Red Hat OpenStack Platform 17.1

创建和管理镜像

使用镜像服务(glance)在 Red Hat OpenStack Platform 中创建和管理镜像。

Last Updated: 2024-06-12

Red Hat OpenStack Platform 17.1 创建和管理镜像

使用镜像服务(glance)在 Red Hat OpenStack Platform 中创建和管理镜像。

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

本指南提供了创建和管理镜像的步骤，以及配置镜像服务(glance)的步骤。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。与我们分享您的成功秘诀。

在 JIRA 中提供文档反馈

使用 Create Issue 表单对文档提供反馈。JIRA 问题将在 Red Hat OpenStack Platform Jira 项目中创建，您可以在其中跟踪您的反馈进度。

1. 确保您已登录到 JIRA。如果您没有 JIRA 帐户，请创建一个帐户来提交反馈。
2. 点击以下链接打开 **Create Issue** 页面： Create Issue
3. 完成 **Summary** 和 **Description** 字段。在 **Description** 字段中，包含文档 URL、章节或章节号以及问题的详细描述。不要修改表单中的任何其他字段。
4. 点 **Create**。

第 1 章 IMAGE 服务(GLANCE)

镜像服务(glance)为磁盘和服务器镜像提供发现、注册和交付服务。它提供了复制或快照服务器镜像的功能，并立即存储它。您可以将存储的镜像用作模板，比安装服务器操作系统和单独配置服务，快速、一致地委托新的服务器。

1.1. 虚拟机镜像格式

虚拟机(VM)镜像是一个包含虚拟磁盘的文件，它安装了可引导的操作系统。VM 镜像采用不同的格式支持。Red Hat OpenStack Platform (RHOSP)提供了以下格式：

- **RAW** - 不结构化的磁盘镜像格式。
- **QCOW2** - QEMU 模拟器支持的磁盘格式。此格式包括 QCOW2v3 （有时称为 QCOW3），这需要 QEMU 1.1 或更高版本。
- **ISO** - 保存在磁盘上的数据的按扇区副本，存储在二进制文件中。
- **AKI** - 表示 Amazon 内核镜像。
- **ami** - 表示 Amazon Machine 镜像。
- **ARI** - 表示 Amazon RAMDisk 镜像。
- **VDI** - VirtualBox 虚拟机监控和 QEMU 模拟器支持的磁盘格式。
- **VHD** - VMware、VirtualBox 等虚拟机监控器使用的通用磁盘格式。
- **PLOOP** - Virtuozzo 支持和使用的磁盘格式来运行 OS 容器。
- **OVA** - 表示镜像服务(glance)中存储的内容是 OVA tar 归档文件。
- **DOCKER** - 表示镜像服务(glance)中存储的内容是容器文件系统的 Docker tar 存档。

虽然 **ISO** 通常不被视为虚拟机镜像格式，因为 ISO 包含有安装操作系统的可引导文件系统，但您使用与其他虚拟机镜像文件相同的方式。

1.2. 支持的镜像服务后端

支持以下镜像服务(glance)后端场景：

- 使用 Ceph 时，RADOS 块设备(RBD)是默认后端。
- RBD 多存储.
- Object Storage (swift)。镜像服务使用 Object Storage 类型和后端作为默认值。
- Block Storage (cinder)。
- NFS

  重要的

  虽然 NFS 是支持的镜像服务部署选项，但提供了更强大的选项。
  NFS 不适用于镜像服务。当您在镜像服务上挂载 NFS 共享时，镜像服务不管理该操作。镜像服务将数据写入文件系统，但不知道后端是 NFS 共享。

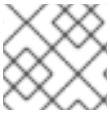
在这种部署中，如果共享失败，镜像服务无法重试请求。这意味着，当后端发生故障时，存储可能会进入只读模式，或者可能会继续将数据写入本地文件系统，在这种情况下，会面临数据丢失的风险。要从这种情况中恢复，您必须确保共享已挂载并同步，然后重启镜像服务。因此，红帽不推荐将 NFS 用作镜像服务后端。

但是，如果您选择将 NFS 用作镜像服务后端，则以下一些最佳实践可帮助降低风险：

- 使用可靠的生产级 NFS 后端。
- 确保您在 Controller 节点和 NFS 后端之间有一个强大且可靠的连接：建议使用第 2 层 (L2)网络连接。
- 包括挂载的共享的监控和警报。
- 设置底层文件系统权限。写入权限必须存在于用作存储的共享文件系统中。
- 确保在其上运行 glance-api 进程的用户和组对本地文件系统没有写入权限。这意味着进程可以检测可能的挂载失败，并在写入尝试期间将存储置于只读模式。

1.3. 镜像签名和验证

镜像签名和验证通过启用部署程序为镜像签名并保存签名和公钥证书作为镜像属性来保护镜像的完整性和真实性。



注意

如果 Nova 使用 RADOS 块设备(RBD)存储虚拟机磁盘，则不支持镜像签名和验证。

有关镜像签名和验证的信息，请参阅 *使用 Key Manager 服务指南* 中的验证镜像服务 (glance) 镜像。

1.4. 镜像格式转换

您可以将镜像转换为不同的格式，方法是在将镜像导入到镜像服务(glance)时激活镜像转换插件。

您可以根据 Red Hat OpenStack Platform (RHOSP)部署配置激活或取消激活镜像转换插件。部署器为部署配置首选镜像格式。

在内部，镜像服务以特定格式接收镜像的位，并将位存储在临时位置。镜像服务触发插件，将镜像转换为目标格式，并将镜像移到最终目的地。任务完成后，镜像服务会删除临时位置。镜像服务不会保留最初上传的格式。

您只能在导入镜像时触发镜像转换。上传镜像时不会运行它。

使用镜像服务命令行客户端进行镜像管理。

例如：

```
$ glance image-create-via-import \
    --disk-format qcow2 \
    --container-format bare \
    --name <name> \
    --visibility public \
    --import-method web-download \
    --uri http://server/image.qcow2
```

