Red Hat Enterprise Linux OpenStack Platform 7

实例和镜像指南

管理实例、镜像、卷和容器

OpenStack 团队

Red Hat Enterprise Linux OpenStack Platform 7 实例和镜像指南

管理实例、镜像、卷和容器

OpenStack 团队
rhos-docs@redhat.com

法律通告

摘要

实例和镜像管理指南介绍了在 Red Hat Enterprise Linux OpenStack Platform 环境中管理实例、镜像、卷和容器的内容。

目录

序言

Red Hat Enterprise Linux OpenStack Platform（RHEL OpenStack Platform）提供了一个在 Red Hat Enterprise Linux 上构建私有或公共 IaaS (Infrastructure-as-a-Service，基础设施即服务）云服务的平台。它为部署使用云技术的计算环境提供了扩展性和容错性。

本指南讨论了与创建和管理镜像（image）、实例（instance）、卷（volume，也被称为云硬盘）和容器（container）相关的内容。同时，它还包括了为实例配置存储，以及为 RHEL OpenStack Platform 配置 NFS 后端的信息。

您可以通过 OpenStack dashboard 或命令行客户端来对云进行管理。多数操作都可以通过这两种方法中的任一方法完成，而一些高级的操作只能通过命令行来完成。

注意

如需完整的 RHEL OpenStack Platform 文档，请访问 Red Hat Enterprise Linux OpenStack Platform Documentation

第 1 章 镜像服务

本章介绍了在 RHEL OpenStack Platform 中管理镜像和存储所需的步骤。

虚拟机镜像（image）就是一个包括安装了可引导操作系统的虚拟磁盘的文件。虚拟机镜像可以有不同的格式，以下是 RHEL OpenStack Platform 支持的格式：

- **RAW** - 没有进行结构化的磁盘镜像格式。
- **QCOW2** - QEMU 仿真程序支持的磁盘格式。
- **ISO** - 保存在一个二进制文件中的、包括磁盘每个扇区中的数据的数据备份。
- **AKI** - Amazon Kernel Image。
- **AMI** - Amazon Machine Image。
- **ARI** - Amazon RAMDisk Image。
- **VDI** - VirtualBox 虚拟机监控程序和 QEMU 仿真程序支持的磁盘格式。
- **VHD** - VMWare、VirtualBox 以及其它系统的虚拟机监控程序支持的通用磁盘格式。
- **VMDK** - 一些通用虚拟机监控程序支持的磁盘格式。

因为 ISO 包括了已安装了操作系统的可引导文件系统，所以我们通常不认为 ISO 是一个虚拟机镜像格式。但是，您可以向处理其它虚拟机镜像文件一样处理 ISO。

为了下载红帽官方的 Red Hat Enterprise Linux 云镜像，您需要一个有效的 Red Hat Enterprise Linux 订阅：

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

1.1. 镜像服务：新功能

RHEL OpenStack Platform 7 发行版本中的镜像服务包括以下新功能：

- **镜像转换** - 在导入镜像时调用任务 API 对镜像进行转换（在 Kilo 中只支持 qcow/raw 格式的转换）。

  作为镜像导入流程的一部分，一个插件提供了对镜像进行转换的功能。这个插件可以根据用户的配置被启用或被禁用。因此，用户需要指定用于进行部署的首选镜像格式。

  镜像服务（Image service）在接收到特定格式镜像的数据后会把这些数据保存在一个临时的存储中。然后，会触发转换插件来把这些数据转换为目标格式，并把转换后的数据保存在最终的存储位置中。当整个过程完成后，临时存储的镜像数据会被删除。因此，上传的原始镜像数据不会被镜像服务保留。

注意

镜像转换的过程只会在**导入（import）**一个镜像时才会被触发。在**上传（upload）**一个镜像时不会触发镜像的转换。例如：

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

- **镜像内省（Image Introspection）** - 每种格式的镜像本身都会包括一组元数据。

  例如，一个 **vmdk** 包括以下参数：

```
$ 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"
```

  通过对 *vmdk* 进行内省，您可以方便地知道 *disk_type* 是 *streamOptimized*，*adapter_type* 是 *buslogic*。通过在镜像服务中对这些元数据进行提取，管理员不再需要了解所有元数据（除非需要覆盖其中的一些数据）。这些元数据参数对镜像的使用者非常有用。例如，在 Compute 节点中，对一个 *streamOptimized* 磁盘进行实例化的过程和对 *flat* 磁盘进行实例化的过程完全不同。现在，您可以在导入镜像的过程中，通过调用任务 API 进行镜像内省。

注意

在 Kilo 中，您只能内省 *virtual_size* 元数据参数。

1.2. 管理镜像

OpenStack Image 服务 (glance) 提供了磁盘和服务器镜像的发现（discovery）、注册（registration）和分发（delivery）功能。您可以对服务器镜像进行复制或进行快照，然后进行保存。存储的镜像可以作为模板来快速设置并运行一个新服务器。

1.2.1. 创建镜像

1.2.1. 创建镜像

本节介绍了使用 Red Hat Enterprise Linux 6 和 Red Hat Enterprise Linux 7 ISO 文件手工创建 OpenStack 兼容的 .qcow2 格式镜像。

1.2.1.1. 在 RHEL OpenStack Platform 中使用 KVM Guest 镜像

您可以使用一个已经准备就绪的 RHEL KVM guest qcow2 镜像（可以从 RHEL 7 KVM Guest Image 或 RHEL 6.6 KVM Guest Image 获得）。这些镜像被配置为使用 **cloud-init**，并需要使用 ec2 兼容的元数据服务来提供 SSH 密钥。

注意

对于 KVM guest 镜像：

- 这个镜像中的 **root** 帐号被禁用，但是，可以通过 **sudo** 来使用一个名为 **cloud-user** 的特殊用户。
- 这个镜像服务没有 **root** 密码设置。

root 密码在 **/etc/shadow** 中被锁定（在第 2 项中加了 **!!**）。

对于一个 OpenStack 实例，我们推荐用户通过 OpenStack dashboard 或命令行产生一个 ssh 密钥对，并使用它们进行 SSH 公共验证来作为实例的 root 用户。

在启动实例时，公共密钥会被植入。您可以使用在创建密钥对时下载的私人密钥进行验证。

如果您不想使用密钥对，您可以使用在为一个实例添加一个 **admin** 密码一节中介绍的过程创建 **admin** 密码。

如果您需要创建一个自定义的 Red Hat Enterprise Linux 镜像，请参阅创建一个 Red Hat Enterprise Linux 7 镜像或创建一个 Red Hat Enterprise Linux 6 镜像。

1.2.1.2. 创建自定义的 Red Hat Enterprise Linux 镜像

先决条件：

- 使用一个运行 Linux 的主机创建镜像。它可以是任何可以安装并运行 Linux 软件包的机器。
- libvirt 和 virt-manager（运行 **yum groupinstall -y @virtualization**）。这会安装创建一个 guest 操作系统所需的所有软件包。
- Libguestfs 工具程序（运行 **yum install -y libguestfs-tools-c**）。这会安装一组用来访问和修改虚拟机镜像的工具程序。
- 一个 Red Hat Enterprise Linux 7 ISO 文件（RHEL 7.0 Binary DVD 或 RHEL 6.6 Binary DVD）。
- 一个用来修改 **kickstart** 文件的文本编辑程序。

注意

在以下的过程中，所有带有 **[user@host]#** 提示符的命令都需要在您的主机系统上运行。

1.2.1.2.1. 创建一个 Red Hat Enterprise Linux 7 镜像

本节介绍了使用 Red Hat Enterprise Linux 7 ISO 文件手工创建 OpenStack 兼容的 .qcow2 格式镜像的步骤。

1. 使用 **virt-install** 开始进行安装：

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

启动一个实例并开始安装过程。

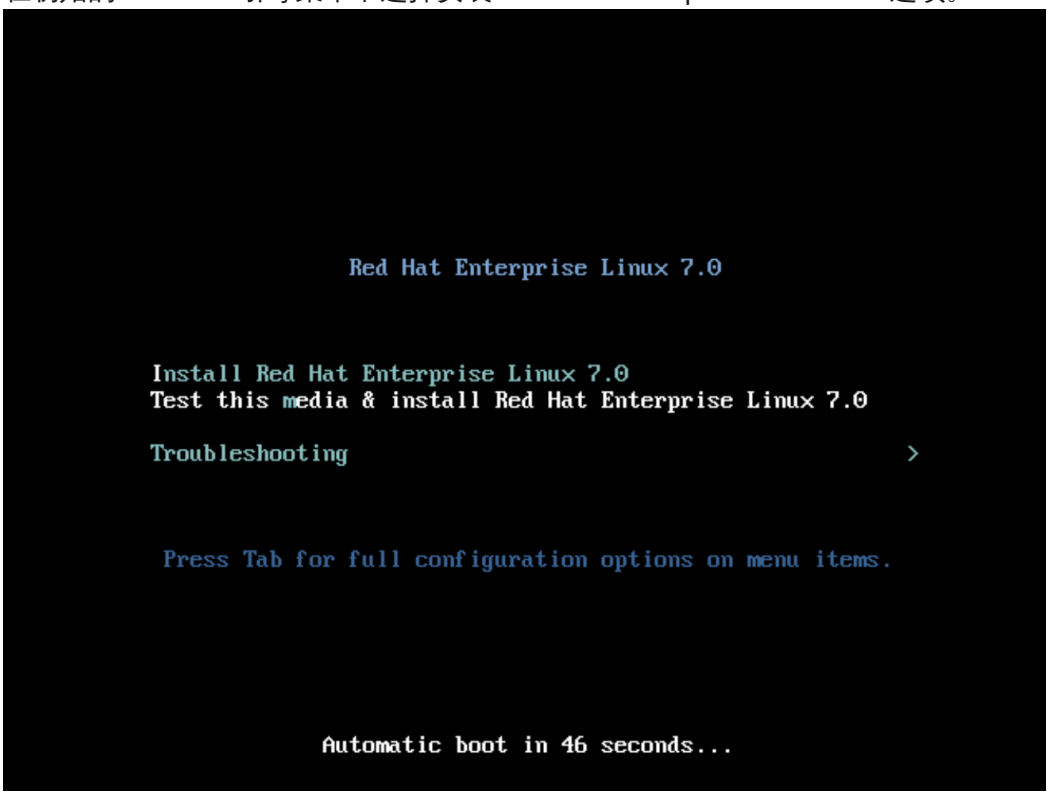
注意

如果实例没有自动启动，运行以下命令：

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

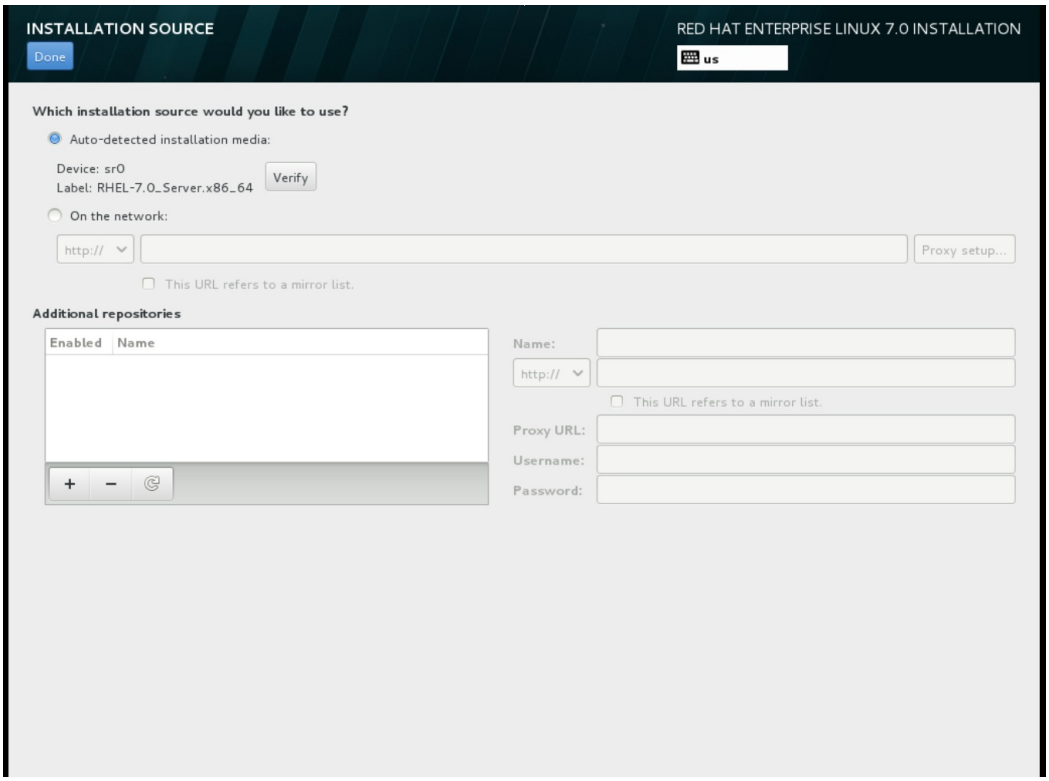
2. 按以下内容设置虚拟机：

   a. 在初始的 Installer 引导菜单中选择安装 Red Hat Enterprise Linux 7.0 选项。

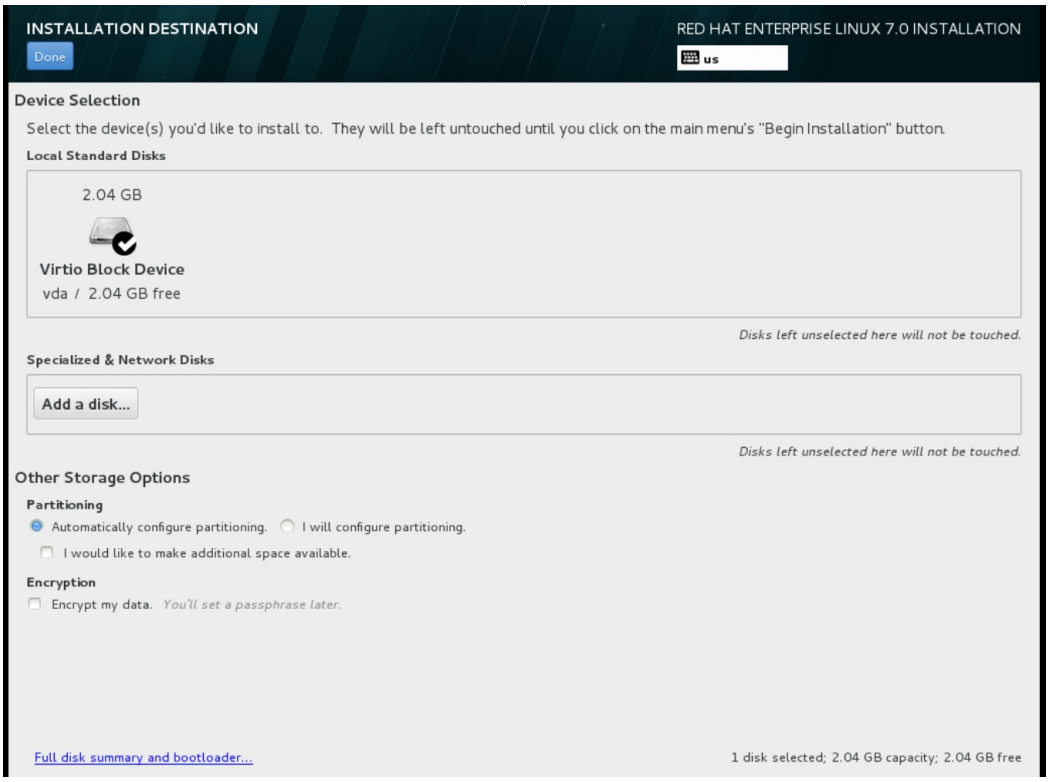


   b. 选择适当的 **Language** 和 **Keyboard** 选项。

c. 在提示输入安装所使用的设备类型时，选择 **Auto-detected installation media**。



d. 在提示输入安装目标的类型时，选择 **Local Standard Disks**。

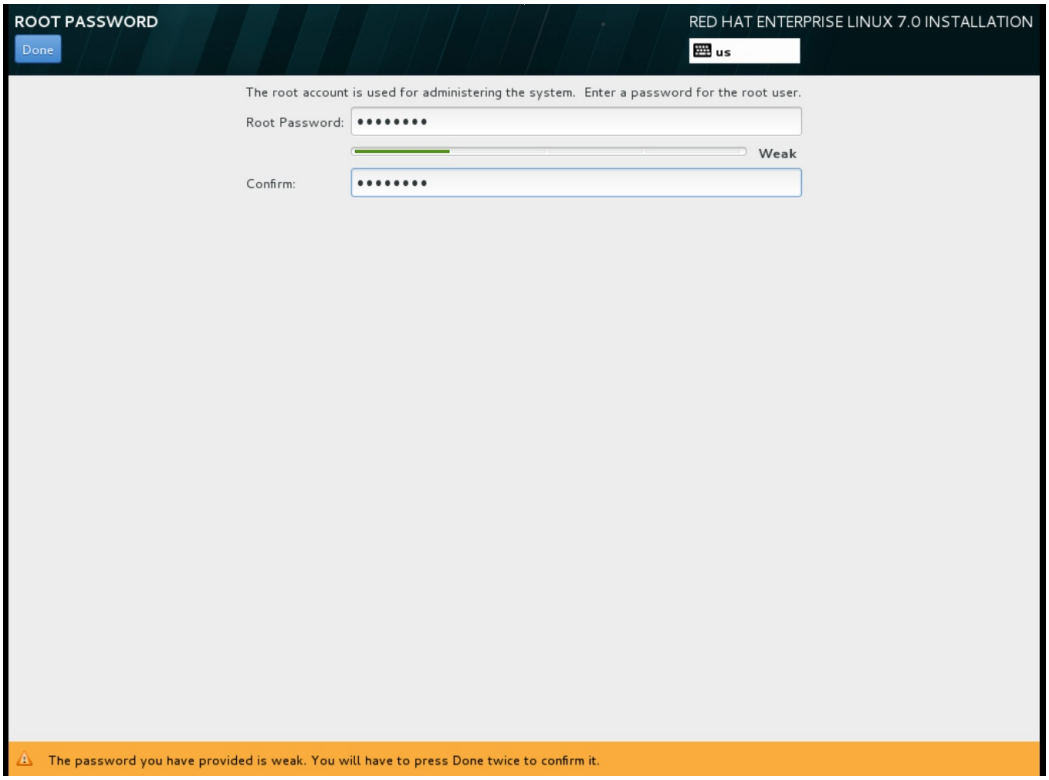


为其它存储选项选择 **Automatically configure partitioning**。

e. 对于软件，选择 **Minimal Install**。

f. 对于网络和主机名，为您的设备选择 **eth0** 作为网络，并输入设备的 **hostname**。默认的主机名是 **localhost.localdomain**。

g. 选择 **root** 密码。



安装过程就此完成，**Complete!** 界面会出现。

3. 安装完成后，重启实例并以 root 用户身份登录。

4. 更新 **/etc/sysconfig/network-scripts/ifcfg-eth0** 文件，使它只包括以下内容：

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

5. 重启机器。

6. 在 Content Delivery Network 中注册机器：

```
# subscription-manager register
```

a. 在提示时输入您的用户门户网站（Customer Portal）的用户名和密码：

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

b. 找到包括所需频道的权利池：

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

c. 使用在上一步所获得的权利池 ID 来把 **Red Hat Enterprise Linux Server** 的权利附加到系统上：

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

d. 启用所需的频道：

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

对于 RHEL OpenStack Platform 7，所需的频道是 **rhel-7-server-openstack-7.0-rpms** 和 **rhel-7-server-rh-common-rpms**。

注意

如需更多相关信息，请参阅 *Installation Reference* 中的 "Subscribe to the Required Channels"。

7. 更新系统。

```
# yum -y update
```

8. 安装 **cloud-init** 软件包。

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

9. 编辑 **/etc/cloud/cloud.cfg** 配置文件，在 **cloud_init_modules** 下添加以下内容：

```
- resolv-conf
```

当实例第一次引导时，**resolv-conf** 选项会自动配置 **resolv.conf**。这个文件包括了与实例相关的信息，如 **nameservers**、**domain** 和其它选项。

10. 把以下行添加到 **/etc/sysconfig/network** 以避免访问 EC2 元数据服务时出现问题。

```
NOZEROCONF=yes
```

