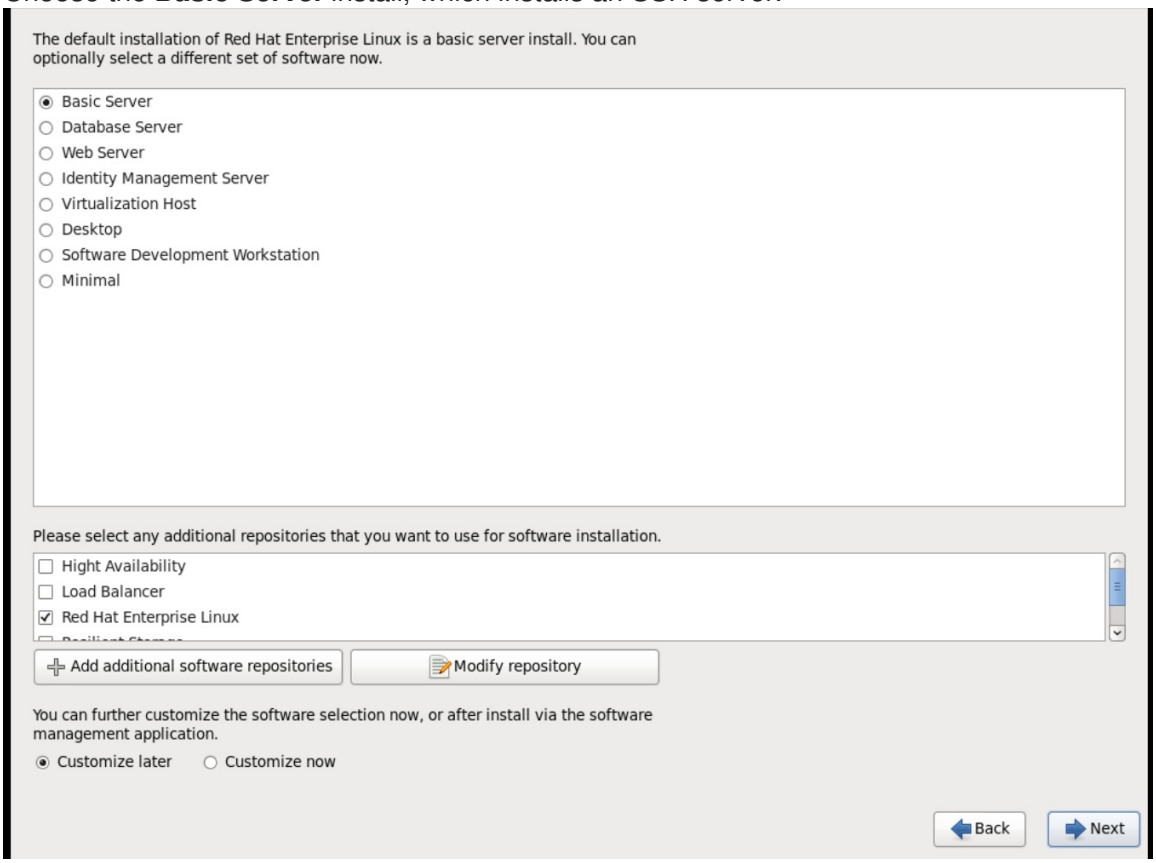
d. Choose a **`hostname`** for your device. The default host name is **`localhost.localdomain`**.

e. Set **timezone** and **`root`** password.

f. Based on the space on the disk, choose the type of installation.



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



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. For details, see Subscribe to the Required Channels in the *Manual Installation Procedures* guide.

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 the **/etc/udev/rules.d/75-persistent-net-generator.rules** file as follows:

```
# 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 used to create instances without issues:

```
[root@host]# virt-sysprep -d rhel6
```

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:

```
[root@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 **rhel6-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

1.2.1.2.3. Create a Windows Image

This section provides you with the steps to manually create an OpenStack-compatible image in the QCOW2 format using a Windows ISO file.

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

```
[root@host]# virt-install --name=name \
--disk size=size \
--cdrom=path \
--os-type=windows \
--network=bridge:virbr0 \
--graphics spice \
--ram=RAM
```

Replace the values of the **virt-install** parameters as follows:

- *name* — the name that the Windows guest should have.
- *size* — disk size in GB.
- *path* — the path to the Windows installation ISO file.
- *RAM* — the requested amount of RAM in MB.

NOTE

The **--os-type=windows** parameter ensures that the clock is set up correctly for the Windows guest, and enables its Hyper-V enlightenment features.

Note that **virt-install** saves the guest image as **/var/lib/libvirt/images/*name*.qcow2** by default. If you want to keep the guest image elsewhere, change the parameter of the **--disk** option as follows:

```
--disk path=filename,size=size
```

Replace *filename* with the name of the file which should store the guest image (and optionally its path); for example **path=win8.qcow2,size=8** creates an 8 GB file named **win8.qcow2** in the current working directory.

TIP

If the guest does not launch automatically, run the **virt-viewer** command to view the console:

```
[root@host]# virt-viewer name
```

2. Installation of Windows systems is beyond the scope of this document. For instructions on how to install Windows, see the relevant Microsoft documentation.

3. To allow the newly installed Windows system to use the virtualized hardware, you may need to install *virtio drivers* in it. To so do, first install the **virtio-win** package on the host system. This package contains the virtio ISO image, which is to be attached as a CD-ROM drive to the Windows guest. See Chapter 8. KVM Para-virtualized (virtio) Drivers in the *Virtualization Deployment and Administration Guide* for detailed instructions on how to install the **virtio-win** package, add the virtio ISO image to the guest, and install the virtio drivers.

4. To complete the setup, download and execute Cloudbase-Init on the Windows system. At the end of the installation of Cloudbase-Init, select the **Run Sysprep** and **Shutdown** check boxes. The **Sysprep** tool makes the guest unique by generating an OS ID, which is used by certain Microsoft services.