- 将 **<name>** 替换为您的镜像的名称。

1.5. 使用镜像服务缓存提高可扩展性

使用镜像服务(glance) API 缓存机制在镜像服务 API 服务器上存储镜像副本，并自动检索它们以提高可扩展性。使用镜像服务缓存，您可以在多个主机上运行 glance-api。这意味着不需要多次从后端存储检索同一镜像。镜像服务缓存不会影响任何镜像服务操作。

使用 Red Hat OpenStack Platform director (tripleo) heat 模板配置镜像服务缓存：

流程

1. 在环境文件中，将 **GlanceCacheEnabled** 参数的值设置为 **true**，这会在 **glance-api.conf** heat 模板中自动将 **flavor** 值设置为 **keystone+cachemanagement** ：

```
parameter_defaults:
  GlanceCacheEnabled: true
```

2. 在重新部署 overcloud 时，将环境文件包含在 **openstack overcloud deploy** 命令中。

3. 可选：在重新部署 overcloud 时将 **glance_cache_pruner** 识别为替代频率。以下示例显示了 5 分钟的频率：

```
parameter_defaults:
  ControllerExtraConfig:
    glance::cache::pruner::minute: '*/5'
```

根据您的需要调整频率，以避免文件系统完全场景。选择替代频率时包含以下元素：

- 要在环境中缓存的文件大小。
- 可用空间量。
- 环境缓存镜像的频率。

1.6. 镜像预缓存

您可以使用 Red Hat OpenStack Platform (RHOSP) director 作为 **glance-api** 服务的一部分来预缓存镜像。

使用 Image 服务(glance)命令行客户端进行镜像管理。

1.6.1. 为定期镜像预缓存配置默认间隔

镜像服务(glance)预缓存定期作业在每个运行 **glance-api** 服务的控制器节点中每 300 秒（默认为 5 分钟）运行。要更改默认时间，您可以在 glance-api.conf 环境文件的 **Default** 部分下设置 **cache_prefetcher_interval** 参数。

流程

1. 根据您的要求，在 undercloud 上的环境文件中添加带有 **ExtraConfig** 参数的新间隔：

```
parameter_defaults:
  ControllerExtraConfig:
```

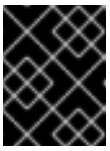
```
glance::config::glance_api_config:
  DEFAULT/cache_prefetcher_interval:
    value: '<300>'
```

- 将 &**lt;300** > 替换为您要作为预缓存镜像间隔的秒数。

2. 在调整 **/home/stack/templates/** 中环境文件中的间隔后，以 **stack** 用户身份登录并部署配置：

```
$ openstack overcloud deploy --templates \
-e /home/stack/templates/<env_file>.yaml
```

- 将 <env_file> 替换为包含您添加的 **ExtraConfig** 设置的环境文件名称。



重要

如果您在创建 overcloud 时传递任何额外的环境文件，请使用 **-e** 选项再次传递它们，以避免对 overcloud 进行不必要的更改。

其他资源

有关 **openstack overcloud deploy** 命令的更多信息，请参阅 *安装和管理 Red Hat OpenStack Platform 指南*中的部署命令。

1.6.2. 准备使用定期作业来预缓存镜像

要使用定期作业预缓存镜像，您必须使用 **glance-cache-manage** 命令直接连接到运行 **glance_api** 服务的节点。不要使用代理，它会隐藏回答服务请求的节点。由于 undercloud 可能无法访问运行 **glance_api** 服务的网络，因此在第一个 overcloud 节点上运行命令，默认为 **controller-0**。

完成以下前提条件步骤，以确保从正确的主机运行命令，具有必要的凭据，并且也从 **glance-api** 容器内运行 **glance-cache-manage** 命令。

流程

1. 以 stack 用户身份登录 undercloud，再识别 **controller-0** 的调配 IP 地址：

```
(undercloud) [stack@site-undercloud-0 ~]$ openstack server list -f value -c Name -c
Networks | grep controller
overcloud-controller-1 ctlplane=192.168.24.40
overcloud-controller-2 ctlplane=192.168.24.13
overcloud-controller-0 ctlplane=192.168.24.71
(undercloud) [stack@site-undercloud-0 ~]$
```

2. 要向 overcloud 进行身份验证，请将存储在 **/home/stack/overcloudrc** 中的凭证复制到 **controller-0** ：

```
$ scp ~/overcloudrc tripleo-admin@192.168.24.71:/home/tripleo-admin/
```

3. 连接到 **controller-0** ：

```
$ ssh tripleo-admin@192.168.24.71
```

4. 在 **controller-0** 上，以 **tripleo-admin** 用户身份识别 **glance_api 服务的** IP 地址。在以下示例中，IP 地址为 **172.25.1.105**:

```
(overcloud) [root@controller-0 ~]# grep -A 10 '^listen glance_api' /var/lib/config-data/puppet-generated/haproxy/etc/haproxy/haproxy.cfg
listen glance_api
 server central-controller0-0.internalapi.redhat.local 172.25.1.105:9292 check fall 5 inter 2000 rise 2
```

5. 由于 **glance-cache-manage** 命令仅在 **glance_api** 容器中可用，因此创建一个脚本，该脚本将执行到该容器中，以向 overcloud 进行身份验证的环境变量已经设置。在 **controller-0** 上的 **/home/tripleo-admin** 中创建一个名为 **glance_pod.sh** 的脚本，其内容如下：

```
sudo podman exec -ti \
 -e NOVA_VERSION=$NOVA_VERSION \
 -e COMPUTE_API_VERSION=$COMPUTE_API_VERSION \
 -e OS_USERNAME=$OS_USERNAME \
 -e OS_PROJECT_NAME=$OS_PROJECT_NAME \
 -e OS_USER_DOMAIN_NAME=$OS_USER_DOMAIN_NAME \
 -e OS_PROJECT_DOMAIN_NAME=$OS_PROJECT_DOMAIN_NAME \
 -e OS_NO_CACHE=$OS_NO_CACHE \
 -e OS_CLOUDNAME=$OS_CLOUDNAME \
 -e no_proxy=$no_proxy \
 -e OS_AUTH_TYPE=$OS_AUTH_TYPE \
 -e OS_PASSWORD=$OS_PASSWORD \
 -e OS_AUTH_URL=$OS_AUTH_URL \
 -e OS_IDENTITY_API_VERSION=$OS_IDENTITY_API_VERSION \
 -e OS_COMPUTE_API_VERSION=$OS_COMPUTE_API_VERSION \
 -e OS_IMAGE_API_VERSION=$OS_IMAGE_API_VERSION \
 -e OS_VOLUME_API_VERSION=$OS_VOLUME_API_VERSION \
 -e OS_REGION_NAME=$OS_REGION_NAME \
glance_api /bin/bash
```