11. 为了确保控制台的信息出现在 dashboard 的**日志**标签页中，以及 **nova console-log** 输出中，把以下的引导选项添加到 **/etc/default/grub** 文件中：

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

运行以下命令：

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

这会输出以下结果：

```
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. 取消注册虚拟机。这样可以避免通过对创建的镜像进行克隆所产生的镜像都有相同的订阅请求。

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

13. 关闭实例：

```
# poweroff
```

14. 使用 **virt-sysprep** 命令重置并清理镜像，从而可以避免在创建实例时出现问题：

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

15. 使用 **virt-sparsify** 命令减少镜像的大小。这个命令会把磁盘镜像中的所有空闲空间转换为主机的空闲空间：

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

这会在运行此命令的地方创建一个新的 **rhel7-cloud.qcow2** 文件。

rhel7-cloud.qcow2 镜像文件可以被上传到 Image 服务。如需了解使用 dashboard 把镜像文件上传到 OpenStack 部署中的信息，请参阅上传镜像。

1.2.1.2.2. 创建 Red Hat Enterprise Linux 6 镜像

本节介绍了使用 Red Hat Enterprise Linux 6 ISO 文件手工创建 OpenStack 兼容的 .qcow2 格式镜像的步骤。

1. 使用 **virt-install** 开始进行安装：

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

启动一个实例并开始安装过程。

注意

如果实例没有自动启动，运行以下命令：

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

2. 按以下内容设置虚拟机：

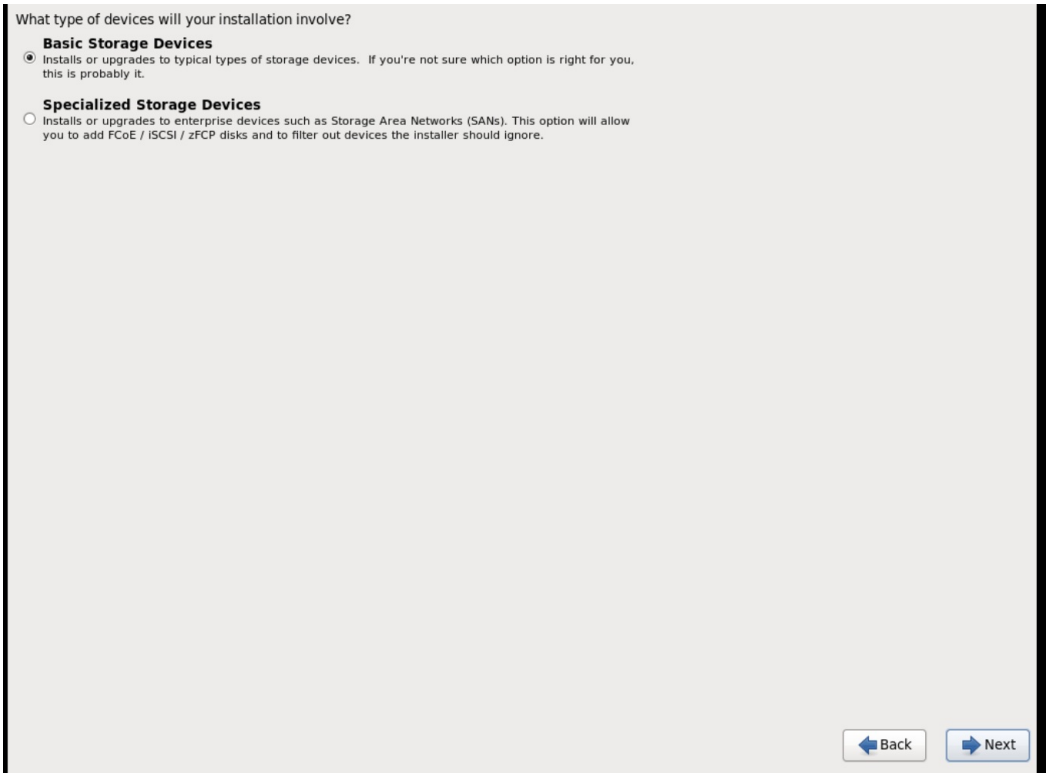
   a. 在初始的 Installer 引导菜单中选择 **Install or upgrade an existing system** 选项。

   根据提示进行安装的过程。接受默认值。

   安装的过程会检查磁盘并进行一个**介质检查**操作。当检查**成功完成**后，会弹出磁盘。

   b. 选择适当的 **Language** 和 **Keyboard** 选项。

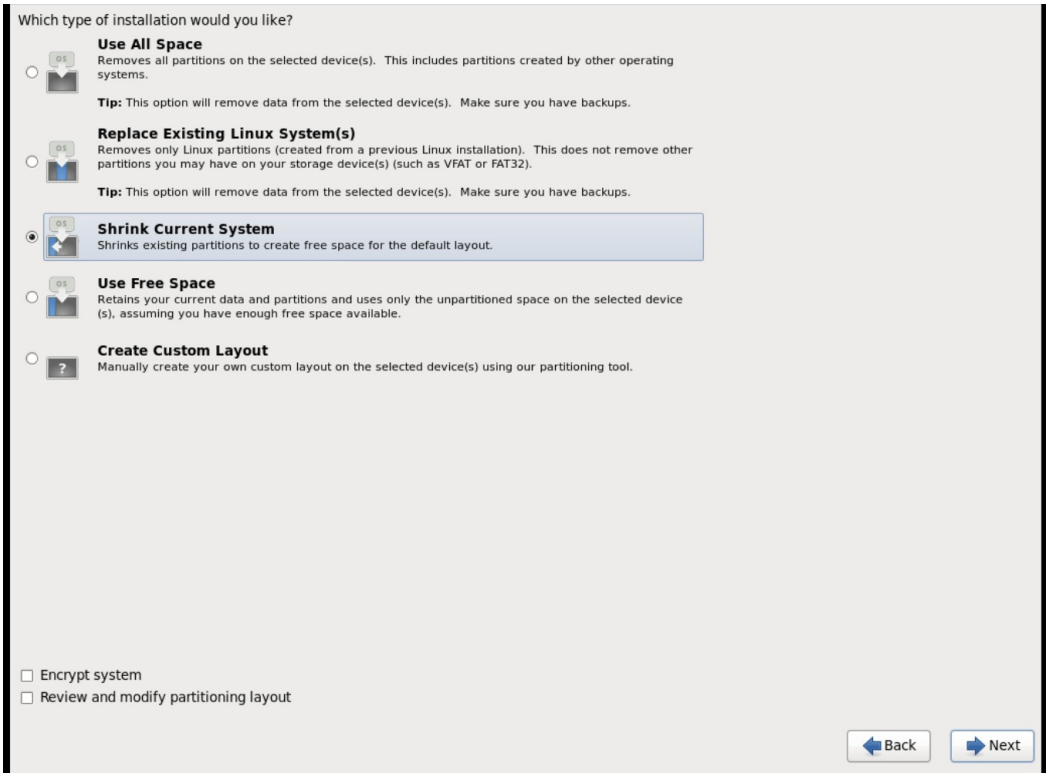
c. 在提示输入安装所使用的设备类型时，选择 **Basic Storage Devices**。



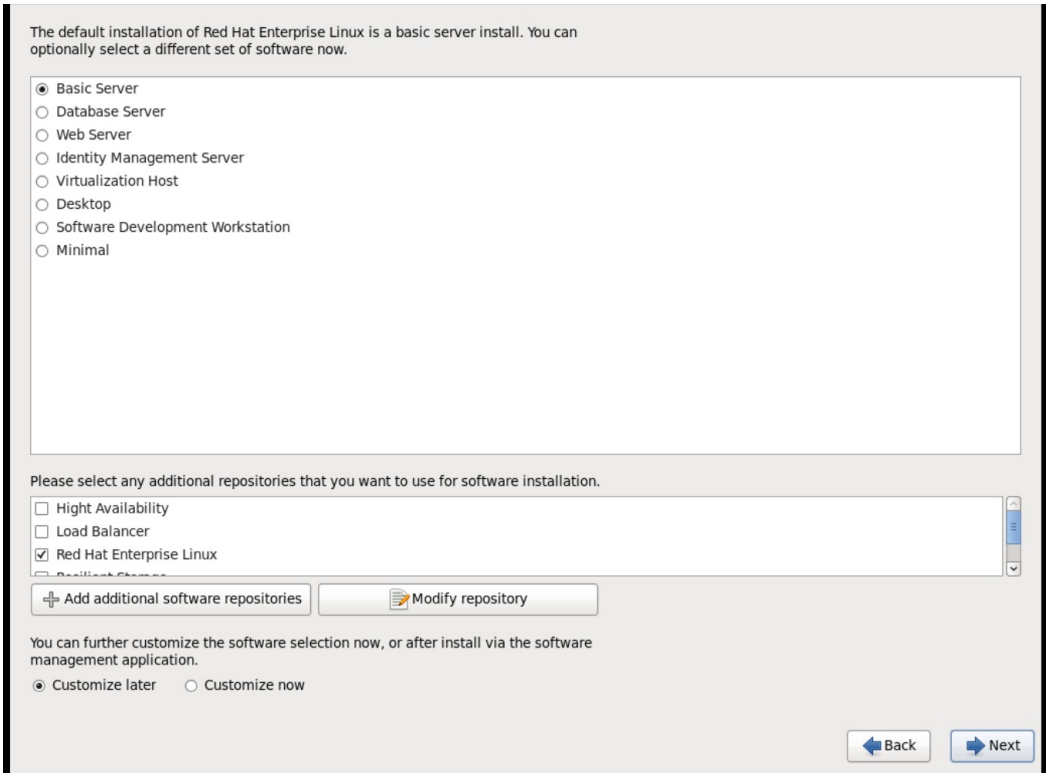
d. 为主机选择一个 **hostname**。默认的主机名是 **localhost.localdomain**。

e. 设置 **timezone** 和 **root** 密码。

f. 根据磁盘的空间选择安装类型。



g. 选择 **Basic Server** 安装，这会安装一个 SSH 服务器。



h. 安装过程完成，**Congratulations, your Red Hat Enterprise Linux installation is complete** 界面会出现。

3. 重启实例，并以 **root** 用户登录。

4. 更新 **/etc/sysconfig/network-scripts/ifcfg-eth0** 文件，使它只包括以下内容：

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

5. 重启机器。

6. 在 Content Delivery Network 中注册机器：

```
# subscription-manager register
```

a. 在提示时输入您的用户门户网站（Customer Portal）的用户名和密码：

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

b. 找到包括所需频道的权利池：

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

c. 使用在上一步所获得的权利池 ID 来把 **Red Hat Enterprise Linux Server** 的权利附加到系统上：

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

d. 启用所需的频道：

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

对于 RHEL OpenStack Platform 7，所需的频道是 **rhel-7-server-openstack-7.0-rpms** 和 **rhel-6-server-rh-common-rpms**。

注意

如需更多相关信息，请参阅 *Installation Reference* 中的 "Subscribe to the Required Channels"。

7. 更新系统。

```
# yum -y update
```

8. 安装 **cloud-init** 软件包。

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

9. 编辑 **/etc/cloud/cloud.cfg** 配置文件，在 **cloud_init_modules** 下添加以下内容：

```
- resolv-conf
```

当实例第一次引导时，**resolv-conf** 选项会自动配置 **resolv.conf**。这个文件包括了与实例相关的信息，如 **nameservers**、**domain** 和其它选项。

10. 为了防止网络出现问题，创建 **/etc/udev/rules.d/75-persistent-net-generator.rules** 文件。

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

这会避免创建 **/etc/udev/rules.d/70-persistent-net.rules** 文件。如果创建了 **/etc/udev/rules.d/70-persistent-net.rules** 文件，当从快照引导时，网络可能无法正常工作（网络接口被创建为 "eth1"，而不是 "eth0"。IP 地址也没有被分配）。

11. 把以下行添加到 **/etc/sysconfig/network** 以避免访问 EC2 元数据服务时出现问题。

```
NOZEROCONF=yes
```

12. 为了确保控制台的信息出现在 dashboard 的**日志**标签页中，以及 **nova console-log** 输出中，把以下的引导选项添加到 **/etc/grub.conf** 中：

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

13. 取消注册虚拟机。这样可以避免通过对创建的镜像进行克隆所产生的镜像都有相同的订阅请求。

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

14. 关闭实例：