IMPORTANT

Red Hat does not provide technical support for Cloudbase-Init. If you encounter an issue, contact Cloudbase Solutions.

When the Windows system shuts down, the *name*.**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 or the command line, see Upload an Image.

1.2.1.3. Use libosinfo

Image Service (glance) can process *libosinfo* data for images, making it easier to configure the optimal virtual hardware for an instance. This can be done by adding the libosinfo-formatted operating system name to the glance image.

1. This example specifies that the image with ID **654dbfd5-5c01-411f-8599-a27bd344d79b** uses the libosinfo value of **rhel7.2**:

```
$ openstack image set 654dbfd5-5c01-411f-8599-a27bd344d79b --
property os_name=rhel7.2
```

As a result, Compute will supply virtual hardware optimized for **rhel7.2** whenever an instance is built using the **654dbfd5-5c01-411f-8599-a27bd344d79b** image.

NOTE

For a complete list of **libosinfo** values, refer to the libosinfo project: https://gitlab.com/libosinfo/osinfo-db/tree/master/data/os

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.

Table 1.1. Image Options

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	• 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).
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.

When the image has been successfully uploaded, its status is changed to **`active`**, which indicates that the image is available for use. Note that the Image service can handle even large images that take a long time to upload — longer than the lifetime of the Identity service token which was used when the upload was initiated. This is due to the fact that the Image service first creates a trust with the Identity service so that a new token can be obtained and used when the upload is complete and the status of the image is to be updated.

NOTE

You can also use the **`glance image-create`** command with the **`property`** option to upload 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**.

1.2.4.1. Purge Deleted Image Data

Information about an image is kept in the OpenStack Image Service database even after the image has been deleted. As a consequence, the database can grow in size over time and become slow to use and hard to upgrade.

To remove information about deleted images, use the **`glance-manage db purge`** command. By default, database entries are removed if they reference images deleted 30 days ago (or more), but you can specify a different amount of time (in days) as a parameter. For example, to remove information about images deleted more than 60 days ago, run the **`glance-manage`** command:

```
$ glance-manage db purge 60
```

CHAPTER 2. CONFIGURE OPENSTACK COMPUTE STORAGE

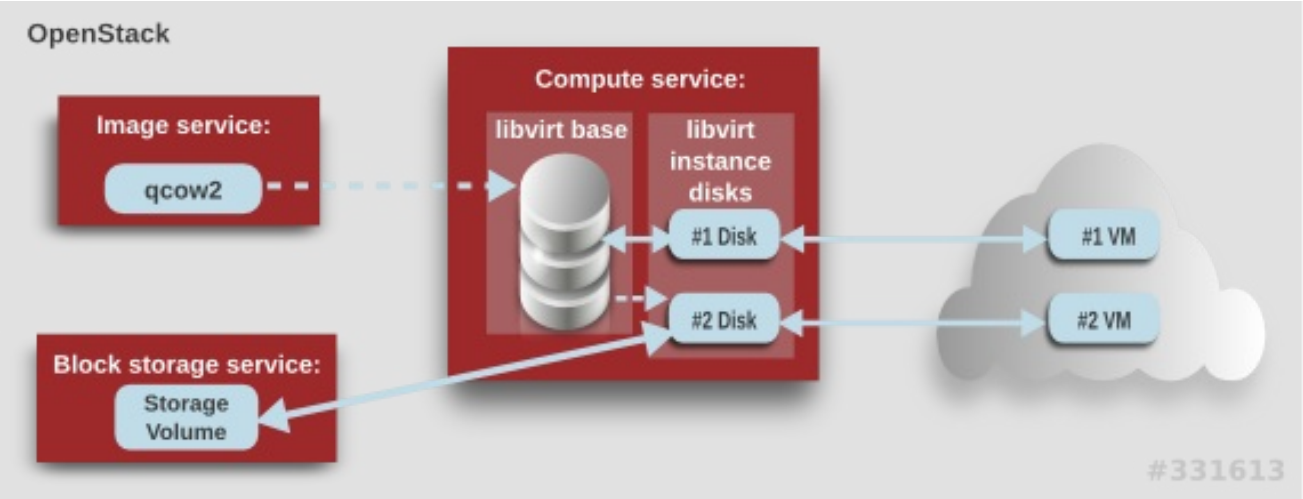
This chapter describes the architecture for the back-end storage of images in OpenStack Compute (nova), and provides basic configuration options.

2.1. ARCHITECTURE OVERVIEW

In Red Hat 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.

Ex. 2025-026

Block storage volume is persistant 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`**	Whether to convert a **`non-raw`** cached base image to be **`raw`** (boolean). If a non-raw image is converted to raw, Compute: • Disallows backing files (which might be a security issue). • Removes existing compression (to avoid CPU bottlenecks). 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. Raw base images are always used with **`libvirt_images_type=lvm`**.	**`true`**
[DEFAULT]	**`use_cow_images`**	Whether to use CoW (Copy on Write) images for **`libvirt`** instance disks (boolean): • **`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**	Preallocation mode for **libvirt** instance disks. Value can be: • **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. 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).	none
[DEFAULT]	**resize_fs_using_block_device**	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). Because this parameter enables the direct mounting of images which might otherwise be disabled for security reasons, it is not enabled by default.	**false**
[DEFAULT]	**default_ephemeral_format**	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.	**ext4**
[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**

Section	Parameter	Description	Default
[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.

Red Hat 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 Create a Network section in the *Networking Guide*).
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 Create a Volume in the *Storage Guide*).
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*).

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.

Ex. 2025-030

Table 3.1. Instance Options

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: • 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 Basic Volume Usage and Configuration in the *Storage Guide*).
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.
	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*).
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.

Tab	Field	Notes
Post-Creation	Customization Script Source	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 host name or a user password). If 'Direct Input' is selected, write your commands in the Script Data field; otherwise, specify your script file. **NOTE** 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).
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:

Table 3.2. Update Instance Options

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.