6. Source **overcloudrc** 文件，运行 **glance_pod.sh** 脚本，以使用必要的环境变量在 **glance_api** 中执行以在 overcloud Controller 节点中进行身份验证。

```
[tripleo-admin@controller-0 ~]$ source overcloudrc
(overcloudrc) [tripleo-admin@central-controller-0 ~]$ bash glance_pod.sh
()[glance@controller-0 /]$
```

7. 使用 **glance image-list** 等命令来验证容器是否可以对 overcloud 运行经过身份验证的用户。

```
()[glance@controller-0 /]$ glance image-list
+--------------------------------------+---------------------------------+
| ID                                   | Name                            |
+--------------------------------------+---------------------------------+
| ad2f8daf-56f3-4e10-b5dc-d28d3a81f659 | cirros-0.4.0-x86_64-disk.img    |
+--------------------------------------+---------------------------------+
()[glance@controller-0 /]$
```

1.6.3. 使用定期作业预缓存镜像

当您完成 第 1.6.2 节 “准备使用定期作业来预缓存镜像” 中的先决条件步骤后，您可以使用定期作业来预缓存镜像。

流程

1. 以 admin 用户身份，将镜像排队为缓存：

```
$ glance-cache-manage --host=<host_ip> queue-image <image_id>
```

- 将 <host_ip> 替换为运行 **glance-api** 容器的 Controller 节点的 IP 地址。
- 将 <image_id> 替换为您要队列的镜像 ID。
  当您将要预缓存的镜像排队后，**cache_images** 定期作业会同时获取所有排队的镜像。

注意

由于镜像缓存对每个节点是本地的，因此如果您的 Red Hat OpenStack Platform (RHOSP)部署是 HA，且有 3、5 或 7 Controller，那么在运行 **glance-cache-manage** 命令时，您必须使用 **--host** 选项指定主机地址。

2. 运行以下命令，以查看镜像缓存中的镜像：

```
$ glance-cache-manage --host=<host_ip> list-cached
```

- 将 <host_ip> 替换为您环境中主机的 IP 地址。

1.6.4. 镜像缓存命令选项

您可以使用以下 **glance-cache-manage** 命令选项对镜像排队以进行缓存和管理缓存的镜像：

- **list-cached**，列出当前缓存的所有镜像。
- **list-queued**，列出当前排队以缓存的所有镜像。
- **queue-image** 以排队镜像以进行缓存。
- **delete-cached-image** 从缓存中清除镜像。
- **delete-all-cached-images** 从缓存中删除所有镜像。
- **delete-queued-image** 从缓存队列中删除镜像。
- **delete-all-queued-images** 从缓存队列中删除所有镜像。

1.7. 使用镜像服务 API 启用稀疏镜像上传

使用镜像服务(glance) API，您可以使用稀疏镜像上传来减少网络流量并节省存储空间。此功能在分布式 Compute 节点 (DCN) 环境中特别有用。使用稀疏镜像文件，镜像服务不会写入空字节序列。镜像服务使用给定偏移写入数据。存储后端将这些偏移解释为不实际消耗存储空间的空字节。

使用镜像服务命令行客户端进行镜像管理。

限制

- 仅 Ceph RADOS 块设备(RBD)支持稀疏镜像上传。
- 文件系统不支持稀疏镜像上传。

- 在客户端与镜像服务 API 之间的传输过程中，不会维护稀疏性。镜像在镜像服务 API 一级进行稀疏。

先决条件

- 您的 Red Hat OpenStack Platform (RHOSP)部署将 RBD 用于镜像服务后端。

流程

1. 以 **stack** 用户身份登录 undercloud 节点。
2. 查找 **stackrc** 凭证文件：

```
$ source stackrc
```

3. 使用以下内容创建环境文件：

```
parameter_defaults:
  GlanceSparseUploadEnabled: true
```

4. 使用其他环境文件将新的环境文件添加到堆栈中，并部署 overcloud：

```
$ openstack overcloud deploy \
  --templates \
  …
  -e <existing_overcloud_environment_files> \
  -e <new_environment_file>.yaml \
  ...
```

有关上传镜像的更多信息，请参阅 将镜像上传到镜像服务。

验证

您可以导入镜像并检查其大小以验证稀疏镜像上传。

以下流程使用示例命令。适当地将值替换为您环境中的这些值。

1. 在本地下载镜像文件：

```
$ wget <file_location>/<file_name>
```

   - 将 **<file_location** > 替换为文件的位置。
   - 将 **<file_name** > 替换为文件的名称。
     例如：

```
$ wget https://cloud.centos.org/centos/6/images/CentOS-6-x86_64-GenericCloud-1508.qcow2
```

2. 检查磁盘大小以及要上传的镜像的虚拟大小：

```
$ qemu-img info <file_name>
```

   例如：

```
$ qemu-img info CentOS-6-x86_64-GenericCloud-1508.qcow2

image: CentOS-6-x86_64-GenericCloud-1508.qcow2
file format: qcow2
virtual size: 8 GiB (8589934592 bytes)
disk size: 1.09 GiB
cluster_size: 65536
Format specific information:
compat: 0.10
refcount bits: 1
```