```
# poweroff
```

15. 使用 **virt-sysprep** 命令重置并清理镜像，从而可以避免在创建实例时出现问题：

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

16. 使用 **virt-sparsify** 命令减少镜像的大小。这个命令会把磁盘镜像中的所有空闲空间转换为主机的空闲空间：

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

这会在运行此命令的地方创建一个新的 **rhel6-cloud.qcow2** 文件。

注意

根据应用到这个实例的 flavor 中的磁盘空间设置，您需要手工调整实例分区的大小。

rhel6-cloud.qcow2 镜像文件可以被上传到 Image 服务。如需了解使用 dashboard 把镜像文件上传到 OpenStack 部署中的信息，请参阅上传镜像。

1.2.2. 上传镜像

1. 在 dashboard 中，选择**项目 > Compute > 镜像**。
2. 点**创建镜像**。
3. 输入相关的值后点**创建镜像**。

项	备注
名称	镜像的名称。在一个项目中，不同镜像的名称需要是唯一的。
描述	对镜像的简单描述。
镜像源	镜像源：镜像地址或镜像文件。根据您的选择，以下项会被显示。

项	备注
镜像地址或镜像文件	选择镜像地址选项来指定镜像的 URL。 选择镜像文件选项来从本地磁盘上传一个镜像。
格式	镜像格式（例如 qcow2）。
构架	镜像架构。例如，使用 i686 作为 32 位架构；或使用 x86_64 作为 64 位架构。
最小磁盘 (GB)	引导镜像所需的最小磁盘大小。如果没有指定，默认值是 0（没有最小磁盘大小的限制）。
最低内存 (MB)	引导镜像所需的最低内存大小。如果没有指定，默认值是 0（没有最低内存大小的限制）。
公有	如果选择这个选项，这个镜像对所有可以访问这个项目的用户有效。
受保护的	如果选择了这个选项，只有具有特定权限的用户可以删除这个镜像。

注意

另外，您也可以使用带有 **property** 选项的 **glance image-create** 命令创建镜像。在这个命令的命令行中可以使用多个值（请参阅镜像配置参数）。

1.2.3. 更新镜像

1. 在 dashboard 中，选择**项目 > Compute > 镜像**。
2. 从下拉菜单中选**编辑镜像**。

注意

编辑镜像选项只在以 **admin** 用户登录时才有效。当以 **demo** 用户登录时，有效的选项是**启动云主机**和**创建云硬盘**。

3. 更新相关的值后点**更新镜像**。您可以编辑以下值：名称、描述、内核 ID、ramdisk ID、架构、格式、最小磁盘、最低内存、公共、受保护的。
4. 点下拉菜单并选**更新元数据**选项。

5. 通过把左面栏中的项添加到右面栏来指定元数据。左面的栏中包括了在 Image Service Metadata Catalog 中定义的元数据。如果需要添加您自己选择的关键字的元数据，选**其它**，然后点**保存**。

注意

另外，您也可以使用带有 **property** 选项的 **glance image-update** 命令创建镜像。在这个命令的命令行中可以使用多个值（请参阅镜像配置参数）。

1.2.4. 删除镜像

1. 在 dashboard 中，选择**项目 > Compute > 镜像**。
2. 选择需要删除的镜像后点**删除镜像**。

第 2 章 配置 OPENSTACK COMPUTE 存储

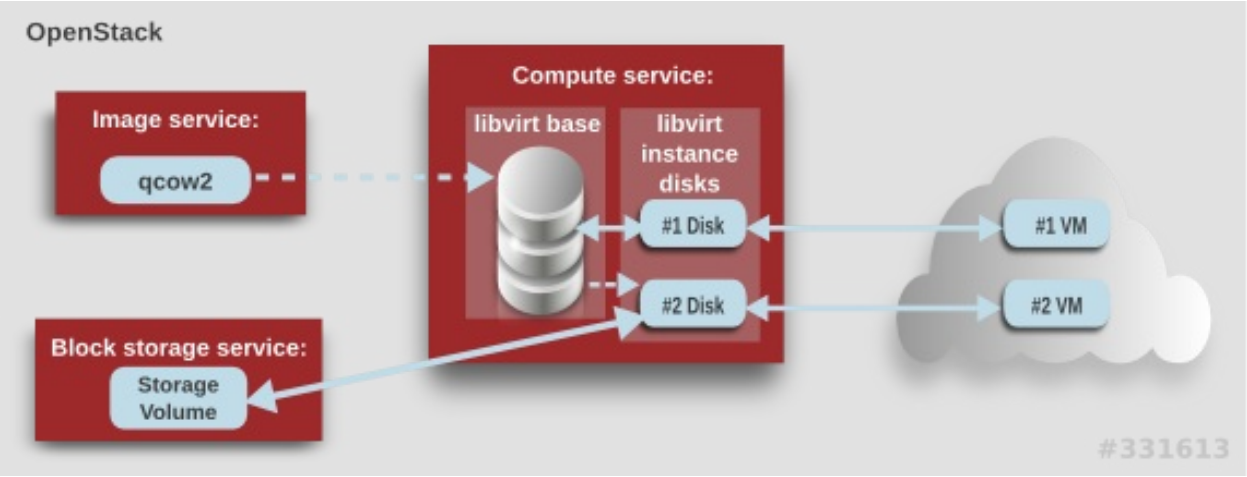
本节介绍了 OpenStack Compute（nova）中的镜像后端存储的架构，以及基本的配置选项。

2.1. 架构概述

在 Red Hat Enterprise Linux OpenStack Platform 中，OpenStack Compute 服务使用 KVM hypervisor 来执行计算负载。**`libvirt`** 驱动被用来处理和 KVM 间的所有交流，并启用对虚拟机的创建功能。

Compute 服务需要考虑两类 **`libvirt`** 存储：

- 基础镜像（base image）：Image 服务镜像的一个缓存的、具有特定格式的副本。
- 实例磁盘（Instance disk）：由 **`libvirt`** 创建作为虚拟机实例的后端。实例磁盘可以存储在 Compute 的临时存储（使用 **`libvirt`**），也可以存储在一个具有持久性的存储中（如使用 Block Storage）。



Compute 服务创建虚拟机实例的步骤是：

1. 缓存 Image 服务的后端镜像作为 **`libvirt`** 的基础。
2. 把基础镜像转换为 raw 格式（如果被配置）。
3. 调整基础镜像的大小来匹配虚拟机 flavor 的规范。
4. 使用基础镜像创建 libvirt 实例磁盘。

在上面的图中，#1 实例磁盘使用临时存储；#2 磁盘使用块存储卷。

2.2. 配置

用来处理 **`libvirt`** 基础镜像和实例磁盘的 Compute 配置（参数在 **`/etc/nova/nova.conf`** 文件中定义）会决定环境的性能和安全性。

表 2.1. Compute 镜像参数

项	参数	描述	默认
[DEFAULT]	**force_raw_images**	是否把 **non-raw** 缓存的基础镜像转换为 **raw**（布尔值）。如果 non-raw 镜像被转换为 raw，Compute 将会： - 不允许备份文件（可能会产生安全性的问题）。 - 删除存在的压缩（避免 CPU 瓶颈）。 把基础镜像转换为 raw 格式会比 hypervisor 直接使用镜像（例如，一个 qcow2 镜像）需要更多空间。如果您的系统有较慢的 I/O 或空间比较小，您可以把它设置为 'false'，这样可以通过提高数据压缩/解压操作对 CPU 的要求来换取对输入带宽要求最小化的目的。 raw 基础镜像需要和 **libvirt_images_type=lvm** 一起使用。	**true**
[DEFAULT]	**use_cow_images**	**libvirt** 实例磁盘是否使用 CoW（Copy on Write）（布尔值）： - **false** - 使用 raw 格式。在不使用 CoW 时，磁盘镜像的公用部分会需要更多的存储空间。 - **true** - 使用 cqow2 格式。在使用 CoW 时，每个虚拟机都有自己的数据副本，因此可以提高多系统共存的性能（这取决于后端的存储和主机缓存）。	true
[DEFAULT]	**preallocate_images**	**libvirt** 实例磁盘的预分配模式。它的值可以是： - **none** - 在实例启动时不分配存储。 - **space** - 在实例启动时完全分配存储（使用 **fallocate**），这可以保证获得所需的存储空间和 I/O 性能。 即使没有使用 CoW 实例磁盘，每个虚拟机所获得的数据副本也是稀疏的，因此，虚拟机可能会在运行时出现 ENOSPC 错误。通过在实例磁盘镜像中运行 **fallocate(1)**，Compute 会马上并高效地为它们在文件系统中分配存储空间（如果支持这个功能）。因为在运行时，文件系统不需要动态地分配数据块，所以运行时的性能也会提高。	none

项	参数	描述	默认
[DEFAULT]	**resize_fs_using_block_device**	是否启用通过访问块设备上的基础镜像来直接调整基础镜像的功能（布尔值）。使用较旧版本的 **cloud-init** 的镜像不需要这个设置（它们无法调整自身的大小）。 因为这个参数会启用直接挂载镜像的功能，而这个功能通常会因为安全性的原因被禁用，所以它在默认情况下没有被启用。	**false**
[DEFAULT]	**default_ephemeral_format**	一个新临时卷的默认格式。它的值可能是：**ext2**、**ext3** 或 **ext4**。对于大的、新的磁盘，**ext4** 格式会比 **ext3** 提供更快的初始化时间。您可以使用 **guest_format** 配置选项单独对每个系统中的这个设置进行覆盖。	**ext4**
[DEFAULT]	**image_cache_manager_interval**	在连续两次运行镜像缓存管理器（image cache manager）间需要等待的秒数，这会影响到 libvirt compute 节点的基础缓存。在这段间隔时间内会自动删除不再使用的缓存镜像（请参阅 **remove_unused_base_images** 和 **remove_unused_original_minimum_age_seconds**）。	**2400**
[DEFAULT]	**remove_unused_base_images**	是否启动自动删除不使用的基础镜像的功能（每 **image_cache_manager_interval** 秒检查一次）。当镜像在 **remove_unused_original_minimum_age_seconds** 秒的时间内没有被访问，则认为它是 **unused**。	**true**
[DEFAULT]	**remove_unused_original_minimum_age_seconds**	在从 **libvirt** 缓存中删除未使用的基础镜像前，这个基础镜像存在的时间（**remove_unused_base_images**）。	**86400**

项	参数	描述	默认
[**libvirt**]	**images_type**	为 **libvirt** 实例磁盘使用的镜像类型（替代 **use_cow_images**）。它的有效值是：**raw**、**qcow2**、**lvm**、**rbd** 和 **default**。如果指定为 **default**，**use_cow_images** 参数的值会被使用。	**default**

第 3 章 虚拟机实例

OpenStack Compute 是一个中心组件，它根据需要提供虚拟机。Compute 会通过 Identity 服务进行用户验证；使用 Image 服务来获得镜像（用来启动实例）；使用 dashboard 服务作为用户和管理界面。

RHEL OpenStack Platform 可以使您方便地管理云中的实例。Compute 服务会创建、调度并管理实例，并使其它 OpenStack 组件可以使用这个功能。本章介绍了相关的操作，以及添加组件（如密钥对、安全组、主机集合、flavor）的信息。*实例（instance）* 在 OpenStack 中被用来代表虚拟机实例。

3.1. 管理实例

在创建实例前，您需要确保其它 OpenStack 组件（例如网络、密钥对、作为引导源的镜像或卷）已为实例准备好。

本节讨论了添加这些组件的步骤，以及创建和管理实例的内容。管理实例包括更新实例、登录到实例、查看实例是如何被使用的、调整实例的大小以及删除实例。

3.1.1. 添加组件

按照以下小节中介绍的内容创建网络、密钥对以及上传镜像或卷源的内容。当创建实例时需要这些元素，而它们在默认的情况下不会被自动提供。同时，您还需要创建一个新的安装组来提供对用户的 SSH 访问。

1. 在 dashboard 中选择**项目**。
2. 选择**网络 > 网络**，确定已经存在了一个新实例可以加入的私人网络。如需了解创建网络的信息，请参阅 **Networking Guide** 的 **Add a Network** 一节（Red Hat Enterprise Linux OpenStack Platform）。
3. 选择**Compute > 访问 & 安全 > 密钥对**，确定已经存在了一个密钥对（如需了解创建密钥对的信息，请参阅 第 3.2.1.1 节 “创建一个密钥对”）。
4. 确定您有一个可以作为引导源的镜像或云硬盘：
   - 要查看引导源镜像，选择**镜像**标签页（如需了解创建镜像的信息，请参阅 第 1.2.1 节 “创建镜像”）。
   - 要查看引导源云硬盘，选择**云硬盘**标签页（如需了解创建云硬盘的信息，请参阅 第 4.1.1 节 “创建云硬盘”）。
5. 选择 **Compute > 访问 & 安全 > 安全组**，确定您已经创建了一个安全组规则（如需了解创建安全组的信息，请参阅 **Users and Identity Management Guide** 中的 **Project Security Management**（Red Hat Enterprise Linux OpenStack Platform）。

3.1.2. 创建实例

1. 在 dashboard 中选择**项目 > Compute > 实例**。
2. 点**启动云主机**。
3. 输入所需的信息（带有 '* '标记的项是必需的）后点**运行**。

标签页	项	备注
详情	可用域	域（zone）就是可以放置您的实例的、包括云资源的逻辑组。如果您不能确定您的选择，使用默认域（如需了解更多信息，请参阅 第 3.4 节 “管理主机集合”）。
	云主机名称	实例的名称。
	云主机类型	云主机类型决定了实例可以获得什么资源（例如，内存）。如需了解更多与云主机类型相关的信息，请参阅 第 3.3 节 “管理 flavor”。
	云主机启动源	根据所做的选择，新的项会被显示用来选择启动源： - 镜像源必须和 OpenStack 兼容（请参阅 第 1.2 节 “管理镜像”）。 - 如果一个云硬盘或云硬盘源被选择，这个源必须使用一个镜像来格式化（请参阅 第 4.1 节 “基本云硬盘（Volume）的使用和配置”）。
访问和安全	密钥对	指定的密钥对会被植入到实例中，并在使用 SSH 远程访问实例时使用它们。通常情况下，每个项目都会创建一个密钥对。
	安全组	安全组包括了一组防火墙规则用来过滤并处理实例的网络数据（如需了解创建安全组的信息，请参阅 **Users and Identity Management Guide** 中的 **Project Security Management**（Red Hat Enterprise Linux OpenStack Platform）。
Networking	已选择的网络	您需要最少指定一个网络。一般情况下，实例会被分配一个私有网络，然后会获得一个浮动 IP 地址用于外部连接。

标签页	项	备注
创建后	定制脚本源	您可以提供一组命令或一个脚本文件，当实例被引导后运行它们。例如，使用它们来设置实例的主机名 ，或设置一个用户密码。如果选择了 '直接输入'，您可以在 '脚本数据项' 中直接输入您的命令；如果使用其它选项，则指定您的脚本文件。  注意 任何以 '#cloud-config' 开始的脚本都会被解析为使用 cloud-config 语法（如需了解与此语法相关的信息，请参阅 http://cloudinit.readthedocs.org/en/latest/topics/examples.html）。
高级选项	磁盘分区	在默认情况下，实例只包括一个分区，并可以根据需要自动调整它的大小。但是，您也可以选择手工配置您自己的分区。
	配置驱动器	如果选择了这个选项，OpenStack 把元数据写到一个只读的配置驱动器中，在引导时会被附加到实例中（而不是加入到 Compute 的元数据服务中）。在实例被引导后，您可以挂载这个驱动器来查看其中的内容（您可以为实例提供文件）。

3.1.3. 更新实例（使用操作菜单）

为了更新实例，您可以选择**项目 > Compute > 实例**，并在 **操作** 栏中选择所需的操作。

操作	描述
创建快照	快照可以保留正在运行的实例的磁盘状态。
绑定浮动 IP/解除浮动 IP 的绑定	在一个实例可以和外部网络进行通讯，或被外部用户访问前，您需要为这个实例关联一个浮动 IP（外部）地址。因为外部子网中的外部地址数量是有限的，所以我们推荐您释放那些不使用的地址。

操作	描述
编辑实例	更新实例的名称和关联的安全组。
编辑安全组	为实例添加和删除安全组。如需了解创建安全组的信息，请参阅 **Users and Identity Management Guide** 中的 **Project Security Management**（Red Hat Enterprise Linux OpenStack Platform）。
控制台	在浏览器中查看实例的控制台（可以方便地访问实例）。
查看日志	查看实例的控制台日志中的最新内容。打开后，您可以点 'View Full Log' 来查看日志的全部内容。
暂停/恢复实例	马上暂停实例（不需要用户进行确认）；实例的当前状态会保存在内存（RAM）中。
挂起/恢复实例	马上挂起实例（不需要用户进行确认）；和休眠一样，实例的当前状态会保存在磁盘中。
重新调整实例的大小	打开重启调整实例窗口（请参阅 第 3.1.4 节 “重新调整实例的大小”）。
软重启	安全地停止并重启实例。软重启会在重启实例前尝试安全地停止所有进程。
硬重启	停止并重启实例。硬重启只是简单地关闭实例的*电源*，然后对它进行重启。
关闭实例	安全地停止实例。
重新构建实例	使用新的镜像和磁盘分区选择来重新构建镜像（关闭、重新构建镜像和重新启动实例）。当操作系统出现问题时，这个选项比终止实例并重新开始更容易。

操作	描述
终止实例	永久删除实例（您会被要求进行确认）。

您可以创建并分配一个外部 IP 地址（请参阅 第 3.2.3 节 “创建、分配和释放浮动 IP 地址”）

3.1.4. 重新调整实例的大小

要调整一个实例的大小（内存或 CPU 的数量），您需要为这个实例选择一个适当的新 flavor（有时也被称为“云主机类型”）。如果您需要增加它的大小，则需要保证包括它的主机有足够的空间。

1. 设置每个主机来使用 SSH 密钥进行身份验证，从而使 Compute 可以使用 SSH 来在主机间移动磁盘（例如，compute 节点可以共享相同的 SSH 密钥）。

   如需了解更多与设置 SSH 密钥验证相关的信息，请参阅 **Migrating Instances Guide** 中的 **Configure SSH Tunneling Between Nodes**（Red Hat Enterprise Linux OpenStack Platform）。

2. 在 **/etc/nova/nova.conf** 文件中设置以下参数来允许在原始主机上重新调整大小：

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

3. 在 dashboard 中选择**项目 > Compute > 实例**。
4. 点主机的**操作**箭头，选**调整云主机大小**。
5. 在**新的云主机类型**项中选择一个新 flavor。
6. 如果您需要在启动实例时手工对它进行分区（构建时间会比较快）：

   a. 选择**高级选项**。

   b. 在**磁盘分区**项中选择**手工**。

7. 点**重新调整大小**。

3.1.5. 连接到一个实例

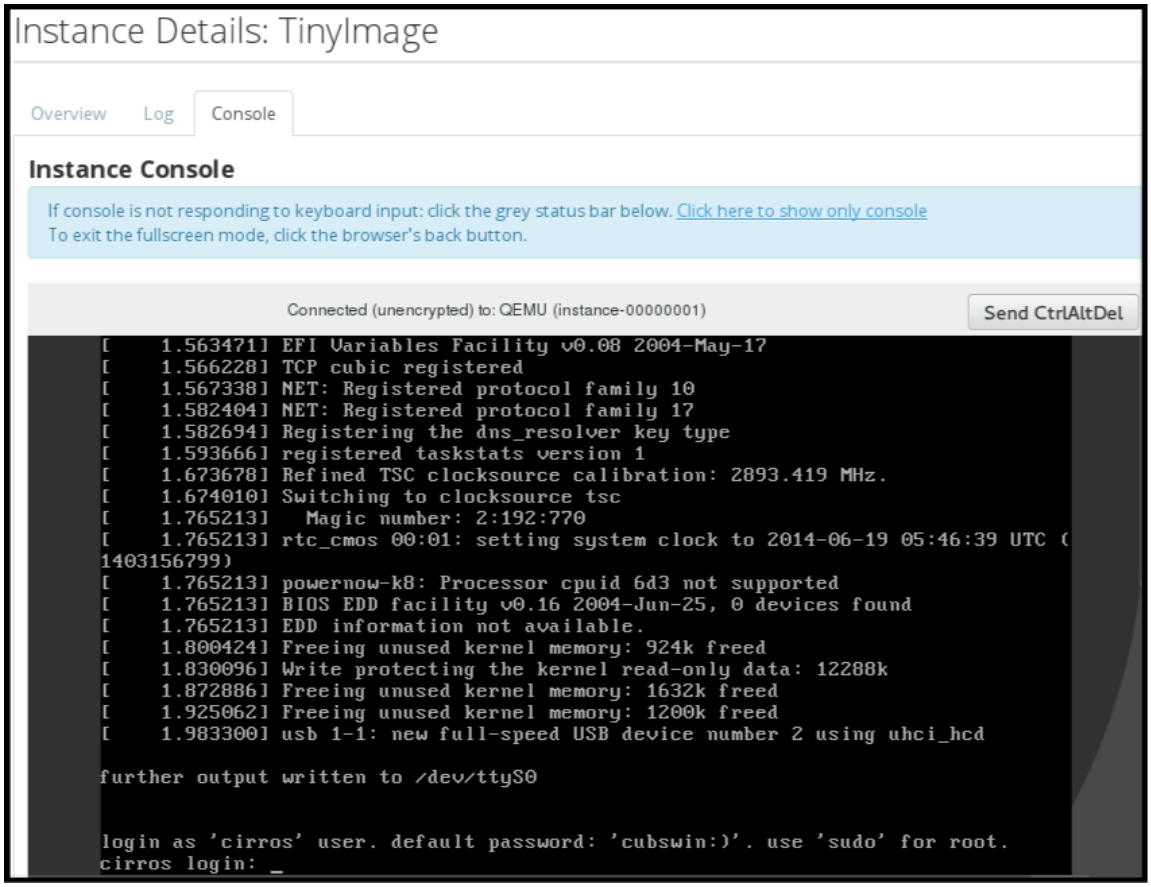
本节介绍了通过 dashboard 或命令行接口访问一个实例控制台的方法。您也可以直接连接到实例的串口来进行故障排除（在网络连接失败时也可以使用这个方法）。

3.1.5.1. 使用 Dashboard 访问实例的控制台

您可以在 dashboard 中使用控制台（console）直接连接到您的实例。

1. 在 dashboard 中选择 **Compute > 实例**。

2. 点实例的**更多**按钮并选**控制台**。



3. 使用镜像的用户名和密码进行登录（例如，一个 CirrOS 镜像使用 *cirros/cubswin:)*）。

3.1.5.2. 直接连接到一个 VNC 控制台

您可以使用 **`nova get-vnc-console`** 命令所返回的 URL 来直接访问一个实例的 VNC 控制台。

浏览器

要获得一个浏览器 URL，使用：

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

Java 客户端

要获得一个 Java-client URL，使用：

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

注意

nova-xvpvncviewer 提供了一个 Java 客户端的简单示例。要下载客户端，运行：

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

使用实例的 Java-client URL 来运行 viewer：

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

提供这个工具的目的只是方便用户的使用，红帽并不正式支持它。

3.1.5.3. 直接连接到一个串行控制台

您可以使用一个 websocket 客户端来直接访问实例的串口。串口连接通常被作为故障排除工具使用（例如，在网络配置失败时，也可以通过它来访问实例）。要获得一个正在运行的实例 URL，运行以下命令：

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

注意

novaconsole 提供了一个 websocket 客户端的简单示例。要下载客户端，运行：

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

使用实例的串口 URL 运行客户端：

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

提供这个工具的目的只是方便用户的使用，红帽并不正式支持它。

但是，根据所进行的安装，管理员可能会需要首先设置 nova-serialproxy 服务。这个代理服务是一个 websocket 代理，它提供了连接到 OpenStack Compute 串口的功能。

3.1.5.3.1. 安装并配置 `nova-serialproxy`

1. 安装 **`nova-serialproxy`** 服务：

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

2. 更新 **`/etc/nova/nova.conf`** 文件中的 **`serial_console`** 部分：

   a. 启用 **`nova-serialproxy`** 服务：

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

b. 指定用来创建由 **nova get-serial-console** 命令提供的 URLS 的字符串。

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

其中的 **PUBLIC_IP** 是运行 **nova-serialproxy** 服务的主机的公共 IP 地址。

c. 指定实例串口控制台监听的 IP 地址（字符串）。

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

d. 指定需要连接的 proxy client 的地址（字符串）。

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

其中的 **HOST_IP** 是您的 Compute 主机的 IP 地址。例如，对于一个启用了 **nova-serialproxy** 服务的配置会和以下类似：

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

3. 重启 Compute 服务：

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

4. 启动 **nova-serialproxy** 服务：

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

5. 重启所有正在运行的实例，从而使它们可以监听正确的网络套接字。

6. 打开防火墙的串行控制器端口连接。串口的端口范围通过 **/etc/nova/nova.conf** 中的 **[serial_console]`** port_range 进行设置。在默认情况下，它的范围是 10000:20000。更新 iptables：

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

3.1.6. 查看实例的使用

使用情况的统计数据可以基于：

- 每个项目

  要查看每个项目的使用情况，选**项目 > Compute > 概况**。所有项目实例的使用数据会被显示。

您可以通过指定时间范围并点**提交**来查看特定时间段内的使用情况数据。

- 每个虚拟机管理器

  如果以管理员的身份进行登录，您也可以查看所有项目的信息。点**管理员 > 系统**，选择其中的一个标签页。例如，**资源使用情况**标签页中提供了特定时间段中的使用情况数据报告。您也可以点**虚拟机管理器**来查看您当前的 vCPU、内存或磁盘的使用情况统计数据。

注意

vCPU 使用情况的值（**y 中的 x**）反映了所有虚拟机的总 vCPU 数（x）和虚拟机管理器的内核数量（y）。

3.1.7. 删除实例

1. 在 dashboard 中选择**项目 > Compute > 实例**，然后选择您的实例。
2. 点**终止云主机**。

注意

删除一个实例的操作并不会删除附加到它上面的卷，您需要单独删除它们（请参阅第 4.1.4 节 “删除卷”）。

3.2. 管理实例安全

您可以通过为实例分配正确的安全组（设置防火墙规则）和密钥对（启用 SSH 用户访问）来管理对它的访问。另外，还可以为实例分配一个浮动 IP 地址，使它可以被外部网络访问。以下介绍了创建和管理密钥对、安全组、浮动 IP 地址以及使用 SSH 登录到一个实例的信息，以及为实例添加一个 **admin** 密码的方法。

如需了解更多与管理安全组相关的信息，请参阅 **Users and Identity Management Guide** 中的 **Project Security Management**（Red Hat Enterprise Linux OpenStack Platform）。

3.2.1. 管理密钥对

密钥对提供了到实例的 SSH 访问能力。每次创建一个密钥对时，它的证书都会被下载到本地系统，从而可以分发给用户。通常情况下，每个项目都会创建一个密钥对（可以在多个实例中使用）。

您也可以为 OpenStack 导入一个已经存在的密钥对。

3.2.1.1. 创建一个密钥对

1. 在 dashboard 中，选**项目 > Compute > 访问 & 安全**。
2. 在**密钥对**标签页中点**创建密钥对**。
3. 在**密钥对名称**项中输入一个名称，点**创建密钥对**。

当密钥对被创建后，一个密钥对文件会通过浏览器自动下载。您需要保存这个文件用于以后的连接。对于使用命令行的 SSH 连接，您可以使用以下命令把这个文件加载到 SSH：

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

3.2.1.2. 导入一个密钥对

1. 在 dashboard 中，选**项目 > Compute > 访问 & 安全**。
2. 在**密钥对**标签页中点**导入密钥对**。
3. 在**密钥对名称**项中输入一个名称，把您的公共密钥复制/粘贴到**公共密钥**项中。
4. 点**导入密钥对**。

3.2.1.3. 删除密钥对

1. 在 dashboard 中，选**项目 > Compute > 访问 & 安全**。
2. 在**密钥对**标签页中点相应密钥的**删除密钥对**按钮。

3.2.2. 创建安全组

安全组就是一组 IP 过滤器规则，它可以分配给项目实例，用来设置对实例的网络访问。安全组是与项目相关联的，项目成员可以编辑安全组的默认设置，并可以向其添加新规则。

1. 在 dashboard 中，选**项目**标签页，点 **Compute > 访问 & 安全**。
2. 在**安全组**标签页中，点 **+ 创建安全组**。
3. 为安全组提供一个名称和描述，点**创建安全组**。

如需了解更多与管理项目安全的信息，请参阅 **Users and Identity Management Guide** 中的 **Project Security Management**（Red Hat Enterprise Linux OpenStack Platform）。

3.2.3. 创建、分配和释放浮动 IP 地址

在默认情况下，实例会在首次创建时被分配一个内部 IP 地址。但是，您可以通过创建和分配一个浮动 IP 地址（外部地址）来使它可以通过公共网络进行访问。无论实例处于什么状态，您都可以修改与实例相关联的 IP 地址。

每个项目的可用浮动 IP 地址资源都是有限的（在默认情况下，是 50 个地址），因此您需要在某个 IP 地址不再被使用时释放它，从而使它可以被重复使用。浮动 IP 地址只能从一个存在的浮动 IP 地址池中分配。如需了解更多相关信息，请参阅 **Networking Guide** 中的 **Create Floating IP Pools**（Red Hat Enterprise Linux OpenStack Platform）。

3.2.3.1. 为项目分配一个浮动 IP 地址

1. 在 dashboard 中，选**项目 > Compute > 访问 & 安全**。
2. 在**浮动 IP** 标签页中，点**分配 IP 给项目**。
3. 在**资源池**中选择用来分配 IP 地址的网络。
4. 点**分配 IP**。

3.2.3.2. 分配一个浮动 IP

1. 在 dashboard 中，选**项目 > Compute > 访问 & 安全**。
2. 在**浮动 IP** 标签页中，点相关地址的**关联**按钮。
3. 在 IP 地址项中选择要分配的地址。

注意

如果没有可用的地址，您可以点 **+** 按钮创建一个新地址。

4. 在**待连接的端口**项中选择要被关联的实例。一个实例只能和一个浮动 IP 地址相关联。
5. 点**关联**。

3.2.3.3. 释放浮动 IP

1. 在 dashboard 中，选**项目 > Compute > 访问 & 安全**。
2. 在**浮动 IP** 标签页中，点地址的菜单箭头（在**关联/取消关联**按钮旁）。
3. 选择**释放浮动 IP**。

3.2.4. 登录到实例

先决条件：

- 确保实例的安全组中有一个 SSH 规则。请参阅 **Users and Identity Management Guide** 中的 **Project Security Management**（Red Hat Enterprise Linux OpenStack Platform）。
- 确保实例有一个与之关联的浮动 IP 地址（外部地址）。请参阅 创建、分配和释放浮动 IP 地址。
- 获得实例的密钥对证书。当密钥对被创建时，这个证书会被下载。如果您没有自己创建密钥对，请联系您的系统管理员（如需更多相关信息，请参阅 第 3.2.1 节 “管理密钥对”）。

首先把密钥对文件加入到 SSH，然后使用它：

1. 修改所产生的密钥对证书的访问权限。

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

2. 检查 **ssh-agent** 是否已运行：

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

3. 如果还没有运行，使用以下命令运行它：

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

4. 在您的本地机器上，把密钥对证书加载到 SSH。例如：

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

5. 现在，您可以 SSH 到镜像文件。

以下是使用 **cloud-user** 用户 SSH 到 Red Hat Enterprise Linux 虚拟机镜像的示例：

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

注意

您也可以直接使用证书。例如：

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

3.2.5. 为实例添加一个 admin 密码

您可以使用以下方法为一个实例添加一个 **admin**（**root**）密码。

1. 在 **/etc/openstack-dashboard/local_settings** 文件中，把 **change_set_password** 参数的值设为 **True**。