Action	Description
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*).
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").
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 *power* 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.

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.

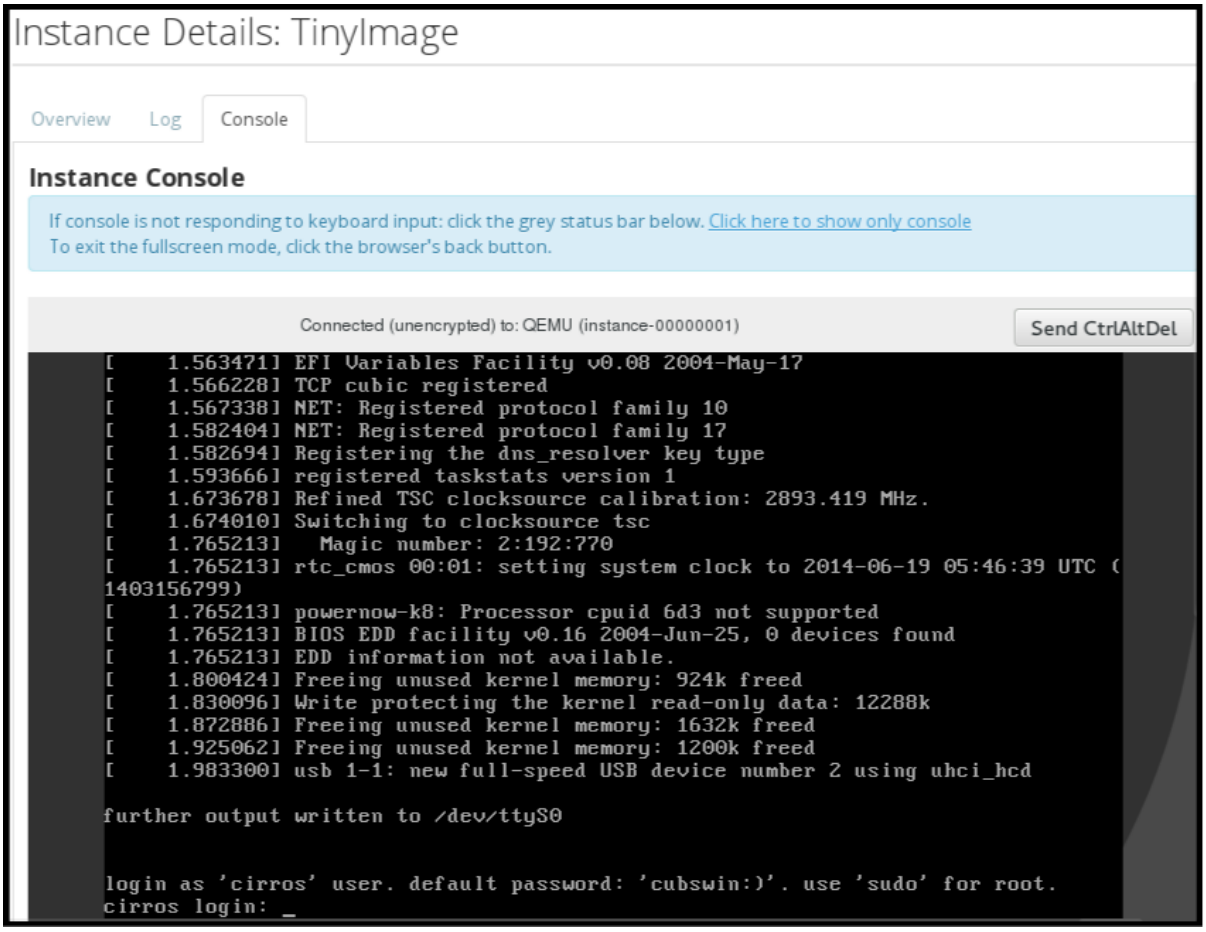
3.1.5.1. Access an Instance Console using the Dashboard

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

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

Ex. 2025-034

2. Click the instance's **More** button and select **Console**.



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
proxyclient_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 Delete a Volume in the *Storage Guide*).

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*.

3.2.1. Manage Key Pairs

Ex. 2025-038

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 group 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*.

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 address 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*.