3. 导入镜像：

```
$ glance image-create-via-import --disk-format qcow2 --container-format bare --name centos_1 --file <file_name>
```

4. 记录镜像 ID。后续步骤中需要用到它。

5. 验证镜像是否已导入并处于 active 状态：

```
$ glance image show <image_id>
```

6. 在 Ceph Storage 节点上，验证镜像的大小是否小于第 1 步中的虚拟大小：

```
$ sudo rbd -p images diff <image_id> | awk '{ SUM += $2 } END { print SUM/1024/1024/1024 " GB" }'

1.03906 GB
```

7. 可选：您可以确认在 Controller 节点上的 Image 服务配置文件中配置了 **rbd_thin_provisioning** ：

   a. 使用 SSH 访问 Controller 节点：

   ```
   $ ssh -A -t tripleo-admin@<controller_node_IP_address>
   ```

   b. 确认该 Controller 节点上的 **rbd_thin_provisioning** 等于 **True** ：

   ```
   $ sudo podman exec -it glance_api sh -c 'grep ^rbd_thin_provisioning /etc/glance/glance-api.conf'
   ```

1.8. 安全 METADEF API

在 Red Hat OpenStack Platform (RHOSP)中，云管理员可以使用元数据定义(metadef) API 定义键值对和标签元数据。对云管理员可以创建的 metadef 命名空间、对象、属性、资源或标签的数量没有限制。

镜像服务策略控制 metadef API。默认情况下，只有云管理员才能创建、更新或删除(CUD) metadef API。这个限制可防止 metadef API 向未授权用户公开信息，并减少恶意用户用无限资源填充镜像服务(glance)数据库的风险，从而可以创建拒绝服务(DoS)的攻击。但是，云管理员可覆盖默认策略。

1.9. 为云用户启用 METADEF API 访问

具有依赖于元数据定义(metadef) API 写入访问权限的云管理员可以通过覆盖默认的 admin-only 策略来使所有用户访问这些 API。但是，在这种配置中，可能会意外泄漏敏感资源名称，如客户名称和内部项目。管理员必须审核其系统，以识别之前创建的资源，这些资源可能容易受到攻击，即使所有用户只启用了读写。

流程

1. 作为云管理员，登录 undercloud 并为策略覆盖创建一个文件。例如：

```
$ cat open-up-glance-api-metadef.yaml
```

2. 配置策略覆盖文件，以允许对所有用户进行 metadef API 读写访问：

```
GlanceApiPolicies: {
    glance-metadef_default: { key: 'metadef_default', value: '' },
    glance-get_metadef_namespace: { key: 'get_metadef_namespace', value:
'rule:metadef_default' },
    glance-get_metadef_namespaces: { key: 'get_metadef_namespaces', value:
'rule:metadef_default' },
    glance-modify_metadef_namespace: { key: 'modify_metadef_namespace', value:
'rule:metadef_default' },
    glance-add_metadef_namespace: { key: 'add_metadef_namespace', value:
'rule:metadef_default' },
    glance-delete_metadef_namespace: { key: 'delete_metadef_namespace', value:
'rule:metadef_default' },
    glance-get_metadef_object: { key: 'get_metadef_object', value: 'rule:metadef_default' },
    glance-get_metadef_objects: { key: 'get_metadef_objects', value: 'rule:metadef_default' },
    glance-modify_metadef_object: { key: 'modify_metadef_object', value:
'rule:metadef_default' },
    glance-add_metadef_object: { key: 'add_metadef_object', value: 'rule:metadef_default' },
    glance-delete_metadef_object: { key: 'delete_metadef_object', value: 'rule:metadef_default'
},
    glance-list_metadef_resource_types: { key: 'list_metadef_resource_types', value:
'rule:metadef_default' },
    glance-get_metadef_resource_type: { key: 'get_metadef_resource_type', value:
'rule:metadef_default' },
    glance-add_metadef_resource_type_association: { key:
'add_metadef_resource_type_association', value: 'rule:metadef_default' },
    glance-remove_metadef_resource_type_association: { key:
'remove_metadef_resource_type_association', value: 'rule:metadef_default' },
    glance-get_metadef_property: { key: 'get_metadef_property', value: 'rule:metadef_default'
},
    glance-get_metadef_properties: { key: 'get_metadef_properties', value:
'rule:metadef_default' },
    glance-modify_metadef_property: { key: 'modify_metadef_property', value:
'rule:metadef_default' },
    glance-add_metadef_property: { key: 'add_metadef_property', value: 'rule:metadef_default'
},
    glance-remove_metadef_property: { key: 'remove_metadef_property', value:
'rule:metadef_default' },
    glance-get_metadef_tag: { key: 'get_metadef_tag', value: 'rule:metadef_default' },
    glance-get_metadef_tags: { key: 'get_metadef_tags', value: 'rule:metadef_default' },
    glance-modify_metadef_tag: { key: 'modify_metadef_tag', value: 'rule:metadef_default' },
    glance-add_metadef_tag: { key: 'add_metadef_tag', value: 'rule:metadef_default' },
    glance-add_metadef_tags: { key: 'add_metadef_tags', value: 'rule:metadef_default' },
```

```
  glance-delete_metadef_tag: { key: 'delete_metadef_tag', value: 'rule:metadef_default' },
  glance-delete_metadef_tags: { key: 'delete_metadef_tags', value: 'rule:metadef_default' }
}
```

注意

您必须将所有 metadef 策略配置为使用 **rule:metadeta_default**。

3. 在部署 overcloud 时，请使用 **-e** 选项将新的策略文件包含在部署命令中：

```
$ openstack overcloud deploy -e open-up-glance-api-metadef.yaml
```

第 2 章 创建 RHEL KVM 或 RHOSP 兼容镜像

要创建可在 Red Hat OpenStack Platform (RHOSP)镜像服务(glance)中管理的镜像，您可以使用 Red Hat Enterprise Linux (RHEL)基于内核的虚拟机(KVM)实例镜像，也可以使用 RHEL ISO 文件或 Windows ISO 文件手动创建与 RHOSP 兼容的镜像。