```
can_set_password: True
```

2. 在 **/etc/nova/nova.conf** 文件中，把 **inject_password** 参数的值设为 **True**。

```
inject_password=true
```

3. 重启 Compute 服务。

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

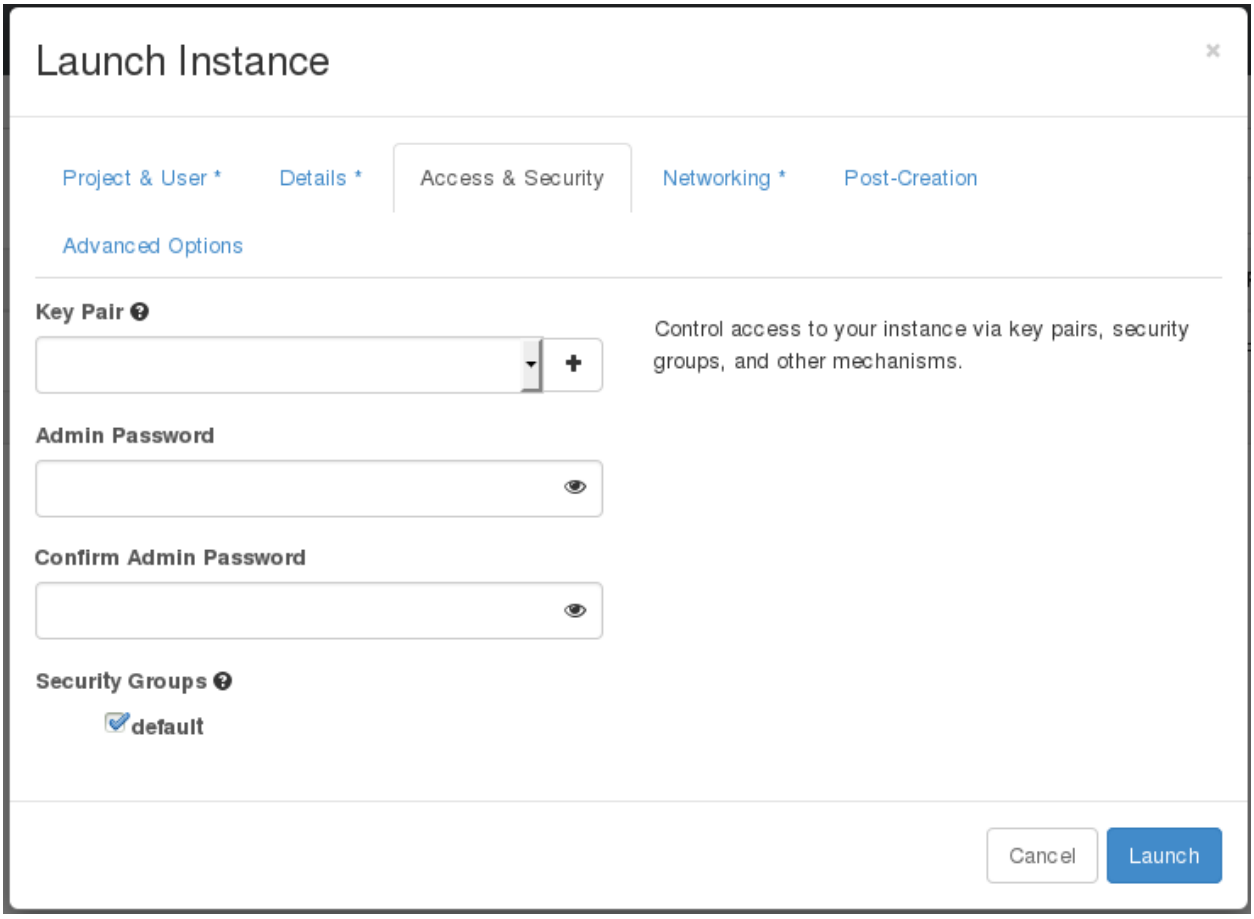
当使用 **nova boot** 命令启动一个新实例时，命令的输出中会显示一个 **adminPass** 参数。您可以使用它作为 **root** 用户的密码登录到这个实例。

Compute 服务会覆盖 **/etc/shadow** 文件中的 **root** 用户的密码。它同时也会为 KVM 客户机镜像激活 **root** 账户。如需了解更多与使用 KVM 客户机镜像相关的信息，请参阅 第 1.2.1.1 节 “在 RHEL OpenStack Platform 中使用 KVM Guest 镜像”

您也可以使用 dashboard 设置一个自定义密码。在把 **can_set_password** 参数的值设为 **true** 后运行以下命令：

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

新添加的 **admin** 密码项如下所示：



这些项可以在启动或重建一个实例时使用。

3.3. 管理 FLAVOR

每个创建的实例都会被分配一个 flavor（资源模板），它定义了实例的大小和容量。Flavor 也可以指定二级临时存储（secondary ephemeral storage）、交换磁盘、元数据来限制资源使用或对特定项目的访问（默认的 flavor 都没有定义这些额外的属性）。flavor 在 Red Hat Enterprise OpenStack Platform 中有时也被称为“云主机类型”。

表 3.1. 默认 Flavor

名称	vCPU	RAM	Root 磁盘大小
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

名称	vCPU	RAM	Root 磁盘大小
m1.xlarge	8	16384 MB	160 GB

多数最终用户都可以使用这些默认的 flavor。但是，您可能会需要创建并管理特殊的 flavor。例如，您可能需要：

- 根据所使用的硬件修改默认的内存和能力。
- 添加元数据来为实例强制一个特定的 I/O 速率，或匹配一个主机集合（host aggregate）。

注意

使用镜像属性设置的行为会覆盖使用 flavors 设置的行为。如需了解更多信息，请参阅第 1.2 节 “管理镜像”。

3.3.1. 更新配置权限

在默认情况下，只有管理员可以创建 flavor 或查看完整的 flavor 列表（选择 "管理员 > 系统 > 云主机类型"）。要允许所有用户配置 flavor，在 **`/etc/nova/policy.json`** 文件（nova-api 服务器）中指定以下内容：

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

3.3.2. 创建 flavor

1. 使用一个 admin 用户，在 dashboard 中选择**管理员 > 系统 > 云主机类型**。
2. 点**创建云主机类型**， 指定以下项：

标签页	项	描述
主机类型信息	名称	唯一的名字。
	ID	唯一的 ID。默认值 **`auto`** 会自动产生一个 UUID4 值，但是您可以手工指定一个整数或 UUID4 值。
	VCPU	虚拟 CPU 的数量
	RAM (MB)	内存（以 MB 为单位）

标签页	项	描述
	根磁盘 (GB)	临时磁盘的大小（以 GB 为单位）。要使用镜像本身的大小，把它设为 **0**。当 **Instance Boot Source=Boot from Volume** 时这个磁盘不被使用。
	临时磁盘 (GB)	二级临时磁盘的大小（以 GB 为单位）。
	Swap 磁盘 (MB)	交换磁盘的大小（以 MB 为单位）
云主机类型访问	选择的项目	可以使用这个云主机类型的项目。如果没有选择项目，所有项目都可以访问（**Public=Yes**）。

3. 点创建云主机类型。

3.3.3. 更新常规属性

1. 使用一个 admin 用户，在 dashboard 中选择**管理员 > 系统 > 云主机类型**。
2. 点 flavor 的 **编辑云主机类型**按钮。
3. 更新相关的值并点**保存**。

3.3.4. 更新云主机类型的元数据

除了常规属性外，您还可以为云主机类型添加元数据（**extra_specs**），它可以更精细地定义实例的使用情况。例如，您可以设置允许使用的最大带宽。

- 预定义的关键字决定了硬件支持或配额。它受您所使用的虚拟机管理器（hypervisor）的限制（对于 libvirt，请参阅 表 3.2 “Libvirt 元数据”）。
- 预定义的关键字和用户定义的关键字都可以决定实例的调度。例如，您可以设置 **SpecialComp=True**，任何使用这个云主机类型的实例都将只能在元数据中带有相同关键字和值的主机集合中运行。如需了解更多相关信息，请参阅 第 3.4 节 “管理主机集合”。

3.3.4.1. 查看元数据

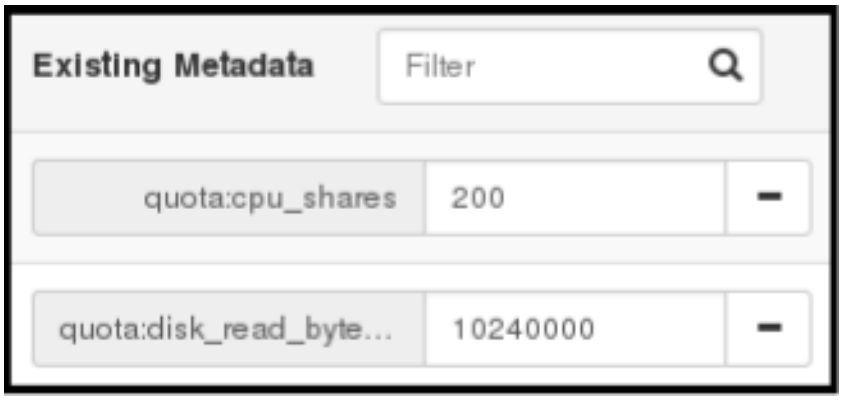
1. 使用一个 admin 用户，在 dashboard 中选择**管理员 > 系统 > 云主机类型**。

2. 点云主机类型的 **Metadata** 链接（**Yes** 或 **No**）。当前所有的值都在右面的 **Existing Metadata** 项中列出。

3.3.4.2. 添加元数据

您可以使用一个**关键字/值**数据对来指定云主机类型的元数据。

1. 使用一个 admin 用户，在 dashboard 中选择**管理员 > 系统 > 云主机类型**。
2. 点云主机类型的 **Metadata** 链接（**Yes** 或 **No**）。当前所有的值都在右面的 **Existing Metadata** 项中列出。
3. 在**可用的元数据**中，点**其它**项，选择您需要添加的关键字（请参阅 表 3.2 “Libvirt 元数据”）。
4. 点 + 按钮；您现在可以看到新的关键字出现在**已存在的元数据**项中。
5. 在右面项中输入关键字的值。



6. 当输入完“关键字/值”数据对后，点**保存**。

表 3.2. Libvirt 元数据

键	描述
hw:action	用来配置每个实例的支持限制。有效值包括： - cpu_max_sockets - 支持的最大 CPU 插槽数量。 - cpu_max_cores - 支持的最大 CPU 内核数量。 - cpu_max_threads - 支持的最大 CPU 线程的量。 - cpu_sockets - 首选的 CPU 插槽数量。 - cpu_cores - 首选的 CPU 内核数量。 - cpu_threads - 首选的 CPU 线程数量。 - serial_port_count - 每个实例的最多串口数量。 例如：**hw:cpu_max_sockets=2**

键	描述

hw:NUMA_def	定义 NUMA 的拓扑。当 flavor 中的 RAM 和 vCPU 的分配值比 compute 主机中的 NUMA 节点大时，通过定义 NUMA 拓扑可以使主机更好地利用 NUMA，并提高客户机 OS 的性能。flavor 中定义的 NUMA 设置会覆盖镜像中的设置。有效的定义包括：
	- numa_nodes - 实例可以看到的 NUMA 节点数量。设置为 '1' 来确保镜像 NUMA 设置被覆盖掉。
	- numa_mempolicy - 内存分配策略。有效策略包括：
	  - strict - 强制实例的内存分配来自于和它绑定的 NUMA 节点（如果设置了 numa_nodes，这是默认值）。
	  - preferred - 内核可以使用其它节点。当 numa_nodes 被设置为 '1' 时会有用。
	- numa_cpus.0 - vCPU N-M 到 NUMA 节点 0 的映射（以逗号分隔）。
	- numa_cpus.1 - vCPU N-M 到 NUMA 节点 1 的映射（以逗号分隔）。
	- numa_mem.0 - 把 N GB 内存映射到 NUMA 节点 0。
	- numa_mem.1 - 把 N GB 内存映射到 NUMA 节点 1。
	- numa_cpu.N 和 numa_mem.N 只有在设置了 numa_nodes 时才有效。另外，它们只有在实例的 NUMA 节点的 CPU 和内存分配不对称时才需要（对于一些 NFV 负载非常重要）。
	  注意
	如果 numa_cpu 或 numa_mem.N 的值比实际有效的值大，则会出现一个异常错误。
	示例（带有 8 个 vCPU 和 4GB 内存的实例）：
	- 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
	调度程序会查找带有 2 个 NUMA 节点的主机，它需要可以在一个节点上运行 6 个 CPU 和 3 GB 内存，在另一个节点上运行 2 个 CPU 和 1 GB 内存。如果某个主机只有一个 NUMA 节点，它可以运行 8 个 CPU 和 4 GB 内存，这个主机不会被认为是一个有效的匹配。无论 numa_mempolicy 是如何设置的，相同的逻辑都适用于调度程序。

键	描述
hw:watchdog_action	当实例出现故障时，可以使用实例的 watchdog 设备触发一个操作。有效的操作是： - disabled - 没有附加的设备（默认值）。 - pause - 暂停实例。 - poweroff - 强制关闭实例。 - reset - 强制重置实例。 - none - 启用 watchdog，但当实例出现故障时不进行任何操作。 例如：**`hw:watchdog_action=poweroff`**
hw_rng:action	通过使用镜像属性把一个随机数生成器设备添加到镜像中（请参阅 RHEL OpenStack Platform 文档的 "Command-Line Interface Reference" 部分中的 hw_rng_model 内容）。 如果添加了设备，有效的操作将包括： - allowed - 如果为 **`True`**，这个设备被启用；如果为 **`False`**，设备被禁用。在默认情况下，设备被禁用。 - rate_bytes - 在每个 rate_period 周期中，为了填充它的熵池，实例的内核可以从主机读的最大字节数（整数）。 - rate_period - 读持续的时间（以秒为单位，整数）。 例如：**`hw_rng:allowed=True`**。
hw_video:ram_max_mb	视频设备允许使用的最大内存数量（以 MB 为单位）。 例如：**`hw:ram_max_mb=64`**

键	描述
quota:option	对实例的限制。有效选项包括： - cpu_period - 强制 cpu_quota 限制的时间（以 ms 为单位）。在指定的 cpu_period 时间中，每个 vCPU 不能占用超过 cpu_quota 的运行时。这个值的范围必须是 [1000, 1000000]，'0' 代表没有值。 - cpu_quota - 在每个 cpu_period 时间段中 vCPU 所允许的最大带宽（以 ms 为单位）。这个值的范围必须是 [1000, 18446744073709551]，'0' 代表没有值，负值代表 vCPU 不被控制。cpu_quota 和 cpu_period 可以被用来保证所有 vCPU 以相同的速度运行。 - cpu_shares - 域共享 CPU 的时间。这个值只有在对同一个域中的其它机器加权重的情况下才有意义。一个云主机类型的相关值为 '200' 的实例会获得比相关值为 '100' 的实例多一倍的机器时间。 - disk_read_bytes_sec - 每秒磁盘可以读取的最大数据量（以字节为单位）。 - disk_read_iops_sec - 每秒所允许的最大读 I/O 操作。 - disk_write_bytes_sec - 每秒磁盘可写的最大数据量（以字节为单位）。 - disk_write_iops_sec - 每秒所允许的最大写 I/O 操作。 - disk_total_bytes_sec - 每秒所允许的最大吞吐量总和的限制。 - disk_total_iops_sec - 每秒所允许的最大总 I/O 操作。 - vif_inbound_average - 期望的平均流入网络流量。 - vif_inbound_burst - 在 vif_inbound_peak 速度上可以接收的最大网络流量。 - vif_inbound_peak - 可以接收流入网络数据的最大速率。 - vif_outbound_average - 期望的平均流出网络流量。 - vif_outbound_burst - 在 vif_outbound_peak 速度上可以发送的最大网络流量。 - vif_outbound_peak - 可以发送流出网络数据的最大速率。 例如：**`quota:vif_inbound_average=10240`**

3.4. 管理主机集合

为了性能和管理的目的，一个单一的 Compute 部署可以被分成不同的逻辑组。OpenStack 会使用以下的相关术语：

- *主机集合（Host aggregate）* - 一个主机集合就是 OpenStack 部署中的一个逻辑组，它包括了一组 Compute 主机，以及相关的元数据。一个主机可以属于多个主机集合。只有系统管理员可以查看或创建主机集合。

  一个主机集合的元数据通常被用来为 Compute 调度器（scheduler）提供信息（如为一部分主机限制特定的云主机类型或镜像）。当在一个主机集合中指定了元数据，只有那些在云主机类型中指定了相同元数据的实例才可以使用其中的主机。

系统管理员可以通过主机集合实现负载均衡、强制物理隔离（或冗余）、对带有相同属性的服务器进行分组或分离不同的硬件系统。当您创建一个集合时，需要指定一个域名（zone name），它会作为最终用户可以看到的这个集合的名称。

- *可用域（Availability zone）* - 可用域就是最终用户可以看到的主机集合。最终用户无法看到这个域是由哪些主机组成的，也无法看到这个域的元数据，而只能看到域的名称。

  系统管理员可以设置包括特定计算能力的域，或位于特定区域中的域，最终用户将可以使用这些配置的域。

3.4.1. 启用主机集合调度

在默认情况下，主机集合元数据不会被用来过滤实例的使用情况，您需要更新 Compute 的调度配置来启用元数据：

1. 编辑 **/etc/nova/nova.conf** 文件（您需要具有 root 或 nova 用户的权限）。
2. 确定 **scheduler_default_filters** 参数包括：
   - **AggregateInstanceExtraSpecsFilter**（对于主机集合元数据）。例如：

     ```
     scheduler_default_filters=AggregateInstanceExtraSpecsFilter,
     RetryFilter,RamFilter,ComputeFilter,ComputeCapabilitiesFilt
     er,ImagePropertiesFilter,CoreFilter
     ```

   - **AvailabilityZoneFilter** 在启动一个实例时的有效域主机的规格。例如：

     ```
     scheduler_default_filters=AvailabilityZoneFilter,RetryFilte
     r,RamFilter,ComputeFilter,ComputeCapabilitiesFilter,ImagePr
     opertiesFilter,CoreFilter
     ```

3. 保存配置文件。

3.4.2. 查看可用域或主机集合

使用一个 admin 用户，在 dashboard 中选择**管理员 > 系统 > 主机集合**。当前所有定义的集合会在**主机集合**项中列出，所有域会在**可用域**项中列出。

3.4.3. 添加一个主机集合

1. 使用一个 admin 用户，在 dashboard 中选择**管理员 > 系统 > 主机集合**。当前所有定义的集合会在**主机集合**项中列出。
2. 点**创建主机集合**。
3. 在**名称**项中指定集合的名称，最终用户将会在**可用域**项中看到这个名称。
4. 点**在集合中管理主机**。
5. 点相应主机的 + 图标来选择主机。
6. 点**创建主机集合**。

3.4.4. 更新一个主机集合

1. 使用一个 admin 用户，在 dashboard 中选择**管理员 > 系统 > 主机集合**。当前所有定义的集合会在**主机集合**项中列出。
2. 更新实例的**名称**或**可用域**：
   - 点集群的**编辑主机集群**按钮。
   - 更新**名称**或**可用域**的值，点**保存**。
3. 更新实例的**分配的主机**：
   - 在**操作**项中选集合的箭头图标。
   - 点**管理主机**。
   - 点主机的 + 或 - 图标来修改主机的分配。
   - 完成后点**保存**。
4. 更新实例的**元数据**：
   - 在**操作**项中选集合的箭头图标。
   - 点**更新元数据**按钮。所有当前值都在右面的**存在的元数据**项中列出。
   - 在**可用的元数据**中点**其它**项，输入您要添加的关键字。您可以使用预定义的关键字（请参阅 表 3.3 “主机集合元数据”）或添加您自己的关键字（只在实例的 flavor 中设置了相同关键字时才有效）。
   - 点 + 按钮；您现在可以看到新的关键字出现在**已存在的元数据**项中。

     注意

     点 - 图标删除一个关键字。

   - 点**保存**。

表 3.3. 主机集合元数据

键	描述
`cpu_allocation_ratio`	设置虚拟 CPU 对物理 CPU 的分配比率。这取决于为 Compute 调度设置的 **`AggregateCoreFilter`** 过滤。
`disk_allocation_ratio`	设置虚拟磁盘对物理磁盘的分配比率。这取决于为 Compute 调度设置的 **`AggregateDiskFilter`** 过滤。
`filter_tenant_id`	如果指定，集合只包括这个租户（项目）的主机。这取决于为 Compute 调度设置的 **`AggregateMultiTenancyIsolation`** 过滤。

键	描述
ram_allocation_ratio	设置虚拟内存对物理内存的分配比率。这取决于为 Compute 调度设置的 **AggregateRamFilter** 过滤。

3.4.5. 删除一个主机集群

1. 使用一个 admin 用户，在 dashboard 中选择**管理员 > 系统 > 主机集合**。当前所有定义的集合会在**主机集合**项中列出。
2. 从集群中删除所有分配的主机：
    a. 在**操作**项中选集合的箭头图标。
    b. 点**管理主机**。
    c. 点主机的 - 图标删除主机。
    d. 完成后点**保存**。
3. 在**操作**项中选集合的箭头图标。
4. 点**删除主机集合**。

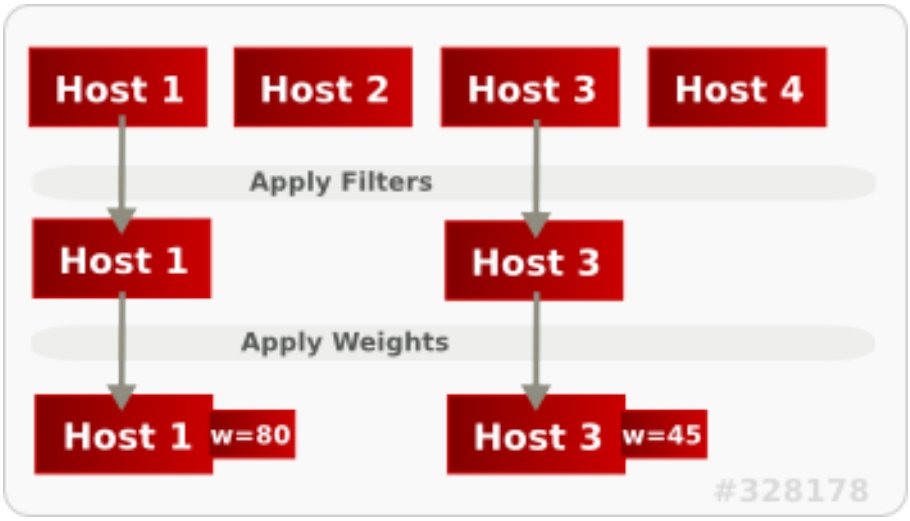
3.5. 调度主机和 CELL

Compute 调度服务决定了一个实例要被放置在哪个 cell 或主机（或主机集合）中。作为一个管理员，您可以设置调度的决定过程。例如，您可以把实例调度到特定组中的主机，或具有特定 RAM 的主机中。

您可以配置以下组件：

- Filters（过滤器） - 决定可以放置实例的初始主机范围（请参阅 第 3.5.1 节 “配置调度过滤器”）。
- Weights（权重） - 通过过滤程序后的主机再根据权重系统来进行优先级排列。具有最高权重值的主机有最高的优先级（请参阅 第 3.5.2 节 “配置调度权重”）。
- Scheduler service（调度程序服务） - 在调度程序所在主机的 **/etc/nova/nova.conf** 文件中包括了以下配置选项，它决定了调度程序如何执行它的任务、处理权重和过滤。主机和 cell 都有调度程序。如需了解这些选项的完整列表，请参阅 "Configuration Reference" (RHEL OpenStack Platform Documentation)。

在下图中，主机 1 和主机 3 都通过了过滤程序。主机 1 具有最高的权重，因此在调度过程中有最高优先级。



3.5.1. 配置调度过滤器

您可以使用 scheduler_default_filters 选项（**/etc/nova/nova.conf** 文件）来定义调度程序（scheduler）使用的过滤器（您需要具有 root 或 nova 用户的权限）。过滤器可以被添加或删除。

默认情况下，以下过滤器被配置并在调度程序中运行：