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*).
- Ensure the instance has a floating IP address (external address) assigned to it (see Section 3.2.3, "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
```

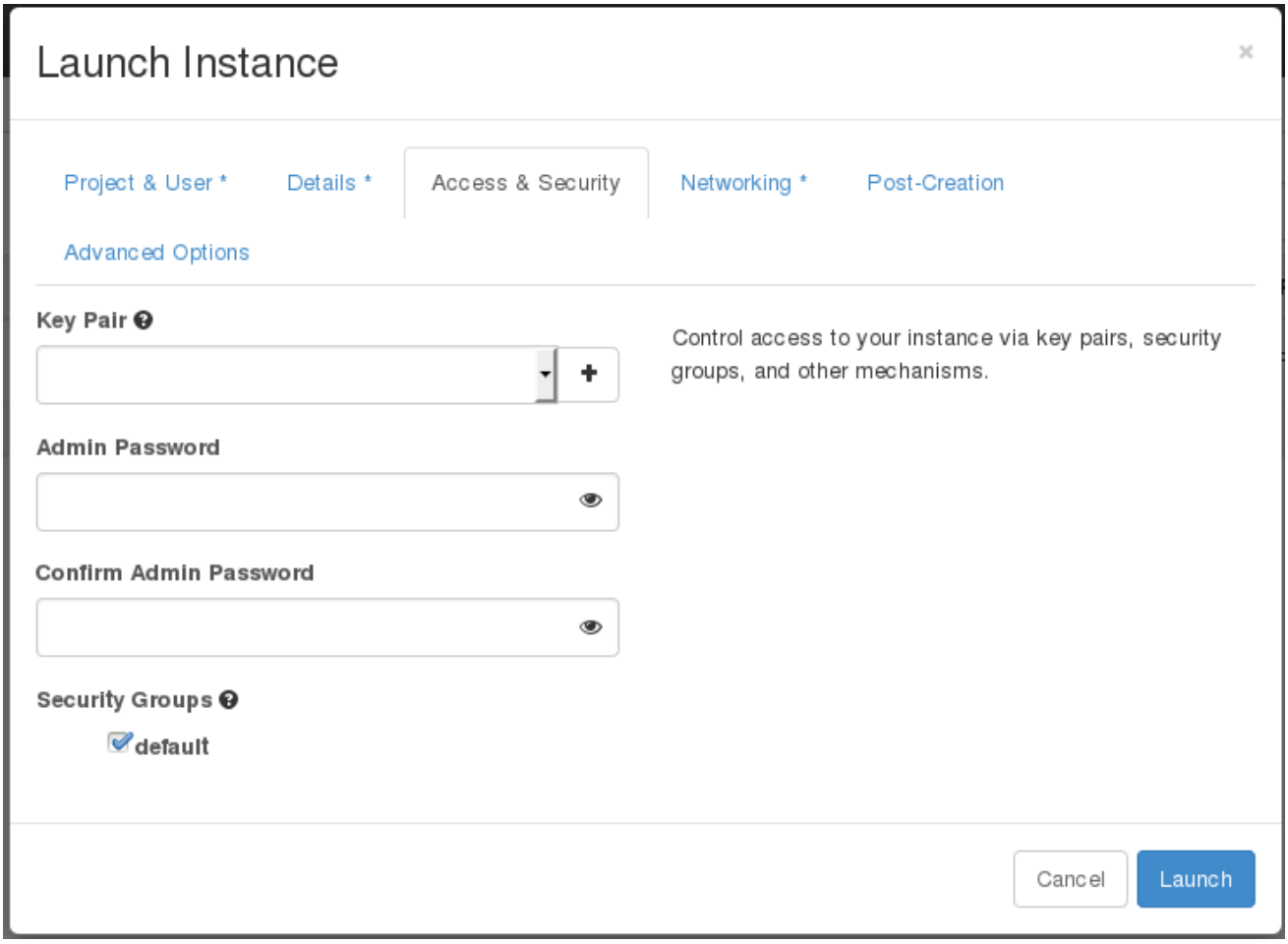
When you use the **nova boot** command to launch a new instance, the output of the command displays an **adminPass** parameter. You can use this password to log into the instance as the **root** user.

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 Red Hat 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:



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 3.3. 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
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:

Table 3.4. Flavor Options

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.

Tab	Field	Description
	VCPUs	Number of virtual CPUs.
	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**.
	Epehemeral 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 3.5, "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”).

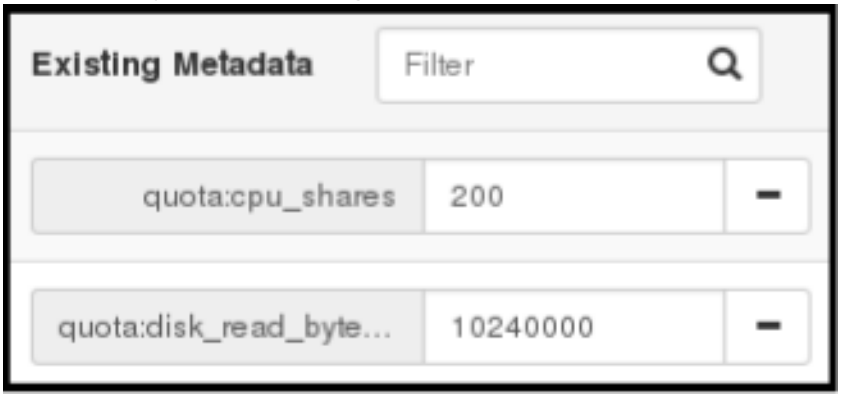
3.3.4.1. View Metadata

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 3.5, “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 3.5. Libvirt Metadata

Key	Description

Key	Description
`hw:action`	Action that configures support limits per instance. Valid actions are: • **`cpu_max_sockets`** - Maximum supported CPU sockets. • **`cpu_max_cores`** - Maximum supported CPU cores. • **`cpu_max_threads`** - Maximum supported CPU threads. • **`cpu_sockets`** - Preferred number of CPU sockets. • **`cpu_cores`** - Preferred number of CPU cores. • **`cpu_threads`** - Preferred number of CPU threads. • **`serial_port_count`** - Maximum serial ports per instance. Example: **`hw:cpu_max_sockets=2`**

Key	Description

`hw:NUMA_def`	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:
	• **`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:
	  ◦ **`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 asymetrical allocation of CPUs and RAM (important for some NFV workloads).
	
	NOTE
	If the values of **`numa_cpu`** or **`numa_mem.N`** specify more than that available, an exception is raised.
	
	Example when the instance has 8 vCPUs and 4GB RAM:
	• **`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`**
	
	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.

Key	Description
hw:watchdog_action	An instance watchdog device can be used to trigger an action if the instance somehow fails (or hangs). Valid actions are: • **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. Example: **hw:watchdog_action=poweroff**
hw_rng:action	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 Red Hat OpenStack Platform documentation). If the device has been added, valid actions are: • **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). Example: **hw_rng:allowed=True**.
hw_video:ram_max_mb	Maximum permitted RAM to be allowed for video devices (in MB). Example: **hw:ram_max_mb=64**
quota:option	Enforcing limit for the instance. Valid options are: • **cpu_period** - Time period for enforcing **cpu_quota** (in microseconds). Within the specified **cpu_period**, each vCPU cannot consume more than **cpu_quota** of runtime. The value must be in range [1000, 1000000]; '0' means 'no value'. • **cpu_quota - Maximum allowed bandwidth (in microseconds) for the vCPU in each `cpu_period**. The value must be in range [1000, 18446744073709551]. '0' means 'no value'; a negative value means that the vCPU is not controlled. **cpu_quota** and **cpu_period** can be used to ensure that all vCPUs run at the same speed. • **cpu_shares** - 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'. • **disk_read_bytes_sec** - Maximum disk reads in bytes per second.

Key	Description