2.1. 创建 RHEL KVM 镜像

使用 Red Hat Enterprise Linux (RHEL)基于内核的虚拟机(KVM)实例镜像来创建您可以在 Red Hat OpenStack Platform (RHOSP)镜像服务(glance)中管理的镜像。

2.1.1. 在 Red Hat OpenStack Platform 中使用 RHEL KVM 实例镜像

您可以在 Red Hat OpenStack Platform (RHOSP)中使用以下 Red Hat Enterprise Linux (RHEL)基于内核的虚拟机(KVM)实例镜像之一：

- Red Hat Enterprise Linux 9 KVM 客户机镜像
- Red Hat Enterprise Linux 8 KVM 客户机镜像

这些 QCOW2 镜像使用 **cloud-init** 配置，且必须具有与 EC2 兼容的元数据服务才能置备 Secure Shell (SSH)密钥才能正常工作。

QCOW2 格式的就绪 Windows KVM 实例镜像不可用。

注意

对于 KVM 实例镜像：

- 镜像中的 **root** 帐户将被停用，但 **sudo** 访问权限被授予一个名为 **cloud-user** 的特殊用户。
- 此镜像没有设置 **root** 密码。

通过将 **!!** 放置到第二个字段中，将 **root** 密码锁定在 **/etc/shadow** 中。

对于 RHOSP 实例，从 RHOSP 仪表板或命令行生成 SSH 密钥对，并使用该组合键以 root 用户身份对实例执行 SSH 公共身份验证。

当您启动实例时，此公钥会注入到其中。然后，您可以使用您在创建密钥对时下载的私钥进行身份验证。

2.1.2. 为裸机实例创建基于 RHEL 的根分区镜像

要为裸机实例创建自定义根分区镜像，请下载基本 Red Hat Enterprise Linux KVM 实例镜像，然后将镜像上传到镜像服务(glance)。

流程

1. 从客户门户网站下载 基本 Red Hat Enterprise Linux KVM 实例镜像。
2. 定义 **DIB_LOCAL_IMAGE** 作为下载的镜像：

```
$ export DIB_LOCAL_IMAGE=rhel-<ver>-x86_64-kvm.qcow2
```

- 将 **<ver** > 替换为镜像的 RHEL 版本号。

3. 根据您的注册方法设置您的注册信息：

- 红帽客户门户网站：

```
$ export REG_USER='<username>'
$ export REG_PASSWORD='<password>'
$ export REG_AUTO_ATTACH=true
$ export REG_METHOD=portal
$ export https_proxy='<IP_address:port>' (if applicable)
$ export http_proxy='<IP_address:port>' (if applicable)
```

- Red Hat Satellite：

```
$ export REG_USER='<username>'
$ export REG_PASSWORD='<password>'
$ export REG_SAT_URL='<satellite-url>'
$ export REG_ORG='<satellite-org>'
$ export REG_ENV='<satellite-env>'
$ export REG_METHOD=<method>
```

- 将尖括号 &**lt**;> 中的值替换为您的红帽客户门户网站或 Red Hat Satellite 注册的正确值。

4. 可选：如果您有任何离线存储库，您可以将 **DIB_YUM_REPO_CONF** 定义为本地存储库配置：

```
$ export DIB_YUM_REPO_CONF=<file-path>
```

- 将 **<file-path** > 替换为本地存储库配置文件的路径。

5. 使用 **diskimage-builder** 工具将内核提取为 **rhel-image.vmlinuz**，初始 RAM 磁盘为 **rhel-image.initrd** ：

```
$ export DIB_RELEASE=<ver>
$ disk-image-create rhel baremetal \
  -o rhel-image
```

6. 将镜像上传到镜像服务：

```
$ KERNEL_ID=$(openstack image create \
  --file rhel-image.vmlinuz --public \
  --container-format aki --disk-format aki \
  -f value -c id rhel-image.vmlinuz)
$ RAMDISK_ID=$(openstack image create \
  --file rhel-image.initrd --public \
  --container-format ari --disk-format ari \
  -f value -c id rhel-image.initrd)
$ openstack image create \
  --file rhel-image.qcow2   --public \
  --container-format bare \
  --disk-format qcow2 \
  --property kernel_id=$KERNEL_ID \
  --property ramdisk_id=$RAMDISK_ID \
  rhel-root-partition-bare-metal-image
```

2.1.3. 为裸机实例创建基于 RHEL 的整个磁盘用户镜像

要为裸机实例创建完整磁盘用户镜像，请下载基本 Red Hat Enterprise Linux KVM 实例镜像，然后将镜像上传到镜像服务(glance)。

流程

1. 从客户门户网站下载 基本 Red Hat Enterprise Linux KVM 实例镜像。
2. 定义 **DIB_LOCAL_IMAGE** 作为下载的镜像：

```
$ export DIB_LOCAL_IMAGE=rhel-<ver>-x86_64-kvm.qcow2
```

   - 将 **<ver** > 替换为镜像的 RHEL 版本号。

3. 根据您的注册方法设置您的注册信息：
   - 红帽客户门户网站：

```
$ export REG_USER='<username>'
$ export REG_PASSWORD='<password>'
$ export REG_AUTO_ATTACH=true
$ export REG_METHOD=portal
$ export https_proxy='<IP_address:port>' (if applicable)
$ export http_proxy='<IP_address:port>' (if applicable)
```

   - Red Hat Satellite：

```
$ export REG_USER='<username>'
$ export REG_PASSWORD='<password>'
$ export REG_SAT_URL='<satellite-url>'
$ export REG_ORG='<satellite-org>'
$ export REG_ENV='<satellite-env>'
$ export REG_METHOD=<method>
```

   - 将尖括号 & **lt**;> 中的值替换为您的红帽客户门户网站或 Red Hat Satellite 注册的正确值。

4. 可选：如果您有任何离线存储库，您可以将 **DIB_YUM_REPO_CONF** 定义为本地存储库配置：

```
$ export DIB_YUM_REPO_CONF=<file-path>
```