```
scheduler_default_filters=RetryFilter,AvailabilityZoneFilter,RamFil
ter,ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,S
erverGroupAntiAffinityFilter,ServerGroupAffinityFilter
```

一些过滤器会使用通过命令中的参数传递给实例的信息：

- **nova boot** 命令，请参阅 RHEL OpenStack Platform Documentation 中的 "Command-Line Interface Reference"。
- 实例的 flavor（参阅 第 3.3.4 节 “更新云主机类型的元数据”）
- 实例的镜像（参阅 附录 A, *镜像配置参数*）。

下表列出了可用的过滤器

表 3.4. 调度过滤器

过滤器	描述
AggregateCoreFilter	使用 host-aggregate 元数据关键字 cpu_allocation_ratio 过滤掉超过过度分配（over-commit）比率（虚拟 CPU 和物理 CPU 分配的比率）的主机，它只在为实例指定了主机集合（host aggregate）的情况下有效。
	如果没有设置这个值，过滤器会使用 /etc/nova/nova.conf 文件中的 cpu_allocation_ratio 的值。它的默认值是 **16.0**（每个物理 CPU 可以分配 16 个虚拟 CPU）。

过滤器	描述
AggregateDiskFilter	使用 host-aggregate 元数据关键字 disk_allocation_ratio 过滤掉超过过度分配（over-commit）比率（虚拟磁盘和物理磁盘分配的比率）的主机，它只在为实例指定了主机集合（host aggregate）的情况下有效。
	如果没有设置这个值，过滤器会使用 /etc/nova/nova.conf 文件中的 disk_allocation_ratio 的值。它的默认值是 **1.0**（每个物理磁盘可以分配一个虚拟磁盘）。
AggregateImagePropertiesIsolation	只通过那些元数据和实例镜像的元数据匹配的主机集合中的主机，它只在为实例指定了主机集合的情况下有效。如需了解更多相关信息，请参阅 第 1.2.1 节 “创建镜像”。
AggregateInstanceExtraSpecsFilter	主机集合中的元数据必须与主机的 flavor 元数据相匹配。如需了解更多相关信息，请参阅 第 3.3.4 节 “更新云主机类型的元数据”。
AggregateMultiTenancyIsolation	指定了 filter_tenant_id 的主机只能包括所指定租户（项目）中的实例。  注意 租户仍然可以在其它主机上放置实例。
AggregateRamFilter	使用 host-aggregate 元数据关键字 ram_allocation_ratio 过滤掉超过过度分配（over-commit）比率（虚拟内存和物理内存分配的比率）的主机，它只在为实例指定了主机集合（host aggregate）的情况下有效。
	如果没有设置这个值，过滤器会使用 /etc/nova/nova.conf 文件中的 ram_allocation_ratio 的值。它的默认值是 **1.5**（每个物理内存可以分配 1.5 的虚拟内存）。
AllHostsFilter	通过所有有效的主机（但不禁用其它过滤器）。
AvailabilityZoneFilter	过滤器使用实例指定的可用域。

过滤器	描述
ComputeCapabilitiesFilter	确定 Compute 元数据被正确读取。**:** 前的所有内容被认为是命名空间（namespace）。例如，**quota:cpu_period** 使用 **quota** 作为命名空间，**cpu_period** 是关键字。
ComputeFilter	通过所有可以正常工作并被启用的主机。
CoreFilter	使用 **/etc/nova/nova.conf** 文件中的 cpu_allocation_ratio 过滤掉超过过度分配比率（虚拟 CPU 到物理 CPU 分配比率）的主机。它的默认值是 **16.0**（每个物理 CPU 可以分配 16 个虚拟 CPU）。
DifferentHostFilter	在一个和指定主机不同的主机上创建实例。使用 **nova boot** 命令的 **--different_host** 选项来指定**不同**的主机。
DiskFilter	使用 **/etc/nova/nova.conf** 文件中的 disk_allocation_ratio 过滤掉超过过度分配比率（虚拟磁盘到物理磁盘分配比率）的主机。它的默认值是 **1.0**（每个物理磁盘可以分配 1 个虚拟磁盘）。
ImagePropertiesFilter	只通过匹配实例镜像属性的主机。如需了解更多信息，请参阅 第 1.2.1 节 “创建镜像”。
IsolatedHostsFilter	只通过运行独立镜像（在 **/etc/nova/nova.conf** 文件中使用 **isolated_hosts** 和 **isolated_images**（以逗号分隔的值） 指定）的独立主机。
JsonFilter	接受并使用实例的自定义 JSON 过滤器： - 有效的操作包括：=、<、>、in、⇐、>=、not、or、and - 可识别的变量是：$free_ram_mb、$free_disk_mb、$total_usable_ram_mb、$vcpus_total、$vcpus_used
	这个过滤器通过 **nova boot** 命令的 '--hint query' 指定。例如： **--hint query='['>=', '$free_disk_mb', 200 * 1024]'**
MetricFilter	过滤掉带有无效指标数据的过滤器。

过滤器	描述
NUMATopologyFilter	根据主机的 NUMA 拓扑过滤主机。如果实例没有定义拓扑，所有主机都可以被使用。这个过滤程序会尝试匹配带有完全相同的 NUMA 拓扑的主机。这个过滤器还会查看每个 NUMA 节点的标准过度订阅（over-subscription）的限制，并根据它为 compute 主机提供限制。
RamFilter	使用 **/etc/nova/nova.conf** 文件中的 ram_allocation_ratio 过滤掉超过过度分配比率（虚拟内存到物理内存的分配比率）的主机。它的默认值是 **1.5**（每个物理内存可以分配 1.5 倍的虚拟内存）。
RetryFilter	过滤掉调度失败的主机。它只在 scheduler_max_attempts 大于 0 的情况下有效（**scheduler_max_attempts=3** 是默认设置）。
SameHostFilter	通过一个或多个指定的主机。使用 **nova boot** 的 **--hint same_host** 选项指定主机。
ServerGroupAffinityFilter	只通过一个特定服务器组中的主机： - 设置服务器组的 affinity 策略（nova server-group-create --policy affinity groupName）。 - 构建带有指定组的实例（**nova boot** 选项 **--hint group=UUID**）。
ServerGroupAntiAffinityFilter	只通过还没有包括任何实例的服务器组中的主机： - 设置服务器组的 anti-affinity 策略（**nova server-group-create --policy anti-affinity groupName**）。 - 构建带有指定组的实例（**nova boot** 选项 **--hint group=UUID**）。
SimpleCIDRAffinityFilter	只通过特定 IP 子网范围内（由实例的 cidr 和 **build_new_host_ip** hint 指定）的主机。例如： **--hint build_near_host_ip=192.0.2.0 --hint cidr=/24**

3.5.2. 配置调度权重

cell 和主机可以被加上权重来进行调度，具有最大权重的主机或 cell 将会在经过过滤后被选择。所有的权重器都具有一个乘数（multiplier），它会在规格化节点权重后被应用。一个节点的权重以以下方法计算：

```
w1_multiplier * norm(w1) + w2_multiplier * norm(w2) + ...
```

您可以在调度程序主机的 **/etc/nova/nova.conf** 文件中配置权重选项（需要有 root 或 nova 的用户权限）。

3.5.2.1. 为主机配置权重选项

您可以在 [DEFAULT] scheduler_weight_classes 选项中定义调度程序需要使用的主机权重器：

- **nova.scheduler.weights.ram** - 主机可用内存的权重。
- **nova.scheduler.weights.metrics** - 主机指标数据（metrics）的权重。
- **nova.scheduler.weights.all_weighers** - 使用所有主机权重器（默认）。

表 3.5. 主机权重选项

权重器	选项	描述
所有	[DEFAULT] *scheduler_host_subset_size*	定义所选主机的子集合的大小（整数），它的值必须不小于 **1**。如果是 1，权重功能返回的第 1 个主机会被选择。如果它的值小于 1，这个值会被忽略，并使用 1 作为它的值。
metrics	[metrics] required	指定当 [metrics] **weight_setting** 无效时如何处理 metrics： - **True** - 需要 metrics，如果无效，会产生一个 exception。为了避免产生 exception，在 **scheduler_default_filters** 选项中使用 **MetricFilter**。 - **False** - 无效的 metric 在权重处理过程中被认为是一个负参数；返回的值由 **weight_of_unavailable** 设置。
metrics	[metrics] **weight_of_unavailable**	当 [metrics] **weight_setting** 中的 metric 无效时的权重值。它只在 **required=False** 时有效。
metrics	[metrics] **weight_multiplier**	权重 metric 使用的乘数。在默认情况下，**weight_multiplier=1.0**，并在所有可能的主机中的实例上有效。如果这个值是负数，带有较低 metric 的主机会有高优先值。

权重器	选项	描述
metrics	[metrics] **weight_setting**	指定 metric 以及它们权重的比率，它是一个使用逗号分隔的 **metric=ratio** 对列表。有效的 metric 名是： - **cpu.frequency** - 当前 CPU 的频率 - **cpu.user.time** - CPU 用户模式时间 - **cpu.kernel.time** - CPU 内核时间 - **cpu.idle.time** - CPU 空闲时间 - **cpu.iowait.time** - CPU I/O 等待时间 - **cpu.user.percent** - CPU 用户模式百分比 - **cpu.kernel.percent** - CPU 内核百分比 - **cpu.idle.percent** - CPU 空闲百分比 - **cpu.iowait.percent** - CPU I/O 等待百分比 - **cpu.percent** - 一般的 CPU 利用率 例如：**weight_setting=cpu.user.time=1.0**
ram	[DEFAULT] **ram_weight_multiplier**	RAM 的乘数（浮点值）。在默认情况下是 **ram_weight_multiplier=1.0**，并在所有可能主机上的实例中有效。如果它的值为负数，具有较少 RAM 的主机具有高优先级。

3.5.2.2. 为 cell 配置权重选项

您可以使用 [cells] scheduler_weight_classes 选项（在 **/etc/nova/nova.conf** 文件中）来定义调度器使用的 cell 权重程序。您需要具有 **root** 或 **nova** 用户权限。

有效的权重程序是：

- **nova.cells.weights.all_weighers** - 使用所有 cell 权重程序（默认）。
- **nova.cells.weights.mute_child** - 对一个子 cell 是否已经有一段时间没有发送容量或容量进行权重。
- **nova.cells.weights.ram_by_instance_type** - 对 cell 的有效 RAM 进行权重。
- **nova.cells.weights.weight_offset** - 评估一个 cell 的权重 offset.

注意

在 `**nova-manage cell create** 命令中使用 --woffset 选项可以指定一个 cell 的权重 offset。

表 3.6. Cell 权重选项

权重程序	选项	描述
mute_child	[cells] **mute_weight_multiplier**	已经沉默了一段时间的主机的乘数（multiplier）。它需要是一个负的浮点数，默认值是 **-10.0**。
mute_child	[cells] **mute_weight_value**	沉默主机的权重值。它的值需要是负的浮点数，默认值是 **1000.0**。
ram_by_instance_type	[cells] **ram_weight_multiplier**	权重 RAM 的乘数。它的值需要是一个浮点数，默认值是 **1.0**，并在所有 cell 上的实例中有效。如果它的值为负数，具有较少 RAM 的 cell 具有高优先级。
weight_offset	[cells] **offset_weight_multiplier**	权重 cell 的乘数（浮点值）。允许实例通过把它的权重偏移（weight offset）设置为**9999999999999999**（最高权重具有优先级）来指定一个优先的 cell。它的默认值是 **1.0**。

3.6. 撤离实例

如果您需要把一个实例从一个有故障或已停止运行的 compute 节点上移到同一个环境中的其它主机服务器上时，可以使用 **nova evacuate** 命令对实例进行撤离（evacuate）。

- 撤离的操作只有在实例的磁盘在共享存储上，或实例的磁盘是块存储卷时才有效。因为在其它情况下，磁盘无法被新 compute 节点访问。
- 实例只有在它所在的服务器停止运行的情况下才可以被撤离；如果服务器没有被关闭，**evacuate** 命令会运行失败。

注意

如果您有一个可以正常工作的 compute 节点，您需要：

- 对实例进行一个静态复制（实例没有处于运行状态）作为备份；或把实例复制到其它环境中。使用 **nova image-create** 创建一个快照（请参阅 Migrate a Static Instance）。
- 把一个静态（没有运行的）实例移到相同环境中的另外一个主机（不需要共享存储），使用 **nova migrate**（请参阅 Migrate a Static Instance）。
- 把一个动态（正在运行的）实例移到相同环境中的另外一个主机，使用 **nova live-migration** 命令（请参阅 Migrate a Live (running) Instance）。

3.6.1. 撤离一个实例

1. 使用以下命令撤离一个实例：

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

其中：

- **--password** - 撤离实例的 admin 密码（如果指定了 **--on-shared-storage**，则无法使用）。如果没有指定密码，一个随机密码会被产生，并在撤离操作完成后被输出。
- **--on-shared-storage** - 指定所有实例文件都在共享存储中。
- **instance_name** - 要撤离的实例名称。
- **target_host** - 实例撤离到的主机；如果您没有指定这个主机，Compute 调度会为您选择一个主机。您可以使用以下命令找到可能的主机：

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

例如：

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

3.6.2. 撤离所有实例

1. 使用以下命令在一个特定主机上撤离所有实例：

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

其中：

- **--target** - 实例撤离到的主机；如果您没有指定这个主机，Compute 调度会为您选择一个主机。您可以使用以下命令找到可能的主机：

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

- **--on-shared-storage** - 指定所有实例文件都在共享存储中。
- **source_host** - 被撤离的主机名。

  例如：

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

3.6.3. 配置共享存储

如果您使用共享存储，以下操作会为 Compute 服务把实例目录导出到两个节点，并保证这些节点可以被访问。**/etc/nova/nova.conf** 文件中的 **state_path** 和 **instances_path** 参数指定了目录的路径。以下操作使用默认值 - **/var/lib/nova/instances**。只有具有 root 访问权限的用户才可以设置共享存储。

1. **在控制器主机上：**

   a. 确保 Compute 服务的用户（这个用户必须在控制器和节点中都是相同的）对 **/var/lib/nova/instances** 目录有读和写的访问权限。例如：

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

   b. 把以下行添加到 **/etc/exports** 文件，使用两个 compute 节点的 IP 地址替换其中的 node1_IP 和 node2_IP：

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

   c. 把 **/var/lib/nova/instances** 目录导出到 compute 节点。