	• **`disk_read_iops_sec`** - Maximum read I/O operations per second. • **`disk_write_bytes_sec`** - Maximum disk writes in bytes per second. • **`disk_write_iops_sec`** - Maximum write I/O operations per second. • **`disk_total_bytes_sec`** - Maximum total throughput limit in bytes per second. • **`disk_total_iops_sec`** - Maximum total I/O operations per second. • **`vif_inbound_average`** - Desired average of incoming traffic. • **`vif_inbound_burst`** - Maximum amount of traffic that can be received at **`vif_inbound_peak`** speed. • **`vif_inbound_peak`** - Maximum rate at which incoming traffic can be received. • **`vif_outbound_average`** - Desired average of outgoing traffic. • **`vif_outbound_burst`** - Maximum amount of traffic that can be sent at **`vif_outbound_peak`** speed. • **`vif_outbound_peak`** - Maximum rate at which outgoing traffic can be sent. Example: **`quota:vif_inbound_average=10240`** In addition, the VMware driver supports the following quota options, which control upper and lower limits for CPUs, RAM, disks, and networks, as well as *shares*, which can be used to control relative allocation of available resources among tenants: • **`cpu_limit`** - Maximum CPU frequency available to a virtual machine (in MHz). • **`cpu_reservation`** - Guaranteed minimum amount of CPU resources available to a virtual machine (in MHz). • **`cpu_shares_level`** - CPU allocation level (shares) in the case of contention. Possible values are **`high`**, **`normal`**, **`low`**, and **`custom`**. • **`cpu_shares_share`** - The number of allocated CPU shares. Applicable when **`cpu_shares_level`** is set to **`custom`**. • **`memory_limit`** - Maximum amount of RAM available to a virtual machine (in MB). • **`memory_reservation`** - Guaranteed minimum amount of RAM available to a virtual machine (in MB). • **`memory_shares_level`** - RAM allocation level (shares) in the case of contention. Possible values are **`high`**, **`normal`**, **`low`**, and **`custom`**. • **`memory_shares_share`** - The number of allocated RAM shares. Applicable when **`memory_shares_level`** is set to **`custom`**.

Key	Description
	• **`disk_io_limit`** - Maximum I/O utilization by a virtual machine (in I/O operations per second). • **`disk_io_reservation`** - Guaranteed minimum amount of disk resources available to a virtual machine (in I/O operations per second). • **`disk_io_shares_level`** - I/O allocation level (shares) in the case of contention. Possible values are **`high`**, **`normal`**, **`low`**, and **`custom`**. • **`disk_io_shares_share`** - The number of allocated I/O shares. Applicable when **`disk_io_shares_level`** is set to **`custom`**. • **`vif_limit`** - Maximum network bandwidth available to a virtual network adapter (in Mbps). • **`vif_reservation`** - Guaranteed minimum network bandwidth available to a virtual network adapter (in Mbps). • **`vif_shares_level`** - Network bandwidth allocation level (shares) in the case of contention. Possible values are **`high`**, **`normal`**, **`low`**, and **`custom`**. • **`vif_shares_share`** - The number of allocated network bandwidth shares. Applicable when **`vif_shares_level`** is set to **`custom`**.

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,Retry
     Filter,RamFilter,ComputeFilter,ComputeCapabilitiesFilter,ImagePro
     pertiesFilter,CoreFilter
     ```

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

     ```
     scheduler_default_filters=AvailabilityZoneFilter,RetryFilter,RamF
     ilter,ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilt
     er,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 **Metatdata**:
   - 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 3.6, "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 3.6. 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

The Compute scheduling service determines on which 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.

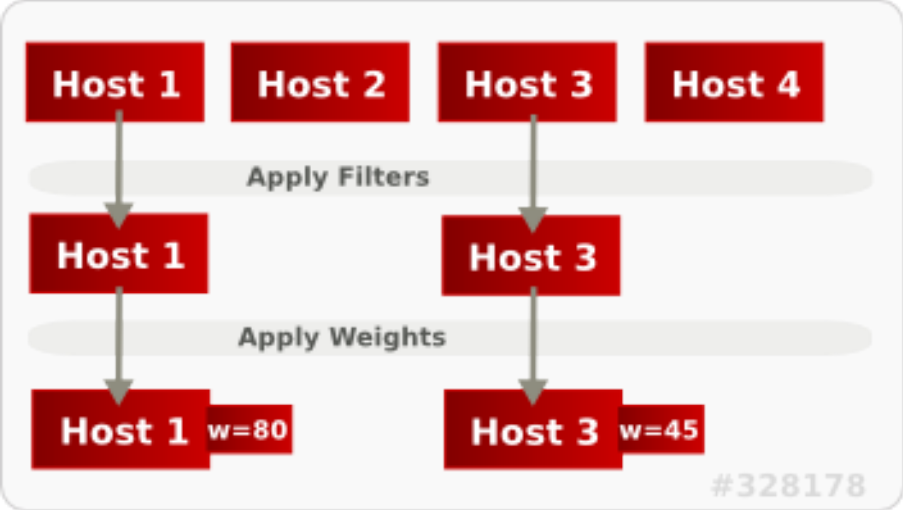
You can configure the following components:

- 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. For a list of these options, see the Configuration Reference guide.

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,Com
puteFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,ServerGroupAnti
AffinityFilter,ServerGroupAffinityFilter
```