   - 将 **<file-path** > 替换为本地存储库配置文件的路径。

5. 将镜像上传到镜像服务：

```
$ openstack image create \
  --file rhel-image.qcow2 --public \
  --container-format bare \
  --disk-format qcow2 \
  rhel-whole-disk-bare-metal-image
```

2.2. 使用 RHEL 或 WINDOWS ISO 文件创建实例镜像

您可以从 ISO 文件以 QCOW2 格式创建自定义 Red Hat Enterprise Linux (RHEL)或 Windows 镜像，并将这些镜像上传到 Red Hat OpenStack Platform (RHOSP)镜像，以便在创建实例时使用。

2.2.1. 先决条件

- 用于创建镜像的 Linux 主机机器。这可以是您可以安装和运行 Linux 软件包的任何计算机，但 undercloud 或 overcloud 除外。
- **advanced-virt** 存储库已启用：

```
$ sudo subscription-manager repos --enable=advanced-virt-for-rhel-<ver>-x86_64-rpms
```

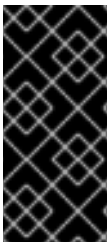
- **virt-manager** 应用程序安装有创建客户机操作系统所需的所有软件包：

```
$ sudo dnf module install -y virt
```

- **libguestfs-tools** 软件包安装有一组用于访问和修改虚拟机镜像的工具：

```
$ sudo dnf install -y libguestfs-tools-c
```

- RHEL 9 或 8 ISO 文件或 Windows ISO 文件。有关 RHEL ISO 文件的详情，请参考 RHEL 9.0 Binary DVD 或 RHEL 8.6 Binary DVD。如果您没有 Windows ISO 文件，请参阅 Microsoft 评估中心 下载评估镜像。
- 文本编辑器，如果要更改 **kickstart** 文件（仅限 RHEL）。



重要

如果在 undercloud 上安装 **libguestfs-tools** 软件包，请取消激活 **iscsid.socket**，以避免在 undercloud 上与 **tripleo_iscsid** 服务冲突：

```
$ sudo systemctl disable --now iscsid.socket
```

当您有先决条件时，您可以继续创建 RHEL 或 Windows 镜像：

- 创建 Red Hat Enterprise Linux 9 镜像
- 创建 Red Hat Enterprise Linux 8 镜像
- 创建 Windows 镜像

2.2.2. 创建 Red Hat Enterprise Linux 9 镜像

您可以使用 Red Hat Enterprise Linux (RHEL) 9 ISO 文件以 QCOW2 格式创建 Red Hat OpenStack Platform (RHOSP)镜像。

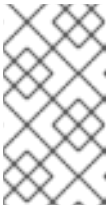
流程

1. 以 **root** 用户身份登录您的主机机器。
2. 使用 **virt-install** 开始安装：

```
[root@host]# virt-install \
```

```
--virt-type kvm \
--name <rhel9-cloud-image> \
--ram <2048> \
--cdrom </var/lib/libvirt/images/rhel-9.0-x86_64-dvd.iso> \
--disk <rhel9.qcow2>,format=qcow2,size=<10> \
--network=bridge:virbr0 \
--graphics vnc,listen=127.0.0.1 \
--noautoconsole \
--os-variant=<rhel9.0>
```

- 将尖括号 &lt **;&** gt; 中的值替换为 RHEL 9 镜像的正确值。
  此命令会启动一个实例并启动安装过程。



注意

如果实例没有自动启动，请运行 **virt-viewer** 命令来查看控制台：

```
[root@host]# virt-viewer <rhel9-cloud-image>
```

3. 配置实例：
   a. 在初始安装程序引导菜单中，选择 **Install Red Hat Enterprise Linux 9**。
   b. 选择相应的 **Language** 和 **Keyboard** 选项。
   c. 当系统提示安装所使用的设备类型时，请选择 **Auto-detected installation media**。
   d. 当提示安装目的地类型时，请选择 **Local Standard Disks**。对于其他存储选项，请选择 **Automatically configure partitioning**。
   e. 在 **您想要的安装类型中**，选择" **基本服务器** 安装"，这将安装 SSH 服务器。
   f. 对于网络和主机名，请选择 **eth0** 作为网络，然后选择您的设备的主机名。默认主机名是 **localhost.localdomain**。
   g. 在 **Root Password** 字段中输入密码，然后在 **Confirm** 字段中输入同一密码。
4. 当屏幕消息确认安装已完成后，重启实例并以 root 用户身份登录。
5. 更新 **/etc/sysconfig/network-scripts/ifcfg-eth0** 文件，使其只包含以下值：

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

6. 重启机器。
7. 使用 Content Delivery Network 注册机器。

```
# sudo subscription-manager register
# sudo subscription-manager attach \
 --pool=<pool-id>
```

```
# sudo subscription-manager repos \
  --enable rhel-9-for-x86_64-baseos-rpms \
  --enable rhel-9-for-x86_64-appstream-rpms
```

- 使用有效的池 ID 替换 **pool-id**。您可以通过运行 **subscription-manager list --available** 命令来查看可用池 ID 列表。

8. 更新系统：

```
# dnf -y update
```

9. 安装 **cloud-init** 软件包：

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

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

```
- resolv-conf
```

resolv-conf 选项会在实例第一次引导时自动配置 **resolv.conf** 文件。此文件包含与实例相关的信息，如 **名称服务器**、**域** 和其他选项。

11. 在 **/etc/sysconfig/network** 中添加以下行以避免访问 EC2 元数据服务时出现问题：

```
NOZEROCONF=yes
```

12. 要确保控制台信息会在仪表板的 **Log** 标签页和 **nova console-log** 输出中显示，请在 **/etc/default/grub** 文件中添加以下引导选项：