```
# exportfs -avr
```

   d. 重启 NFS 服务器：

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

2. **在每个 compute 节点上：**

   a. 确定 **/var/lib/nova/instances** 目录在本地存在。

   b. 把以下行添加到 **/etc/fstab** 文件：

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

   c. 挂载控制器的实例目录（所有设备在 **/etc/fstab** 中列出）：

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

   d. 确认 qemu 可以访问目录的镜像：

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

e. 确定节点可以看到实例目录：

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

注意

您可以运行以下命令查看所有挂载的设备：

```
# df -k
```

第 4 章 管理卷（云硬盘）

卷就是一个块存储设备，它为 OpenStack 实例提供持久性的存储。在 Red Hat Enterprise OpenStack Platform 中，卷有时也被称为云硬盘。

4.1. 基本云硬盘（VOLUME）的使用和配置

以下介绍了对最终用户的云硬盘进行基本管理的方法

4.1.1. 创建云硬盘

1. 在 dashboard 中选择**项目 > Compute > 云硬盘**。
2. 点**创建云硬盘**，编辑以下项：

项	描述
云硬盘名称	云硬盘的名称
描述	对云硬盘的描述信息（可选）
类型	卷类型（可选）。请参阅 第 4.2.3 节 “使用云硬盘类型进行云硬盘分组设置”。 如果您有多个块存储后端，您可以在这里选择一个后端。请参阅 第 4.1.2 节 “为创建的云硬盘指定后端”。
大小（GB）	卷的大小（以 GB 为单位）
可用域	可用域（逻辑服务器组）和主机集合（host aggregate）是在 OpenStack 中进行资源聚合的常用方法。可用域在安装的过程中被定义。如需了解更多与可用域和主机集合相关的信息，请参阅 第 3.4 节 “管理主机集合”。

3. 指定**云硬盘源**：

源	描述
没有源，空白云硬盘	云硬盘为空，不包括文件系统和分区表。

源	描述
快照	使用一个存在的快照作为云硬盘的源。如果您选择这个选项，一个“使用快照作为源”的列表会被显示，您可以从这个列表中选择一个快照。如需了解更多与云硬盘快照相关的信息，请参阅 第 4.1.8 节 “创建、克隆或删除云硬盘快照”。
镜像	使用一个存在的镜像作为云硬盘源。如果您选择这个选项，一个“使用镜像作为源”的列表会出现，您可以从这个列表中选择一个镜像。
卷	使用一个存在的云硬盘作为云硬盘源。如果您选择这个选项，一个“将云硬盘作为源”的列表会出现，您可以从这个列表中选择一个云硬盘。

4. 点**创建云硬盘**。云硬盘被创建后，它的名字会出现在**云硬盘**列表中。

4.1.2. 为创建的云硬盘指定后端

您可以配置 Block Storage 服务来使用多个后端。例如，配置 OpenStack 使用 NFS 后端介绍了如果配置 Block Storage 服务来使用一个 NFS 共享和默认后端的方法。

在创建多个 Block Storage 后端时，您还需要为每个后端创建一个云硬盘类型。然后，您可以使用这个类型来指定创建的云硬盘使用哪个后端。如需了解更多相关信息，请参阅 第 4.2.3 节 “使用云硬盘类型进行云硬盘分组设置”。

在创建云硬盘时通过在“类型”下拉列表中选择云硬盘类型来为它指定后端（请参阅 第 4.1.1 节 “创建云硬盘”）。

如果您在创建云硬盘时没有指定后端，Block Storage 服务会自动选择一个。默认情况下，它会选择有最大可用空间的设备作为后端。您也可以配置 Block Storage 服务在所有可用后端中随机进行选择。如需了解更多相关信息，请参阅 第 4.2.6 节 “配置如何在多后端间分配卷”。

4.1.3. 编辑一个云硬盘的名称或描述信息

1. 在 dashboard 中选择**项目 > Compute > 云硬盘**。
2. 按云硬盘的**编辑云硬盘**按钮。
3. 根据需要修改云硬盘的名称和描述。
4. 点**编辑云硬盘**保存您的改变。

注意

要创建一个加密的卷，首先需要一个可以提供卷加密功能的卷类型。另外，Compute 和 Block Storage 服务都需要被配置为使用相同的静态密钥。如需了解更详细的信息，请参阅 第 4.2.5 节 “使用静态密钥加密卷”。

4.1.4. 删除卷

1. 在 dashboard 中选择**项目 > Compute > 云硬盘**。
2. 在**云硬盘**列表中，选择要删除的云硬盘。
3. 点**删除云硬盘**。

注意

当一个云硬盘（卷）存在快照时，这个云硬盘将无法被删除。如需了解删除快照的信息，请参阅 第 4.1.8 节 “创建、克隆或删除云硬盘快照”。

4.1.5. 为实例附加或取消附加一个云硬盘

实例可以使用一个云硬盘来进行持久保存。一个云硬盘同时只能附加到一个实例。如需了解更多与实例相关的信息，请参阅 第 3.1 节 “管理实例”。

4.1.5.1. 为实例挂载一个云硬盘

1. 在 dashboard 中选择**项目 > Compute > 云硬盘**。
2. 选择云硬盘的**编辑挂载**操作。如果这个云硬盘还没有挂载到一个实例，“挂载到云主机”下拉列表会被显示。
3. 在**挂载到云主机**列表中选择需要挂载这个云硬盘的云主机。
4. 点**挂载云硬盘**。

4.1.5.2. 从实例上断开一个云硬盘

1. 在 dashboard 中选择**项目 > Compute > 云硬盘**。
2. 选择云硬盘的**管理挂载**操作。如果这个云硬盘已被挂载到一个实例，实例的名称会出现在**挂载**项中。
3. 点**端口云硬盘**。

4.1.6. 把云硬盘设为只读

您可以把一个云硬盘设置为允许多个用户对它进行访问，但不能对它的内容进行编辑。使用以下命令把云硬盘设置为**只读**：

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

使用目标云硬盘的 **ID** 替换 **VOLUME**。

如果需要把只读云硬盘设为可读写，运行以下命令：

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

4.1.7. 改变一个云硬盘的所有者

为了改变云硬盘的所有者，您需要进行一个"云硬盘传送（volume transfer）"操作。云硬盘传送操作由云硬盘的当前所有者发起，当云硬盘的新所有者接受了这个操作，云硬盘传送操作就完成了。

4.1.7.1. 通过命令行进行云硬盘传送

1. 以云硬盘当前所有者身份登录。
2. 列出有效的云硬盘：

   ```
   # cinder list
   ```

3. 启动云硬盘操作：

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

   其中的 **VOLUME** 是云硬盘的名称或 **ID**。例如：

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

   cinder transfer-create 命令清除了云硬盘的所有者设置，并为所有者转换的过程创建了一个 **id** 和 **auth_key**。其它用户可以使用这些值接受所有者转换请求，并成为这个云硬盘的新所有者。

4. 新用户现在可以接受成为云硬盘的新所有者。用户需要从命令行登录并运行：

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

   其中，**TRANSFERID** 和 **TRANSFERKEY** 分别是 **cinder transfer-create** 命令返回的 **id** 和 **auth_key** 的值。例如：

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

注意

您可以使用以下命令查看所有有效的云硬盘转移：

```
# cinder transfer-list
```

4.1.7.2. 通过 Dashboard 进行云硬盘传送

通过 Dashboard 进行云硬盘传送

1. 使用云硬盘（卷）所有者的身份在 dashboard 中选择**项目 > 云硬盘**。
2. 在要进行传送的云硬盘的**操作**栏中选**创建传送**。
3. 在**创建传送**对话框中，输入传送的名称，点**新增存储卷迁移**。

   云硬盘传送操作会被创建，在**存储卷迁移**界面中包括了需要发送到接收项目的**传送号**和**认证密钥**信息。

   注意

   认证密钥信息只会出现在**存储卷迁移**界面中。如果您丢失了这个信息，则需要取消这个存储卷迁移，并创建一个新的迁移来产生一个新的认证密钥。

4. 关闭**存储卷迁移**界面返回卷列表。

   卷的状态会变为 **`awaiting-transfer`**，直到接收项目接受了这个转移操作

通过 dashboard 接受一个存储卷转移

1. 使用接收项目所有者的身份在 dashboard 中选择**项目 > 云硬盘**。
2. 点**接受转移**。
3. 在**接受存储卷转移**对话框中，输入从存储卷当前所有者那里获得的**转移号**和**认证密钥**，点**接受存储卷转移**。

   这个存储卷现在会出现在活跃项目的云硬盘列表中。

4.1.8. 创建、克隆或删除云硬盘快照

您可以通过创建一个云硬盘快照来保存一个云硬盘在某个时间点上的状态。然后，可以使用这个快照来克隆新的云硬盘。

警告

为一个已经挂载到某个实例上的云硬盘创建快照可能会破坏快照的数据。如需了解如何把云硬盘从实例上断开，请参阅 第 4.1.5.2 节 “从实例上断开一个云硬盘”。

注意

卷备份和快照有所不同。备份会保存包括在卷中的数据，而快照会保存一个卷在某个时间点上的状态。另外，当一个卷带有快照时，这个卷将无法被删除。卷备份的目的是防止数据丢失，而快照的目的是用于克隆。

因为这个原因，快照后端通常会和存储卷后端在一起，从而可以在进行克隆时最小化延迟的影响。相反的，在一个企业级的环境中，备份存储库通常会存在于不同的位置（例如，在不同的节点上、不同的物理存储上，甚至不同的地理位置）。这可以在存储卷后台被破坏时，备份数据不会受到影响。

如需了解更多与云硬盘备份相关的信息，请参阅 第 4.2.1 节 “备份和恢复一个云硬盘”。

创建云硬盘快照：

1. 在 dashboard 中选择**项目 > Compute > 云硬盘**。
2. 选择目标云硬盘的**创建快照**操作。
3. 为快照输入一个**快照名**，点**创建云硬盘快照**。**云硬盘快照**标签也会显示所有快照。

当一个快照出现在**云硬盘快照**表中，您就可以使用它来克隆新云硬盘。点快照的**创建云硬盘**来克隆云硬盘。如需了解更多相关信息，请参阅 第 4.1.1 节 “创建云硬盘”。

要删除一个快照，选择**删除云硬盘快照**操作。

如果您的 OpenStack 部署使用 Red Hat Ceph 后端，请参阅 第 4.1.8.1 节 “Red Hat Ceph 后端中保护的快照和未保护的快照” 来获取更多相关信息。

4.1.8.1. Red Hat Ceph 后端中保护的快照和未保护的快照

当使用 Red Hat Ceph 作为 OpenStack 部署的后端时，可以把快照在后端设置为*受保护的*。在 OpenStack 中进行删除受保护的快照操作（通过 dashboard 或使用 **`cinder snapshot-delete`** 命令）将会失败。

如果需要删除受保护的快照，您需要先在 Red Hat Ceph 后端把快照设置为*未保护*，然后，就可以使用一般的方法在 OpenStack 中删除这个快照。

如需了解相关信息，请参阅 Protecting a Snapshot 和 Unprotecting a Snapshot。

4.1.9. 把一个云硬盘上传到 Image 服务

您可以把一个存在的云硬盘作为一个镜像直接上传到 Image 服务：

1. 在 dashboard 中选择**项目 > Compute > 云硬盘**。
2. 选择目标云硬盘的**上传到镜像**操作。
3. 为云硬盘提供一个**镜像名称**，并从**磁盘格式**列表中选择一个值。
4. 点**上传**。QEMU 磁盘镜像工具程序会使用您所提供的名称上传您所选择格式的镜像。

要查看上传的镜像，选**项目 > Compute > 镜像**。新的镜像会出现在**镜像**列表中。如需了解更多关于使用和配置镜像的信息，请参阅 第 1.2 节 “管理镜像”。

4.2. 高级卷配置

以下介绍了如何进行高级的卷管理。这些操作需要管理权限。

4.2.1. 备份和恢复一个云硬盘

云硬盘备份就是一个可以持久保存的云硬盘内容的副本。云硬盘备份通常被创建为项存储，在默认情况下，通过 Object Storage 服务进行管理。您也可以为您的备份设置不同的存储技术，OpenStack 支持 Ceph、GlusterFS 和 NFS 作为替代的存储后端。

当创建云硬盘备份时，所有备份的元数据被保存在 Block Storage 服务的数据库中。当 **cinder** 使用备份进行数据恢复时会使用这个元数据。因此，如果数据库已经被破坏，您在恢复云硬盘备份前需要首先恢复 Block Storage 服务的数据库。

如果您只希望配置云硬盘的一部分来使它可以在数据库被破坏的情况下仍然可以工作，您也可以导出备份的元数据。导出元数据后，您就可以在需要时把元数据重新导入到 Block Storage 数据库中。然后，就可以正常地恢复云硬盘。

注意

卷备份和快照有所不同。备份会保存包括在卷中的数据，而快照会保存一个卷在某个时间点上的状态。另外，当一个卷带有快照时，这个卷将无法被删除。卷备份的目的是防止数据丢失，而快照的目的是用于克隆。

因为这个原因，快照后端通常会和存储卷后端在一起，从而可以在进行克隆时最小化延迟的影响。相反的，在一个企业级的环境中，备份存储库通常会存在于不同的位置（例如，在不同的节点上、不同的物理存储上，甚至不同的地理位置）。这可以在存储卷后台被破坏时，备份数据不会受到影响。

如需了解更多与云硬盘快照相关的信息，请参阅 第 4.1.8 节 “创建、克隆或删除云硬盘快照”。

4.2.1.1. 创建一个完全云硬盘备份

使用 **cinder backup-create** 命令可以对云硬盘进行备份。在默认情况下，这个命令会对云硬盘创建一个完全备份。如果这个云硬盘已经存在了备份，则可以创建一个**增量**备份（请参阅 第 4.2.1.2 节 “创建一个增量云硬盘备份”）。

您可以对您有访问权限的云硬盘进行备份。这意味着，有管理权限的用户可以备份任何云硬盘（无论这些云硬盘的所有者是谁）。如需了解更详细的信息，请参阅 第 4.2.1.1.1 节 “以 Admin 身份创建云硬盘备份”。

1. 查看需要备份的云硬盘的 **ID** 或 **Display Name**：

   ```
   # cinder list
   ```

2. 备份云硬盘：

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

   使用要备份的云硬盘的 **ID** 或 **Display Name** 替换 *VOLUME*。例如：

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

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

注意

备份的 **volume_id** 会和被备份的源云硬盘的 **ID** 相同。

3. 运行以下命令检查云硬盘备份是否已经成功完成：

```
# cinder backup-list
```

当备份项的 **Status** 为 **available** 时，意味着云硬盘备份已完成。

现在，您还可以导出并保存云硬盘备份的元数据。这将使您可以在 Block Storage 数据库被破坏的情况下仍然可以恢复云硬盘的备份。运行以下命令：

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

其中的 *BACKUPID* 是云硬盘备份的 ID 或名称。例如：

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

云硬盘备份的元数据包括 **backup_service** 和 **backup_url** 的值。

4.2.1.1.1. 以 Admin 身份创建云硬盘备份

具有管理权限的用户（例如默认的 **admin** 用户）可以备份 OpenStack 管理的任何云硬盘。当 admin 用户备份一个所有者是非 admin 用户的云硬盘时，在默认情况下，所做的备份对云硬盘的所有者不可见。

作为一个 admin 用户，您可以创建一个云硬盘的备份，**并且**使这个备份对特定的租户有效。运行以下命令：

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

其中：

- *TENANTNAME* 是可以访问这个备份的租户的名称。
- *USERNAME* 和 *PASSWD* 是 *TENANTNAME* 内的一个用户的 username/password 信息
- *VOLUME* 是您需要备份的云硬盘的名称或 ID。

- *KEYSTONEURL* 是 Identity 服务的 URL 端点（通常是 http://*IP*:5000/v2，其中的 *IP* 是 Identity 服务主机的 IP 地址）。

在进行这个操作时，备份的大小会占用 *TENANTNAME* 的配额，而不会占用 admin 的配额。

4.2.1.2. 创建一个增量云硬盘备份

在默认情况下，**`cinder backup-create`** 命令会对云硬盘创建一个完全备份。如果这个云硬盘已经存在了备份，则可以创建一个**增量**备份。

因为云硬盘的数据可能会随着使用时间的增加变得非常大，因此定期进行完全备份会消耗大量资源。而增量备份只记录从上一次备份（完全备份或增量备份）以来数据的变化，这样就可以定期记录云硬盘上数据的所有变化，同时最小化对资源的消耗。

使用 **`--incremental`** 选项创建一个增量备份：

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

使用要备份的云硬盘的 **`ID`** 或 **`Display Name`** 替换 *VOLUME*。

注意

当存在增量备份时，不能删除相关的完全备份。例如，当一个完全备份有多个增量备份时，您只能删除最后一个增量备份。

增量备份在 NFS 和 Object Storage 备份存储库中被完全支持。Ceph 备份存储库也支持增量备份，但只针对于存储在 Ceph 后端的存储卷。

4.2.1.3. 当 Block Storage 数据库损坏时恢复云硬盘

Block Storage 数据库包括了云硬盘备份服务（**`openstack-cinder-backup`**）所需的元数据，因此一般情况下，Block Storage 数据库损坏时将无法恢复一个云硬盘备份。元数据包括 **`backup_service`** 和 **`backup_url`** 的值，您可以在创建备份后导出它们（第 4.2.1.1 节 “创建一个完全云硬盘备份”）。

如果您已经导出并保存了元数据，就可以在需要时把它导入到一个新的 Block Storage 数据库，这样就可以对云硬盘备份进行恢复。

1. 使用带有管理员权限的用户运行：

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

   其中的 *backup_service* 和 *backup_url* 包括在您导出的元数据中。例如，使用 第 4.2.1.1 节 “创建一个完全云硬盘备份” 导出的元数据：

   ```
   # cinder --os-volume-api-version 2 backup-import
   cinder.backup.drivers.swift eyJzdGF0dXMi...c2l6ZSI6IDF9
   +----------+----------------------------------------+
   | Property |                 Value                  |
   ```

```
+----------+--------------------------------------+
|    id    | 77951e2f-4aff-4365-8c64-f833802eaa43 |
|   name   |                 None                 |
+----------+--------------------------------------+
```

2. 在元数据被导入到 Block Storage 服务数据库后，您就可以象一般情况一样恢复云硬盘（请参阅 第 4.2.1.4 节 “从备份恢复一个云硬盘”）。

4.2.1.4. 从备份恢复一个云硬盘

1. 找到您需要使用的云硬盘备份的 **ID**：