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

- The **`nova boot`** command, see Command-Line Interface Reference guide.
- 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 3.7. Scheduling Filters

Filter	Description
AggregateCoreFilter	Uses the host-aggregate metadata key cpu_allocation_ratio 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.
	If this ratio is not set, the filter uses the cpu_allocation_ratio value in the /etc/nova/nova.conf file. The default value is **`16.0`** (16 virtual CPU can be allocated per physical CPU).
AggregateDiskFilter	Uses the host-aggregate metadata key disk_allocation_ratio 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.
	If this ratio is not set, the filter uses the disk_allocation_ratio value in the /etc/nova/nova.conf file. The default value is **`1.0`** (one virtual disk can be allocated for each physical disk).
AggregateImagePropertiesIsolation	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".
AggregateInstanceExtraSpecsFilter	Metadata in the host aggregate must match the host's flavor metadata. For more information, see Section 3.3.4, "Update Flavor Metadata" .
AggregateMultiTenancyIsolation	A host with the specified filter_tenant_id can only contain instances from that tenant (project). **NOTE** The tenant can still place instances on other hosts.
AggregateRamFilter	Uses the host-aggregate metadata key ram_allocation_ratio 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.

Filter	Description
	If this ratio is not set, the filter uses the ram_allocation_ratio value in the /etc/nova/nova.conf file. The default value is **`1.5`** (1.5 RAM can be allocated for each physical RAM).
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 **`:`** 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 cpu_allocation_ratio in the **`/etc/nova/nova.conf`** 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 disk_allocation_ratio in the **`/etc/nova/nova.conf`** 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 **`/etc/nova/nova.conf`** file using **`isolated_hosts`** and **`isolated_images`** (comma-separated values).
JsonFilter	Recognises and uses an instance's custom JSON filters: • Valid operators are: =, <, >, in, ⇐, >=, not, or, and • Recognised variables are: $free_ram_mb, $free_disk_mb, $total_usable_ram_mb, $vcpus_total, $vcpus_used
	The filter is specfied as a query hint in the **`nova boot`** command. For example: **`--hint query='['>=', '$free_disk_mb', 200 * 1024]'`**
MetricFilter	Filters out hosts with unavailable metrics.

Filter	Description
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 ram_allocation_ratio in the **`/etc/nova/nova.conf`** 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 scheduler_max_attempts 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	Only passes hosts for a specific server group: • Give the server group the affinity policy (nova server-group-create --policy affinity groupName). • Build the instance with that group (**`nova boot`** option **`--hint group=UUID`**)
ServerGroupAntiAffinityFilter	Only passes hosts in a server group that do not already host an instance: • 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`**).
SimpleCIDRAffinityFilter	Only passes hosts on the specified IP subnet range specified by the instance's cidr and **`build_new_host_ip`** hints. Example: **`--hint build_near_host_ip=192.0.2.0 --hint cidr=/24`**

3.5.2. Configure Scheduling Weights

Hosts can be weighted for scheduling; the host 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) + ...
```

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.affinity`** - Weighs the host's proximity to other hosts in the given server group.
- **`nova.scheduler.weights.all_weighers`** - Uses all host weighers (default).

Table 3.8. Host Weight Options

Weigher	Option	Description
All	[DEFAULT] *scheduler_host_subset_size*	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).
affinity	[default] **`soft_affinity_weight_multiplier`**	Used for weighing hosts for group soft-affinity. Should be a positive floating-point number, because a negative value results in the opposite behavior, which is normally controlled by **`soft_anti_affinity_weight_multiplier`**.
affinity	[default] **`soft_anti_affinity_weight_multiplier`**	Used for weighing hosts for group soft-anti-affinity. Should be a positive floating-point number, because a negative value results in the opposite behavior, which is normally controlled by **`soft_affinity_weight_multiplier`**.
metrics	[metrics] required	Specifies how to handle metrics in [metrics] **`weight_setting`** that are unavailable: • **`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`**.

Weigher	Option	Description
metrics	[metrics] **weight_multiplier**	Mulitplier 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: • **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 Example: **weight_setting=cpu.user.time=1.0**
ram	[DEFAULT] **ram_weight_multiplier**	Multiplier 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.

3.6. 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 How to Migrate a Static Instance). Images created using **`nova image-create`** are only usable by nova (and not glance).
- 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] instance_name [target_host]
```

Where:

- **`--password`** - Admin password to set for the evacuated instance. If a password is not specified, a random password is generated and output when evacuation is complete.
- **`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]
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
```

- **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:

Table 3.9. Snapshot Options

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

NOTE

If you intend to use an instance snapshot as a template to create new instances, you must ensure that the disk state is consistent. Before you create a snapshot, set the snapshot image metadata property **os_require_quiesce=yes**. For example,

```
$ glance image-update IMAGE_ID --property
os_require_quiesce=yes
```

For this to work, the guest should have the **qemu-guest-agent** package installed, and the image should be created with the metadata property parameter **hw_qemu_guest_agent=yes** set. For example,