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

13. 运行 **grub2-mkconfig** 命令：

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

输出如下：

```
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-229.9.2.el9.x86_64
Found initrd image: /boot/initramfs-3.10.0-229.9.2.el9.x86_64.img
Found linux image: /boot/vmlinuz-3.10.0-121.el9.x86_64
Found initrd image: /boot/initramfs-3.10.0-121.el9.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-b82a3044fb384a3f9aeacf883474428b
Found initrd image: /boot/initramfs-0-rescue-b82a3044fb384a3f9aeacf883474428b.img
done
```

14. 取消注册实例，以便生成的镜像不包含此实例的订阅详情：

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

15. 关闭实例：

```
# poweroff
```

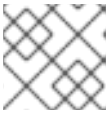
16. 使用 **virt-sysprep** 命令重置并清理镜像，以便它可用于在没有问题的情况下创建实例：

```
[root@host]# virt-sysprep -d <rhel9-cloud-image>
```

17. 通过将磁盘镜像中的任何可用空间转换为主机中的可用空间来减小镜像大小：

```
[root@host]# virt-sparsify \
  --compress <rhel9.qcow2> <rhel9-cloud.qcow2>
```

此命令在运行命令的位置中创建一个新的 < **rhel9-cloud.qcow** 2> 文件。



注意

您必须根据应用到实例的类别中的磁盘空间手动调整实例的分区大小。

& **lt;rhel9-cloud.qcow** 2> 镜像文件可上传到镜像服务。有关将此 镜像上传到 RHOSP 部署的更多信息，请参阅将镜像上传到镜像服务。

2.2.3. 创建 Red Hat Enterprise Linux 8 镜像

您可以使用 Red Hat Enterprise Linux (RHEL) 8 ISO 文件以 QCOW2 格式创建 Red Hat OpenStack Platform (RHOSP)镜像。

流程

1. 以 **root** 用户身份登录您的主机机器。
2. 使用 **virt-install** 开始安装：

```
[root@host]# virt-install \
  --virt-type kvm \
  --name <rhel86-cloud-image> \
  --ram <2048> \
  --vcpus <2> \
  --disk <rhel86.qcow2>,format=qcow2,size=<10> \
  --location <rhel-8.6-x86_64-boot.iso> \
  --network=bridge:virbr0 \
  --graphics vnc,listen=127.0.0.1 \
  --noautoconsole \
  --os-variant <rhel8.6>
```

- 将尖括号 &lt **;&** gt; 中的值替换为 RHEL 镜像的正确值。
  此命令会启动一个实例并启动安装过程。

注意

如果实例没有自动启动，请运行 **virt-viewer** 命令来查看控制台：

```
[root@host]# virt-viewer <rhel86-cloud-image>
```

3. 配置实例：

   a. 在初始安装程序引导菜单中，选择 **Install Red Hat Enterprise Linux 8**。

   b. 选择相应的 **Language** 和 **Keyboard** 选项。

   c. 当系统提示安装所使用的设备类型时，请选择 **基本存储设备**。

   d. 为您的设备选择一个主机名。默认主机名是 **localhost.localdomain**。

   e. 设置 **时区和 root** 密码。

   f. 在 **您想要的安装类型中**，选择" **基本服务器** 安装"，这将安装 SSH 服务器。

4. 当屏幕消息确认安装已完成后，重启实例并以 root 用户身份登录。

5. 更新 **/etc/sysconfig/network-scripts/ifcfg-eth0** 文件，使其只包含以下值：

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

6. 重启机器。

7. 使用 Content Delivery Network 注册机器：

```
# sudo subscription-manager register
# sudo subscription-manager attach \
  --pool=<pool-id>
# sudo subscription-manager repos \
  --enable rhel-8-for-x86_64-baseos-rpms \
  --enable rhel-8-for-x86_64-appstream-rpms
```

   - 使用有效的池 ID 替换 **pool-id**。您可以通过运行 **subscription-manager list --available** 命令来查看可用池 ID 列表。

8. 更新系统：

```
# dnf -y update
```

9. 安装 **cloud-init** 软件包：

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

10. 编辑 **/etc/cloud/cloud.cfg** 配置文件，并在 **cloud_init_modules** 下添加以下内容。

```
- resolv-conf
```

    resolv-conf 选项会在实例第一次引导时自动配置 **resolv.conf** 文件。此文件包含与实例相关的信息，如 **名称服务器**、**域** 和其他选项。

11. 要防止网络问题，请创建 **/etc/udev/rules.d/75-persistent-net-generator.rules** ：

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

这可防止创建 **/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，当您从快照引导时，网络可能无法正常工作，因为网络接口是作为 **eth1** 而不是 **eth0** 创建的，且 IP 地址没有被分配。

12. 在 **/etc/sysconfig/network** 中添加以下行以避免访问 EC2 元数据服务时出现问题：

```
NOZEROCONF=yes
```

13. 要确保控制台信息会在仪表板的 **Log** 标签页和 **nova console-log** 输出中显示，请在 **/etc/grub.conf** 文件中添加以下引导选项：

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

14. 取消注册实例，以便生成的镜像不包含此实例的相同订阅详情：

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

15. 关闭实例：

```
# poweroff
```

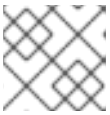
16. 使用 **virt-sysprep** 命令重置并清理镜像，以便它可用于在没有问题的情况下创建实例：

```
[root@host]# virt-sysprep -d <rhel86-cloud-image>
```

17. 通过将磁盘镜像中的任何可用空间转换为主机中的可用空间来减小镜像大小：

```
[root@host]# virt-sparsify \
  --compress <rhel86.qcow2> <rhel86-cloud.qcow2>
```

此命令在运行命令的位置中创建一个新的 < **rhel86-cloud.qcow** 2> 文件。



注意

您必须根据应用到实例的类别中的磁盘空间手动调整实例的分区大小。

& **lt;rhel86-cloud.qcow** 2> 镜像文件可上传到镜像服务。有关将此 镜像上传到 RHOSP 部署的更多信息，请参阅将镜像上传到镜像服务。