```
# cinder backup-list
```

Volume ID 需要和您要恢复的云硬盘的 ID 相匹配。

2. 恢复云硬盘备份：

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

其中的 *BACKUP_ID* 是您要使用的云硬盘备份的 ID。

3. 如果您不再需要一个备份，运行以下命令删除它：

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

4.2.1.5. 查看和修改租户的备份配额

和多数租户存储配额（云硬盘数量、云硬盘存储、快照等）不同，备份的配额当前还无法通过 dashboard 进行修改。

当前，备份配额只能通过命令行进行修改（使用 **cinder quota-update** 命令）。

使用以下命令查看某个特定租户（*TENANTNAME*）的存储配额：

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

运行以下命令更新一个特定租户可以创建的最大备份数量（*MAXNUM*）：

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

运行以下命令更新一个特定租户可以创建的所有备份的总量（*MAXGB*）：

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

运行以下命令查看一个特定租户的存储配额的使用情况：

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

4.2.1.6. 通过 Dashboard 启用云硬盘备份管理功能

现在，您可以通过 dashboard 创建、查看、删除和恢复云硬盘备份。使用**项目 > Compute > 云硬盘 > 云硬盘备份**标签页可以进行这些操作。

在默认情况下，**云硬盘备份**标签页没有被启用，您需要根据以下步骤配置 dashboard 来启用它：

1. 打开 **/etc/openstack-dashboard/local_settings** 文件。
2. 找到以下设置：

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

把这个设置改为：

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

3. 通过重启 **httpd** 服务来重启 dashboard：

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

4.2.1.7. 设置一个 NFS 共享作为备份存储库

在默认情况下，Block Storage 服务使用 Object Storage 服务作为备份的存储库。您可以使用以下方法，把 Block Storage 服务配置为使用一个存在的 NFS 共享作为备份存储库：

1. 使用一个带有管理权限的用户登录到提供备份服务（**openstack-cinder-backup**）的节点上。
2. 把 Block Storage 服务配置为使用 NFS 备份驱动（**cinder.backup.drivers.nfs**）：

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

3. 设置您需要作为备份存储库使用的 NFS 共享：

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

其中：

- *NFSHOST* 是 NFS 服务器的 IP 地址或主机名。
- *PATH* 是 NFS 共享在 *NFSHOST* 上的绝对路径。

4. 运行以下命令可以为 NFS 共享设置可选的挂载配置：

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

其中，*NFSMOUNTOPTS* 是由逗号分隔的 NFS 挂载选项（例如，**rw,sync**）。如需了解更多相关信息，请参阅 **nfs** 和 **mount** 的 **man** 页。

5. 重启 Block Storage 备份服务来使所做的改变生效：

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

4.2.1.7.1. 设置不同的备份文件大小

备份服务会使用一个最大的**备份文件大小**设置来限制备份文件的大小，当云硬盘备份的实际大小超过这个值时，备份会被分为多个文件。备份文件大小的默认值是 1.8GB。

运行以下命令可以修改备份文件大小的设置：

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

使用您实际需要的值替换 *SIZE*（以字节为单位）。重启 Block Storage 备份服务使所做的修改生效：

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

4.2.2. 迁移卷

只有管理员可以迁移卷。迁移卷时卷不能被使用，也不能有任何快照。

1. 以一个管理员用户的身份，列出所有有效的卷：

```
# cinder list
```

2. 列出有效的后端（主机）和它们相关的可用域：

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

3. 初始迁移的过程：

```
# cinder migrate VOLUME BACKEND
```

其中：

- **VOLUME** 是要迁移卷的 **ID**。
- **BACKEND** 是卷要被迁移到的后端。

4. 查看迁移卷的当前状态：

```
# cinder show VOLUME
```

例如：

```
# cinder show 45a85c3c-3715-484d-ab5d-745da0e0bd5a
+---------------------------------------+------------------+
|                Property               |      Value       |
```

```
+-----------------------------------------+------------------+
|                   ...                   |       ...        |
|         os-vol-host-attr:host           |      server1     |
|     os-vol-mig-status-attr:migstat      |       None       |
|                   ...                   |       ...        |
+-----------------------------------------+------------------+
```

在迁移过程中，请关注以下属性：

os-vol-host-attr:host

卷当前的后台。当迁移完成后，它的值会变为目标后台（BACKEND 的值）。

os-vol-mig-status-attr:migstat

迁移的状态。当它的值为 **None** 时说明迁移没有在进行。

4.2.3. 使用云硬盘类型进行云硬盘分组设置

OpenStack 允许您创建云硬盘类型，并在创建一个云硬盘时应用云硬盘类型中包括的设置（第 4.1.1 节 “创建云硬盘”）。例如，您可以设置：

- 是否对云硬盘进行加密（第 4.2.5.2 节 “配置云硬盘类型加密”）
- 云硬盘所使用的后端（第 4.1.2 节 “为创建的云硬盘指定后端”）
- Quality-of-Service (QoS) 规格

使用“键-值”对来与云硬盘类型进行关联的设置被称为特别设定（Extra Specs）。当您在创建云硬盘时指定一个云硬盘类型，Block Storage 调度器会应用这些使用“键/值”对指定的设置。您可以为同一个云硬盘类型关联多个“键/值”对。

云硬盘类型可以为不同用户提供不同级别的存储级别。通过使用“键/值”对为云硬盘类型关联相关的设置，您可以为不同的云硬盘类型指定不同的存储级别。然后，您就可以在创建云硬盘时通过指定云硬盘类型来为云硬盘指定相应的存储级别。

注意

被支持的特别设定会根据卷驱动的不同而有所不同。请参考您的卷驱动的相关文档来获得详细信息。

4.2.3.1. 创建并配置一个云硬盘类型

1. 使用一个 admin 用户，在 dashboard 中选择**管理员 > 云硬盘 > 云硬盘类型**。
2. 点**创建云硬盘类型**。
3. 在**名称**项中输入云硬盘类型的名称。
4. 点**创建云硬盘类型**。新类型会出现在**云硬盘类型**项中。
5. 选择云硬盘类型的**查看特别设定**操作。
6. 点**创建**，设置**键**和**值**。所指定的键/值对必须有效，否则在创建云硬盘时指定云硬盘类型会出现错误。

7. 点**创建**。相关联的设置（“键/值”对）会出现在**特别设定**列表中。

在默认情况下，所有的云硬盘类型对于所有 OpenStack 租户都有效。如果您创建的云硬盘类型需要对租户访问进行限制时，则需要使用 CLI 进行设置。如需了解相关信息，请参阅 第 4.2.3.4 节 “创建和配置私有云硬盘类型”。

注意

您也可以为云硬盘类型关联一个 QOS 规格。如需了解详细信息，请参阅 第 4.2.4.2 节 “为云硬盘类型关联一个 QOS 规格”。

4.2.3.2. 编辑一个云硬盘类型

1. 使用一个 admin 用户，在 dashboard 中选择**管理员 > 云硬盘 > 云硬盘类型**。
2. 在**云硬盘类型**项中，选择云硬盘类型的**查看特定设置**操作。
3. 在**卷扩展规格属性**列表中，您可以：
   - 为云硬盘类型添加新设置。点**创建**，指定需要和这个云硬盘类型相关联的“键/值”对。
   - 编辑一个存在的设置。选择**编辑**操作。
   - 删除已存在的设置。点**删除额外规格**按钮。

4.2.3.3. 删除一个云硬盘类型

要删除一个云硬盘类型，在**云硬盘类型**列表中选择相应的云硬盘并点**删除云硬盘类型**。

4.2.3.4. 创建和配置私有云硬盘类型

在默认情况下，所有的云硬盘类型对于所有 OpenStack 租户都有效。您可以在创建云硬盘类型的过程中修改这个设置，把所创建的云硬盘类型设置为**私有**（把它的 **`Is_Public`** 标记设置为 **`False`**）。

通过设置私有云硬盘类型，可以限制对特定云硬盘类型的访问。一般情况下，某些设置（例如，需要进行测试的新后端或超高性能配置）应该只可以被特定租户使用。

运行以下命令创建一个私有云硬盘类型：

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

+ 使用私有云硬盘类型的名称替换 *VTYPE*。

在默认情况下，私有云硬盘类型只能被它的创建者访问。admin 用户可以使用以下命令找到并查看私有云硬盘类型：

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

这个命令会列出公共云硬盘类型和私有云硬盘类型，并列出它们的名称和 ID。您需要使用云硬盘类型的 ID 对它们进行访问。

分配私有云硬盘类型的访问权限是在租户一级进行的。运行以下命令可以为一个租户分配访问一个私有云硬盘类型的权限：

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

其中：

- *VTYPEID* 是私有云硬盘类型的 ID。
- *TENANTID* 是被分配可以访问 *VTYPEID* 权限的项目/租户 ID。

运行以下命令可以查看哪些租户可以访问某个私有云硬盘类型：

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

运行以下命令可以把一个租户从一个私有云硬盘的访问列表中删除：

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

注意

在默认情况下，只有具有管理权限的用户才可以创建、查看或配置私有云硬盘类型的访问权限。

4.2.4. 使用服务质量规格

您可以把多个性能设置包括在一个服务质量规格（ Quality-of-Service specification，简称 QOS Specs）中，这样，就可以为不同类型的用户提供相应服务质量的性能。

和卷类型相似，性能设置通过一个“关键字/值”数据对来映射到规格。服务质量规格和卷类型的不同之处在于：

- 服务质量规格被用来实现性能设置，包括对磁盘读/写的限制。不同的存储驱动会支持不同性能设置。

  如需了解您的后端所支持的 QOS 规格，请参阅您的后端设备的卷驱动文档。

- 卷类型会直接应用到卷，而 QOS 规格会与卷类型相关联。在创建卷时，指定卷类型的同时也会应用和这个卷类型相关联的 QOS 规格中定义的性能设置。

4.2.4.1. 创建并配置一个 QOS 规格

作为管理员，您可以通过 QOS 规格列表创建并配置 QOS 规格。您可以为同一个 QOS 规格管理多个“键/值”对。

1. 使用一个 admin 用户，在 dashboard 中选择**管理员 > 云硬盘 > 云硬盘类型**。
2. 在 **QOS 规格**项中点**创建 QOS 规格**。
3. 为 **QOS 规格** 输入一个名称。

4. 在**消费者**项中指定 QOS 规格在哪里被强制执行：

   表 4.1. 消费者类型

类型	描述
back-end	QOS 规格需要应用到 Block Storage 后端。
front-end	QOS 规格需要应用到 Compute。
both	QOS 规格将应用到 Block Storage 和 Compute。

5. 点**创建**。新创建的 QOS 规格应该出现在 **QOS 规格**列表中。
6. 在 **QOS 规格**列表中，选择新规格的**管理规格**操作。
7. 点**创建**，设置**键**和**值**。所指定的键/值对必须有效，否则在创建云硬盘时，指定和这个 QOS 规格相关联的云硬盘类型时会出现错误。
8. 点**创建**。相关联的设置（“键/值”对）会出现在**键值对**列表中。

4.2.4.2. 为云硬盘类型关联一个 QOS 规格

作为一个管理员，您可以使用**云硬盘类型**列表为一个存在的卷类型关联 QOS 规格。

1. 使用一个 admin 用户，在 dashboard 中选择**管理员 > 云硬盘 > 云硬盘类型**。
2. 在**云硬盘类型**项中，选择云硬盘类型的**管理 QOS 规格关联**操作。
3. 从**要关联的 QOS 规格**下拉列表中选择一个 QOS 规格。
4. 点**关联**。所选的 QOS 规格将会出现在被编辑的云硬盘类型的**已关联 QOS 规格**栏中。

4.2.4.3. 为云硬盘类型取消关联一个 QOS 规格

1. 使用一个 admin 用户，在 dashboard 中选择**管理员 > 云硬盘 > 云硬盘类型**。
2. 在**云硬盘类型**项中，选择云硬盘类型的**管理 QOS 规格关联**操作。
3. 从要关联的 QOS 规格列表中选择 **None**。
4. 点**关联**。所选的 QOS 规格将会不会出现在被编辑的云硬盘类型的**已关联 QOS 规格**栏中。

4.2.5. 使用静态密钥加密卷

对卷进行加密可以在卷后台的安全性被破坏时为数据提供基本的安全保护。被加密卷的内容只有在使用一个特定密钥时才可以被读写；Compute 和 Block Storage 服务都需要被配置为使用相同的密钥来使实例可以使用加密的卷。以下介绍了如何配置 OpenStack 部署来使用一个密钥来加密卷。

4.2.5.1. 创建一个静态密钥

实现基本卷加密的第一步是设置一个*静态密钥（static key）*。这个密钥需要是一个十六进制的字符串，并被 Block Storage 卷服务（**openstack-cinder-volume**）和所有 Compute 服务（**openstack-nova-compute**）使用。要设置这两个服务使用这个密钥，在它们的相关配置文件中的 **[keymgr]** 项中使用这个密钥作为 **fixed_key** 的值。

1. 通过命令行，以 **root** 身份登录到包括 **openstack-cinder-volume** 的节点。
2. 设置静态密钥：

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

把 **HEX_KEY** 替换为一个 16 位的十六进制密钥值（例如 **00000000000000000000000000000000000000000000000000000000000000000**）。

3. 重启 Block Storage 卷服务：

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

4. 登录到包括 **openstack-nova-compute** 的节点，设置相同的静态密钥：

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

注意

如果您有多个 Compute 节点（多个节点都包括 **openstack-nova-compute**），您需要在每个节点的 **/etc/nova/nova.conf** 中设置相同的静态密钥。

5. 重启 Compute 服务：

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

注意

同样，如果您在多个 Compute 节点上设置了静态密钥，您需要在每个节点上都重启 **openstack-nova-compute** 服务。

到目前为止，Compute 和 Block Storage 卷服务都可以使用相同的静态密钥对卷进行加密和解密。新的实例将可以使用经过静态密钥（**HEX_KEY**）加密的卷。

4.2.5.2. 配置云硬盘类型加密

为了使用 第 4.2.5.1 节 “创建一个静态密钥” 创建的静态密钥对云硬盘进行加密，您需要一个*加密的云硬盘类型*。配置一个加密的云硬盘类型包括设置 provider class、cipher 和 key_size。运行以下命令：

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

其中：

- **BITSIZE** 是密钥的大小（例如，**512** 代表 512 位的密钥）。
- **VOLTYPE** 是您需要加密的云硬盘类型的名称。

这个命令设置了 **nova.volume.encryptors.luks.LuksEncryptor** provider class 和 **aes-xts-plain64** cipher。在这个发行版本中，云硬盘加密只支持这个 class/cipher 配置。

在您创建了一个加密的云硬盘类型后，您就可以在创建加密云硬盘时自动使用它。在**创建云硬盘**窗口中的类型下拉菜单中选择需要的加密云硬盘类型（请参阅 第 4.1 节 “基本云硬盘（Volume）的使用和配置”）。

4.2.6. 配置如何在多后端间分配卷

如果 Block Storage 服务被配置为使用多后端，您可以使用配置的云硬盘类型来指定卷可以在哪里创建。如需更详细的信息，请参阅 第 4.1.2 节 “为创建的云硬盘指定后端”。

如果在创建云硬盘的过程中没有进行指定，Block Storage 服务会自动选择一个后端。Block Storage 把第一个定义的后端作为默认的后端，在这个后端还有空间时，它会被使用。如果第一个后端没有可用的空间，Block Storage 会把第 2 个后端设置为默认后端。以此类推。

如果这无法满足您的要求，则可以使用过滤调度程序来控制 Block Storage 如何选择后端。调度程序会根据不同的过滤规则选择适当的后端，例如：

AvailabilityZoneFilter
: 过滤掉不符合所需卷的可用域要求的后端。

CapacityFilter
: 只选择有足够空间的后端来保存云硬盘

CapabilitiesFilter
: 只选择可以支持卷中的所有指定设置的后端

按照以下步骤配置过滤器：

1. 启用 **FilterScheduler**。

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

2. 设置哪个过滤器应该被激活：

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

3. 配置调度程序如何选择适当的后端。如果您需要调度程序：

- 总选择具有最多可用空间的后端，运行：

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

- 在适当后端中随机选择，运行：

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

4. 重启 Block Storage 调度程序来使您的改变生效：

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

第 5 章 管理容器

OpenStack Object Storage (swift) 使用容器（container）来保存它的对象（数据），容器和一个文件系统的目录相似，只是不能进行嵌套。它为用户提供了一个存储无结构数据的方法。例如，对象可以包括图形、文本文件或镜像。存储的对象没有被加密，也不被压缩。

为了更好地组织数据，可以使用虚拟文件夹（pseudo-folder）。虚拟文件夹是包括对象的逻辑设备，并可以进行嵌套。例如，您可以在存储照片的容器中创建一个 *Images* 文件夹；在存储视频的容器中创建一个 *Media* 文件夹。

您可以在每个项目中创建一个或多个容器，在每个容器中创建一个或多个对象或虚拟文件夹。

5.1. 创建一个容器

1. 在 dashboard 中选择**项目 > 对象存储 > 容器**。
2. 点**创建容器**。
3. 指定**容器名**，在**容器访问**项中选择以下值之一：

类型	描述
Private	限制当前的项目只能被某个用户访问。
Public	允许 API 访问所有具有公共 URL 的容器。但是在 dashboard 中，项目用户无法看到其它项目的公共容器和数据。

4. 点**创建容器**。

5.2. 为容器创建虚拟文件夹

1. 在 dashboard 中选择**项目 > 对象存储 > 容器**。
2. 点您需要添加虚拟文件夹的容器名称。
3. 点**创建虚拟文件夹**。
4. 在**虚拟文件夹名称**项中输入名称，点**创建**。

5.3. 上传一个对象

如果您没有上传一个实际的文件，对象仍然会被创建（作为一个占位符），它可以在以后被用来上传文件。

1. 在 dashboard 中选择**项目 > 对象存储 > 容器**。
2. 点放置上传对象的容器的名称；如果容器中存在虚拟文件夹，您可以点它的名称。

3. 找到您的文件，点**上传对象**。
4. 在**对象名**项中输入一个名称：
   - 虚拟文件夹的名称可以使用 /（如 *Images/myImage.jpg*）。如果指定的文件夹不存在，在对象上传时会被自动创建。
   - 如果您使用的名称不是唯一的（同名的对象已经存在），它会覆盖以前同名对象的内容。
5. 点**上传对象**。

5.4. 复制一个对象

1. 在 dashboard 中选择**项目 > 对象存储 > 容器**。
2. 点对象容器或文件夹的名称来显示对象。
3. 点**上传对象**。
4. 找到需要复制的文件，在菜单中选**复制**。
5. 指定下列值：

项	描述
目标容器	新对象的目标容器。
路径	目标容器中的虚拟文件夹；如果文件夹不存在，会创建它。
目标对象名	新对象的名称。如果这个名称不是唯一的（同名的对象已经存在），它会覆盖以前同名对象的内容。

6. 点**复制对象**。

5.5. 删除对象

1. 在 dashboard 中选择**项目 > 对象存储 > 容器**。
2. 找到对象，在菜单中选择**删除对象**。
3. 点**删除对象**来确认删除操作。

5.6. 删除容器

1. 在 dashboard 中选择**项目 > 对象存储 > 容器**。

2. 在**容器**项中找到需要删除的容器，确认其中的对象都已经被删除（请参阅 第 5.5 节 “删除对象”）。
3. 在容器的菜单中选择**删除容器**。
4. 点**删除容器**来确认删除操作。

第 6 章 配置 OPENSTACK 使用 NFS 后端

本章介绍了如何配置 OpenStack 卷服务（**openstack-cinder-volume**）来使用一个存在的 NFS 服务器作为一个额外的后端。另外，本章还介绍了如何创建一个云硬盘类型，从而可以在创建云硬盘时使用它来设置 NFS 共享作为后端。

先决条件：

- 要作为后端的 NFS 共享已被正确配置。
- 提供 OpenStack 卷服务的节点有到 NFS 共享的读/写访问权限。
- 具有运行 OpenStack 卷服务的系统的 **root** 访问权限。

基于的假设：

- 您的 OpenStack 环境不是通过 Red Hat Enterprise Linux OpenStack Platform Installer 进行部署的。
- 您的 OpenStack Block Storage 服务使用默认的后端（通过 Packstack 部署的、名为 **lvm** 的后端）。

6.1. 配置 SELINUX

当客户端启用了 SELinux 时，如果需要从这个客户端访问一个实例上的 NFS 卷，则需要启用 virt_use_nfs 布尔值。以 **root** 身份运行以下命令来启用这个布尔值（并使它在重启后仍然有效）：

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

在需要访问实例上的 NFS 共享的所有客户端主机上运行这个命令（包括所有 Compute 节点）。

6.2. 配置共享

添加 NFS 后端的第一步是定义 OpenStack 卷服务要使用的 NFS 共享。根据以下步骤进行操作：

1. 以 root 身份登录到运行 OpenStack 卷服务的节点上。
2. 在 **/etc/cinder/** 目录中创建一个新的、名为 nfs_share 的文本文件：

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

3. 在 **/etc/cinder/nfs_share** 中使用以下格式定义 NFS 共享：

   ```
   HOST:SHARE
   ```

   其中：

   - HOST 是 NFS 服务器的 IP 地址或主机名。
   - SHARE 是 HOST 输出的 NFS 共享的绝对路径。

4. 把 **/etc/cinder/nfs_share** 的所有者设置为 root 用户和 cinder 组：

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

5. 最后，把 **/etc/cinder/nfs_share** 设置为 cinder 组中的用户可以进行读访问：

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

6.3. 创建一个新的后端定义

在默认情况下，Packstack 在 **/etc/cinder/cinder.conf** 中为 LVM 定义一个后端：

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

在 **/etc/cinder/cinder.conf** 中定义了一个 NFS 共享后，可以按照以下步骤为它配置一个额外的后端：

1. 以 **root** 身份登录到运行 OpenStack 卷服务的节点上。
2. 为 NFS 后端创建一个新定义，把卷服务设置为使用定义了 NFS 共享的文件（**/etc/cinder/nfs_share**）：

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

   在这里，我们使用 **nfsbackend** 作为一个定义的名称。

3. 配置卷服务使用 NFS 卷驱动（名为 **cinder.volume.drivers.nfs.NfsDriver**）：

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

4. 为 NFS 后端定义一个卷后端名（在以下命令中是 **nfs**）：

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

5. 在 nfs_mount_options 配置关键字中添加需要的挂载选项（**MOUNTOPTIONS**）：

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

现在，以下项应该出现在 **/etc/cinder/cinder.conf** 中：

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

现在，您可以启用 NFS 后端（使用 **/etc/cinder/cinder.conf** 中的 **enabled_backends** 配置关键字）。Packstack 创建的默认后端应该已在这里列出：