```
$ glance image-create --name NAME \
--disk-format raw \
--container-format bare \
--file FILE_NAME \
--is-public True \
--property hw_qemu_guest_agent=yes \
--progress
```

If you unconditionally enable the **hw_qemu_guest_agent=yes** parameter, then you are adding another device to the guest. This consumes a PCI slot, and will limit the number of other devices you can allocate to the guest. It also causes Windows guests to display a warning message about an unknown hardware device.

For these reasons, setting the **hw_qemu_guest_agent=yes** parameter is optional, and the parameter should be used for only those images that require the QEMU guest agent.

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 in the *Storage Guide*.
   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 Section 1.2.3, “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**.

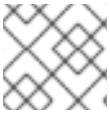
For more information on launching an instance, see Section 3.1.2, “Create an Instance”.

3.7.4. Consistent Snapshots

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

Compute’s **`libvirt`** driver 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.

3.8. USE RESCUE MODE FOR INSTANCES

Compute has a method to reboot a virtual machine in rescue mode. Rescue mode provides a mechanism for access when the virtual machine image renders the instance inaccessible. A rescue virtual machine allows a user to fix their virtual machine by accessing the instance with a new root password. This feature is useful if an instance’s filesystem is corrupted. By default, rescue mode starts an instance from the initial image attaching the current boot disk as a secondary one.

3.8.1. Preparing an Image for a Rescue Mode Instance

Due to the fact that both the boot disk and the disk for rescue mode have same UUID, sometimes the virtual machine can be booted from the boot disk instead of the disk for rescue mode.

To avoid this issue, you should create a new image as rescue image based on the procedure in Section 1.2.1, “Create an Image”:

NOTE

The **rescue** image is stored in **glance** and configured in the **nova.conf** as a default, or you can select when you do the rescue.

3.8.1.1. Rescue Image if Using *ext4* Filesystem

When the base image uses **ext4** filesystem, you can create a rescue image from it using the following procedure:

1. Change the *UUID* to a random value using the **tune2fs** command:

   ```
   # tune2fs -U random /dev/DEVICE_NODE
   ```

   Here *DEVICE_NODE* is the root device node (for example, **sda**, **vda**, and so on).

2. Verify the details of the filesystem, including the new *UUID*:

   ```
   # tune2fs -l
   ```

3. Update the **/etc/fstab** to use the new *UUID*. You may need to repeat this for any additional partitions you have, that are mounted in the **fstab** by *UUID*.

4. Update the **/boot/grub2/grub.conf** file and update the *UUID* parameter with the new *UUID* of the root disk.

5. Shut down and use this image as your rescue image. This will cause the rescue image to have a new random *UUID* that will not conflict with the instance that you are rescuing.

NOTE

The XFS filesystem cannot change the UUID of the root device on the running virtual machine. Reboot the virtual machine until the virtual machine is launched from the disk for rescue mode.

3.8.2. Adding the Rescue Image to the OpenStack Image Service

When you have completed modifying the UUID of your image, use the following commands to add the generated rescue image to the OpenStack Image service:

1. Add the rescue image to the Image service:

   ```
   # glance image-create --name IMAGE_NAME --disk-format qcow2 \
     --container-format bare --is-public True --file IMAGE_PATH
   ```

   Here *IMAGE_NAME* is the name of the image, *IMAGE_PATH* is the location of the image.

2. Use the **image-list** command to obtain the *IMAGE_ID* required for launching an instace in the rescue mode.

   ```
   # glance image-list
   ```

You can also upload an image using the OpenStack Dashboard, see Section 1.2.2, “Upload an Image”.

3.8.3. Launching an Instance in Rescue Mode

1. Since you need to rescue an instance with a specific image, rather than the default one, use the **--image** parameter:

   ```
   # nova rescue --image IMAGE_ID VIRTUAL_MACHINE_ID
   ```

   Here *IMAGE_ID* is the ID of the image you want to use and *VIRTUAL_MACHINE_ID* is ID of a virtual machine that you want to rescue.

   NOTE

   The **nova rescue** command allows an instance to perform a soft shut down. This allows the guest operating system to perform a controlled shutdown before the instance is powered off. The shut down behavior is configured by the **shutdown_timeout** parameter that can be set in the **nova.conf** file. The value stands for the overall period (in seconds) a guest operation system is allowed to complete the shutdown. The default timeout is 60 seconds.

   The timeout value can be overridden on a per image basis by means of **os_shutdown_timeout** that is an image metadata setting allowing different types of operating systems to specify how much time they need to shut down cleanly.

2. Reboot the virtual machine.

3. Confirm the status of the virtual machine is *RESCUE* on the controller node by using **nova list** command or by using dashboard.

4. Log in to the new virtual machine dashboard by using the password for rescue mode.

You can now make the necessary changes to your instance to fix any issues.

3.8.4. Unrescuing an Instance

You can **unrescue** the fixed instance to restart it from the boot disk.

1. Execute the following commands on the controller node.

   ```
   # nova unrescue VIRTUAL_MACHINE_ID
   ```

   Here *VIRTUAL_MACHINE_ID* is ID of a virtual machine that you want to unrescue.

The status of your instance returns to *ACTIVE* once the unrescue operation has completed successfully.

3.9. SET A CONFIGURATION DRIVE FOR INSTANCES

You can use the **config-drive** parameter to present a read-only drive to your instances. This drive can contain selected files that are then accessible to the instance. The configuration drive is attached to the instance at boot, and is presented to the instance as a partition. Configuration drives are useful when combined with *cloud-init* (for server bootstrapping), and when you want to pass large files to your instances.

3.9.1. Configuration Drive Options

Set the initial configuration drive options under **[DEFAULT]** in *nova.conf*:

- **config_drive_format** - sets the format of the drive, and accepts the options **iso9660** and **vfat**. By default, it uses **iso9660**.
- **force_config_drive=true** - this forces the configuration drive to be presented to all instances.
- **mkisofs_cmd=genisoimage** - specifies the command to use for ISO file creation. This value must not be changed, as only *genisoimage* is supported.

3.9.2. Use a Configuration Drive

An instance attaches its configuration drive at boot time. This is enabled by the **--config-drive** option. For example, this command creates a new instance named *test-instance01* and attaches a drive containing a file named */root/user-data.txt*:

```
# nova boot --flavor m1.tiny --config-drive true --file /root/user-
data.txt=/root/user-data.txt --image cirros test-instance01
```

Once the instance has booted, you can log in to it and see a file named */root/user-data.txt*.

NOTE

You can use the configuration drive as a source for *cloud-init* information. During the initial instance boot, *cloud-init* can automatically mount the configuration drive and run the setup scripts.

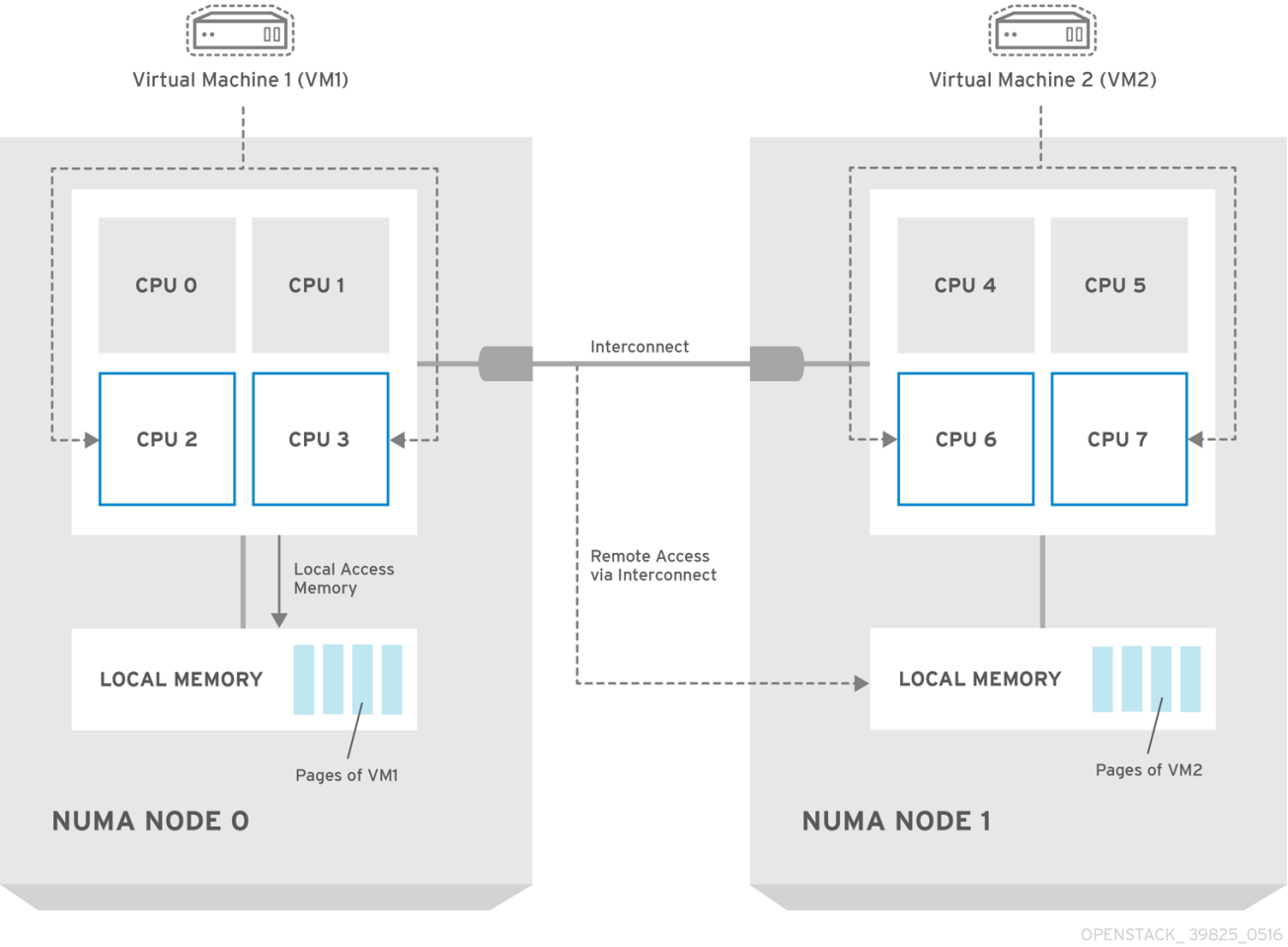
CHAPTER 4. CONFIGURE CPU PINNING WITH NUMA

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:



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".

4.1. 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 4.1. 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.

WARNING

The **isolcpus** argument is not fully functional on Red Hat Enterprise Linux 7.1. There is a bug which allows kernel processes to use CPUs that have been isolated. This bug was fixed in Red Hat Enterprise Linux 7.2, however some users have experienced performance problems.

Because of this, the **isolcpus** solution has been deprecated and there is a replacement solution which relies on **systemd**. However, this solution is still a work-in-progress, since it currently cannot isolate all of the kernel threads.

To use the systemd solution, edit the **/etc/systemd/system.conf** file and uncomment the following line:

```
CPUAffinity=1 2
```

The **CPUAffinity** option takes a list of CPU indices or ranges separated by a whitespace.

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.

4.2. 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,RamFilt
er,
ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,CoreFi
lter,
NUMATopologyFilter,AggregateInstanceExtraSpecsFilter
```

2. Restart the openstack-nova-scheduler service:

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

4.3. 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, and also the **hw:cpu_thread_policy** specification to **require**, which places each vCPU on thread siblings:

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

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

NOTE

If the host does not have an SMT architecture or enough CPU cores with free thread siblings, scheduling will fail. If such behavior is undesired, or if your hosts simply do not have an SMT architecture, do not use the **hw:cpu_thread_policy** specification, or set it to **prefer** instead of **require**. The (default) **prefer** policy ensures that thread siblings are used when available.

9. 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.

10. Add some hosts to 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 Section 3.3, “Manage Flavors”.

Table A.1. Property Keys

Specific to	Key	Description	Supported values
All	architecture	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.	• 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)

Specific to	Key	Description	Supported values
			• 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)
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	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. Deprecated values are listed to assist you in searching for the recognized value.	• 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

Specific to	Key	Description	Supported values
All	os_version	The operating system version as specified by the distributor.	Version number (for example, "11.10")
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, see the hw:NUMA_def key in Add Metadata.
libvirt API driver	hw_numa_mempolicy	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'.
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.

Specific to	Key	Description	Supported values
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	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: • 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: rng_dev_path=/dev/hwrng	**`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.	• 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).	
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. • 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.

Specific to	Key	Description	Supported values
VMware API driver	vmware_adapter type	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`**
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`**