```
enabled_backends=lvm
```

在这个列表中添加新的 NFS 后端定义：

```
enabled_backends=lvm,nfs
```

当 NFS 后端被启用后，重启 OpenStack 卷服务：

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

6.4. 为 NFS 后端创建一个云硬盘类型

现在，新创建的 NFS 后端已经有效，但在创建新的云硬盘时还不能使用。为了使新的云硬盘可以使用这个 NFS 后端，您需要首先为它创建一个*云硬盘类型*。

1. 查看存在的云硬盘类型。在默认情况下，使用 lvm 后端的云硬盘类型应该已经存在（名为 iscsi）：

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

2. 为 NFS 后端创建一个新的、名为 **nfstype** 的云硬盘类型：

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

3. 使用后端的名称（**nfsbackend**）把 **nfstype** 云硬盘类型配置为使用 NFS 后端：

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

4. 检查新创建的云硬盘类型被正确创建并配置：

```
+--------------------------------------+---------+
|                  ID                  |   Name  |
+--------------------------------------+---------+
| bbff44b5-52b1-43d6-beb4-83aa2d20bc59 | nfstype |
| f8d31dc8-a20e-410c-81bf-6b0a971c61a0 |  iscsi  |
+--------------------------------------+---------+
+----------------------+-------+----------------------------
-----------+
|           ID         |  Name |               extra_specs
|
+----------------------+-------+----------------------------
-----------+
|bbff44b5-~-83aa2d20bc59|nfstype|{u'volume_backend_name':
u'nfsbackend'}|
```

```
|f8d31dc8-~-6b0a971c61a0| iscsi |    {u'volume_backend_name':
u'lvm'}   |
+----------------------+-------+--------------------------
-----------+
```

注意

您可以通过 dashboard 创建并配置云硬盘类型。如需了解更多信息，请参阅 第 4.2.3 节 “使用云硬盘类型进行云硬盘分组设置”。

6.5. 测试新的 NFS 后端

为了测试新的 NFS 后端，在创建一个名为 **nfsvolume** 的云硬盘时调用云硬盘类型 **nfstype**：

```
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2015-01-06T05:14:09.271114      |
| display_description |                 None                 |
|     display_name    |              nfsvolume               |
|      encrypted      |                False                 |
|          id         | 0cd7ac45-622a-47b0-9503-7025bbedc8ed |
|       metadata      |                  {}                  |
|         size        |                  1                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |               creating               |
|     volume_type     |               nfstype                |
+---------------------+--------------------------------------+
```

当云硬盘被成功创建后，在 NFS 服务器上检查 NFS 共享。一个相应的云硬盘（它的名字中包括新创建云硬盘的 ID）应该在那里出现：

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

第 7 章 配置带有 NUMA 的 CPU 固定（CPU PINNING）

本章介绍了与 NUMA 拓扑相关的信息，以及如何在支持 NUMA 的系统上配置 OpenStack 环境。在这类设置中，虚拟机实例被固定到专用的 CPU 内核，从而可以通过更加智能的调度来提高客户机的性能。

如需了解 NUMA 的背景信息，请参阅 What is NUMA and how does it work on Linux ?。如需了解使用 libvirt 对 NUMA 进行性能调优的信息，请参阅 Virtualization Tuning and Optimization Guide。

7.1. COMPUTE 节点的配置

虽然具体的配置取决于您的主机系统的 NUMA 拓扑，但是，您必须为主机进程在所有 NUMA 节点上保留一些 CPU 内核，并让剩余的 CPU 内核处理客户虚拟机实例。例如，有 8 个 CPU 内核平均分布在 2 个 NUMA 节点上，它的结构可以和以下类似：

NUMA 拓扑示例

```
┌──────────────────┬──────────────────┬──────────────────┐
│                  │ Node 0           │ Node 1           │
├──────────────────┼─────────┬────────┼─────────┬────────┤
│ Host processes   │ Core 0  │ Core 1 │ Core 4  │ Core 5 │
├──────────────────┼─────────┼────────┼─────────┼────────┤
│ Guest processes  │ Core 2  │ Core 3 │ Core 6  │ Core 7 │
└──────────────────┴─────────┴────────┴─────────┴────────┘
```

注意

为了决定为主机进程保留的内核数量，需要在特定工作负载的情况下对主机性能进行观察。

配置 Compute 节点包括以下几个步骤：

1. 把 **/etc/nova/nova.conf** 文件中的 **vcpu_pin_set** 选项设置成为客户系统进程保留的 CPU 内核列表。对于上面的示例，您可以进行以下设置：

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

2. 把相同文件中的 **reserved_host_memory_mb** 选项设置成为主机进程保留的 RAM 的数量。如果您需要保留 512 MB 内存，配置如下：

   ```
   reserved_host_memory_mb=512
   ```

3. 运行以下命令在 Compute 节点上重启 Compute 服务：

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

4. 在引导选项中，使用 *isolcpus* 并加上 CPU 列表。这可以把主机进程限制为只能访问列出的 CPU 内核。对于以上的示例，您可以运行：

   -

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

5. 更新引导记录使这个改变生效：

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

使用包括引导记录的设备名替换 *device*，这个设备名通常是 *sda*。

6. 重启系统。

7.2. 调度程序配置

1. 在运行 OpenStack Compute Scheduler 的每个系统上编辑 **/etc/nova/nova.conf** 文件。找到 **scheduler_default_filters** 选项，如果它被注释掉，取消它的注释，在过滤器列表中加入 **AggregateInstanceExtraSpecFilter** 和 **NUMATopologyFilter**。这个行的内容应该和以下类似：

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

2. 重启 openstack-nova-scheduler 服务：

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

7.3. 集合（AGGREGATE）和 FLAVOR 配置

为了把您的 OpenStack 环境中的虚拟机固定在特定资源上运行，在一个系统的 Compute 命令行接口中执行以下步骤：

1. 加载 **admin** 身份信息：

```
source ~/keystonerc_admin
```

2. 创建一个可以接收固定请求的主机集合：

```
nova aggregate-create name
```

使用适当的名称（如 *performance* 或 *cpu_pinning*）替换 *name*。

3. 编辑集合的元数据来启用固定功能：

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

在这个命令中，数字 *1* 匹配在前一步中创建的集合 ID。

4. 为其它主机创建一个集合：

```
nova aggregate-create name
```

使用适当的名称（如 *normal*）替换 *name*。

5. 为这个集合编辑元数据：

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

其中，使用的数字是 *2*，这是因为它在 *1*（第一个集合的 ID）之后。

6. 修改存在的 flavors 设置：

```
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. 为可以接收固定请求的主机创建一个 flavor：

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

使用适当的名称（如 *m1.small.performance* 或 *pinned.small*）替换 *name*；使用新 flavor 的 ID 替换 *ID*（如果您有 5 个标准的 flavor，使用 **6**；如果您需要 **nova** 来产生 UUID，使用 **auto**）；使用需要的内存数量（以 MB 为单位）替换 *RAM*；使用需要的磁盘大小（以 GB 为单位）替换 *disk*；使用需要保留的虚拟 CPU 数量替换 *vCPUs*。

8. 把 flavor 的 **hw:cpu_policy** 设置为 **dedicated**，这将启用 CPU 固定功能：

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

使用在前一步中创建的 flavor 的 ID 替换 *ID*。

9. 把 **aggregate_instance_extra_specs:pinned** 设置为 *true*，从而确保基于这个 flavor 的实例在它们的元数据中有这个设置：

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

使用 flavor 的 ID 替换 *ID*。

10. 为新的集合添加以下主机：

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

使用第一个（"performance"/"pinning"）集合的 ID 替换 *ID_1*；把 *host_1* 替换为需要被加入到这个集合中的主机名。

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

使用第 2 个（"normal"）集合的 ID 替换 *ID_2*；把 *host_2* 替换为需要被加入到这个集合中的主机名。

运行以下命令使用新 flavor 引导一个实例：

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

把 *image* 替换为保存的 VM 镜像名（请参阅 **nova image-list**）；把 *flavor* 替换为实际的 flavor 名（*m1.small.performance*、*pinned.small* 或任何您使用的名称）；把 *server_name* 替换为新服务器的名称。

为了检查新服务器被正确配置，运行以下命令，在输出中检查 **OS-EXT-SRV-ATTR:hypervisor_hostname** 的值：

```
nova show server_name
```

附录 A. 镜像配置参数

在 **glance image-update** 和 **glance image-create** 命令中，**property** 选项可以使用以下关键字.

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

注意

通过镜像属性进行的配置会覆盖通过 flavor 进行的配置。如需了解更多信息，请参阅 Manage Flavors。

表 A.1. 属性关键字

针对于	键	描述	支持的值
所有	architecture	hypervisor 支持的 CPU 架构。例如，**x86_64**、**arm** 或 **ppc64**。运行 **uname -m** 来获得一个机器的架构信息。我们强烈推荐使用 libosinfo project 中定义的架构数据名称。	• 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

针对于	键	描述	支持的值
			ppcemb-PowerPC (Embedded 32-bit) s390-IBM Enterprise Systems Architecture/390 s390x-S/390 64-bit sh4-SuperH SH-4 (Little Endian) sh4eb-SuperH SH-4 (Big Endian) sparc-Scalable Processor Architecture, 32-bit sparc64-Scalable Processor Architecture, 64-bit unicore32-Microprocessor Research and Development Center RISC Unicore32 x86_64-64-bit extension of IA-32 xtensa-Tensilica Xtensa configurable microprocessor core xtensaeb-Tensilica Xtensa configurable microprocessor core (Big Endian)
所有	hypervisor_type	hypervisor 的类型。	**`kvm`**、**`vmware`**
所有	instance_uuid	对于快照镜像，这个值就是创建这个镜像的服务器的 UUID。	有效的服务器 UUID
所有	kernel_id	在引导一个 AMI 镜像时，作为内核使用的、存储于Image 服务中的镜像 ID。	有效的镜像 ID

针对于	键	描述	支持的值
所有	os_distro	操作系统的通用名称（使用小写）。请使用 libosinfo project 中定义的名称。在这个项中只指定可以被识别的名称，过时的名称同时被列在这里来帮助您找到可识别的名称。	• arch-Arch Linux。不要使用 archlinux 或 org.archlinux • centos-Community Enterprise Operating System。不要使用 org.centos 或 CentOS • debian-Debian。不要使用 Debian 或 org.debian • fedora-Fedora。不要使用 Fedora、org.fedora 或 org.fedoraproject • freebsd-FreeBSD。不要使用 org.freebsd、freeBSD 或 FreeBSD • gentoo-Gentoo Linux。不要使用 Gentoo 或 org.gentoo • mandrake-Mandrakelinux (MandrakeSoft) 。不要使用 mandrakelinux 或 MandrakeLinux • mandriva-Mandriva Linux。不要使用 mandrivalinux • mes-Mandriva Enterprise Server。不要使用 mandrivaent 或 mandrivaES • msdos-Microsoft Disc Operating System。不要使用 ms-dos • netbsd-NetBSD。不要使用 NetBSD 或 org.netbsd • netware-Novell NetWare。不要使用 novell 或 NetWare • openbsd-OpenBSD。不要使用 OpenBSD 或 org.openbsd • opensolaris-OpenSolaris。不要使用 OpenSolaris 或 org.opensolaris • opensuse-openSUSE。不要使用 suse、SuSE 或 org.opensuse • rhel-Red Hat Enterprise Linux。不要使用 redhat、RedHat 或 com.redhat • sled-SUSE Linux Enterprise Desktop。不要使用 com.suse • ubuntu-Ubuntu。不要使用 Ubuntu、com.ubuntu、org.ubuntu 或 canonical • windows-Microsoft Windows。不要使用 com.microsoft.server
所有	os_version	发行厂商指定的操作系统版本。	版本号（例如，"11.10"）

针对于	键	描述	支持的值
所有	ramdisk_id	在引导一个 AMI 镜像时，作为 ramdisk 使用的、存储于 Image 服务中的镜像 ID。	有效的镜像 ID
所有	vm_mode	虚拟机模式。它代表了为虚拟机使用的主机/客户机 ABI（application binary interface）。	**hvm** - 完全虚拟化。QEMU 和 KVM 使用这个模式。
libvirt API 驱动	hw_disk_bus	指定附加到磁盘设备的磁盘控制器类型。	**scsi**、**virtio**、**ide** 或 **usb**。
libvirt API 驱动	hw_numa_nodes	对实例有效的 NUMA 节点的数量（不会覆盖 flavor 的定义）。	整数。关于 NUMA 拓扑定义的详情，请参阅添加元数据中关于 hw:NUMA_def 关键字的介绍。
libvirt API 驱动	hw_numa_mempolicy	NUMA 内存分配策略（不会覆盖 flavor 的定义）。	strict - 强制实例的内存分配来自于和它绑定的 NUMA 节点（如果设置了 numa_nodes，这是默认值）。preferred - 内核可以使用其它节点。当 'hw:numa_nodes' 被设置为 '1' 时会有用。
libvirt API 驱动	hw_numa_cpus.0	vCPU N-M 到 NUMA 节点 0 的映射（不会覆盖 flavor 的定义）。	用逗号隔开的整数列表
libvirt API 驱动	hw_numa_cpus.1	vCPU N-M 到 NUMA 节点 1 的映射（不会覆盖 flavor 的定义）。	用逗号隔开的整数列表

针对于	键	描述	支持的值
libvirt API 驱动	hw_numa_mem.0	把 N GB 内存映射到 NUMA 节点 0（不会覆盖 flavor 的定义）。	整数
libvirt API 驱动	hw_numa_mem.1	把 N GB 内存映射到 NUMA 节点 1（不会覆盖 flavor 的定义）。	整数
libvirt API 驱动	hw_qemu_guest_agent	guest agent 支持。如果设为 **yes**，**qemu-ga** 也被安装，文件系统就可以被静默（quiesced 或称为 frozen），并自动创建快照。	**yes / no**
libvirt API 驱动	hw_rng_model	为镜像实例添加的一个随机数生成器。云管理员可以通过配置实例的 flavor 来启用并控制设备的行为。在默认情况下： - 随机数生成器设备被禁用。 - /dev/random 被作为默认的熵池源。要指定一个物理的 HW RNG 设备，在 nova.conf 文件中使用以下选项：rng_dev_path=/dev/hwrng	**virtio**，或其它支持的设备。

针对于	键	描述	支持的值
libvirt API 驱动	hw_scsi_model	启用 VirtIO SCSI（virtio-scsi）来为 compute 实例提供块设备访问。在默认情况下，实例使用 VirtIO Block（virtio-blk）。VirtIO SCSI 是一个半虚拟化的 SCSI 控制器设备，它可以提供更高的可扩展性和性能，并支持高级 SCSI 硬件。	**virtio-scsi**
libvirt API 驱动	hw_video_model	使用的视频驱动	**vga**、**cirrus**、**vmvga**、**xen** 或 **qxl**
libvirt API 驱动	hw_video_ram	视频影像的最大内存数量。它只有当 **hw_video:ram_max_mb** 在 flavor 的 **extra_specs** 中设置，而且它的值大于 **hw_video_ram** 中设置的值时才有效。	整数（以 MB 为单位，如 '64'）

针对于	键	描述	支持的值
libvirt API 驱动	hw_watchdog_action	启用一个虚拟硬件 watchdog 设备，它会在服务器停止工作时执行指定的任务。watchdog 使用 i6300esb 设备（仿真一个 PCI Intel 6300ESB）。如果 **hw_watchdog_action** 没有指定，watchdog 将被禁用。	• disabled - 设备没有被附加。即使已经使用镜像的 flavor 启用了 watchdog，用户仍然可以为这个镜像禁用 watchdog。这个参数的默认值是 disabled。 • reset - 强制重置客户机。 • poweroff - 强制关闭客户机。 • pause - 暂停客户机。 • none - 只启用 watchdog；当服务器停止工作时不进行任何操作。
libvirt API 驱动	os_command_line	libvirt 驱动使用的内核命令行（替代默认值）。对于 Linux Containers（LXC），这个值被用作初始化参数。这个关键字只对 Amazon 内核、ramdisk 和机器镜像（aki、ari 和 ami）有效。	
libvirt API 驱动和 VMware API 驱动	hw_vif_model	指定要使用的虚拟网络接口设备的型号。	有效值取决于配置的 hypervisor。 • KVM 和 QEMU：e1000、ne2k_pci、pcnet、rtl8139 和 virtio。 • VMware：e1000、e1000e、VirtualE1000、VirtualE1000e、VirtualPCNet32、VirtualSriovEthernetCard 和 VirtualVmxnet。 • Xen：e1000、netfront、ne2k_pci、pcnet 和 rtl8139。

针对于	键	描述	支持的值
VMware API 驱动	vmware_adaptertype	hypervisor 使用的虚拟 SCSI 或 IDE 控制器。	**lsiLogic**、**busLogic** 或 **ide**
VMware API 驱动	vmware_ostype	一个 VMware GuestID，它描述了安装在镜像中的操作系统。当创建虚拟机时，这个值被传递给 hypervisor。如果没有指定，这个关键字的值被默认设为 **otherGuest**。	请参阅 thinkvirt.com。
VMware API 驱动	vmware_image_version	当前没有使用。	**1**
XenAPI 驱动	auto_disk_config	如果设为 true，磁盘的 root 分区会在实例引导前自动重新调整它的大小。这个值只在 Compute 服务使用一个带有 XenAPI 启动的、基于 Xen 的 hypervisor 时有效。Compute 服务只在镜像中包括一个分区，而且分区是 **ext3** 或 **ext4** 格式的情况下才会尝试重新调整分区大小。	**true / false**

针对于	键	描述	支持的值
XenAPI 驱动	`os_type`	在镜像中安装的操作系统。XenAPI 驱动中所包括的逻辑代码会根据镜像的 **`os_type`** 参数值的不同执行不同的操作。例如，设置为 **`os_type=windows`** 的镜像，它会创建一个基于 FAT32 的交换分区（swap partition），而不会创建 Linux 交换分区，它会限制其中的主机名的长度要小于 16 个字符。	**`linux`** 或 **`windows`**