2.2.4. 创建 Windows 镜像

您可以使用 Windows ISO 文件以 QCOW2 格式创建 Red Hat OpenStack Platform (RHOSP)镜像。

流程

1. 以 **root** 用户身份登录您的主机机器。
2. 使用 **virt-install** 开始安装：

```
[root@host]# virt-install \
  --name=<windows-image> \
  --disk size=<size> \
  --cdrom=<file-path-to-windows-iso-file> \
  --os-type=windows \
  --network=bridge:virbr0 \
  --graphics spice \
  --ram=<ram>
```

- 将尖括号 &lt **;&** gt; 中的值替换为您的 Windows 镜像的正确值。

注意

--os-type=windows 参数确保为 Windows 实例正确配置时钟，并启用其 Hyper-Vlightenment 功能。在将镜像上传到镜像服务(glance)之前，还必须在镜像元数据中设置 **os_type=windows**。

3. **virt-install** 命令默认将实例镜像保存为 **/var/lib/libvirt/images/<windows-image>.qcow2**。如果要在其他位置保留实例镜像，请更改 **--disk** 选项的参数：

```
--disk path=<file-name>,size=<size>
```

- 将 **<file-name** > 替换为存储实例镜像的文件的名称，以及其路径（可选）。例如，**path=win8.qcow2,size=8** 在当前工作目录中创建一个名为 **win8.qcow2** 的 8 GB 文件。

注意

如果实例没有自动启动，请运行 **virt-viewer** 命令来查看控制台：

```
[root@host]# virt-viewer <windows-image>
```

有关如何安装 Windows 的更多信息，请参阅 Microsoft 文档。

4. 要允许新安装的 Windows 系统使用虚拟化硬件，您可能需要安装 VirtIO 驱动程序。如需更多信息，请参阅 *配置和管理虚拟化* 中的 为 Windows 虚拟机安装 KVM 半虚拟驱动程序。

5. 要完成配置，请在 Windows 系统上下载并运行 Cloudbase-Init。在安装 Cloudbase-Init 的末尾，选择 **Run Sysprep** 和 **Shutdown** 复选框。**Sysprep** 工具通过生成操作系统 ID （供某些 Microsoft 服务使用）使实例是唯一的。

重要

红帽不提供对 Cloudbase-Init 的技术支持。如果您遇到问题，请联系 Cloudbase Solutions。

当 Windows 系统关闭时，& **lt;windows-image.qcow** 2> 镜像文件已准备好上传到镜像服务。有关将此 镜像上传到 RHOSP 部署的更多信息，请参阅将镜像上传到镜像服务。

2.3. 为 UEFI 安全引导创建镜像

当 overcloud 包含 UEFI 安全引导 Compute 节点时，您可以创建一个安全引导实例镜像，供云用户用于启动安全引导实例。

流程

1. 为 UEFI 安全引导创建新镜像：

```
$ openstack image create --file <base_image_file> uefi_secure_boot_image
```

   - 将 **<base_image_file** > 替换为支持 UEFI 和 GUID 分区表(GPT)标准的镜像文件，并包括 EFI 系统分区。

2. 如果默认机器类型不是 **q35**，请将机器类型设置为 **q35** ：

```
$ openstack image set --property hw_machine_type=q35 uefi_secure_boot_image
```

3. 指定实例必须调度到 UEFI 安全引导主机上：

```
$ openstack image set \
 --property hw_firmware_type=uefi \
 --property os_secure_boot=required \
 uefi_secure_boot_image
```

2.4. 虚拟硬件的元数据属性

Compute 服务(nova)已弃用了对使用 **libosinfo** 数据来设置默认设备模型的支持。反之，使用以下镜像元数据属性为实例配置最佳虚拟硬件：

- **os_distro**
- **os_version**
- **hw_cdrom_bus**
- **hw_disk_bus**
- **hw_scsi_model**
- **hw_vif_model**
- **hw_video_model**
- **hypervisor_type**

有关这些元数据属性的更多信息，请参阅 镜像配置参数。

第 3 章 管理镜像、镜像属性和镜像格式

管理镜像以及您在 Red Hat OpenStack Platform (RHOSP)镜像服务(glance)中上传、导入或存储的镜像的属性和格式。

3.1. 将镜像上传到镜像服务

将镜像上传到 Red Hat OpenStack Platform (RHOSP)镜像服务(glance)。

流程

- 使用 **glance image-create** 命令和 **property** 选项来上传镜像。
  例如：

```
$ glance image-create --name <name> \
  --is-public true --disk-format qcow2 \
  --container-format bare \
  --file <image-file> \
  --property <image-metadata>
```

  - 将尖括号 &lt **;&** gt; 中的值替换为您的镜像的正确值。
  - 有关 **glance image-create** 命令选项列表，请参阅 镜像服务命令选项。
  - 有关属性键的列表，请参阅 镜像配置参数。

3.2. 镜像服务镜像导入方法

您可以使用以下方法将镜像导入到 Red Hat OpenStack Platform (RHOSP)镜像服务(glance)：

- 使用 **web-download** （默认）方法从 URI 导入镜像。
- 使用 **glance-direct** 方法从本地文件系统导入镜像。
- 使用 **copy-image** 方法将现有镜像复制到部署中的其他镜像服务后端。只有在部署中启用了多个镜像服务后端时，才使用此导入方法。

web-download 方法默认为启用。云管理员配置导入方法。您可以运行 **glance import-info** 命令来列出可用的导入选项。

3.2.1. 从远程 URI 导入镜像

您可以使用 **web-download** 镜像导入方法将镜像从远程 URI 复制到 Red Hat OpenStack Platform (RHOSP)镜像服务(glance)。

镜像服务 **Web 下载** 方法使用两阶段进程来执行导入：

1. **Web 下载** 方法创建一个镜像记录。
2. **Web 下载** 方法从指定的 URI 检索镜像。

URI 受可选的 denylist 和 allowlist 过滤的影响。

Image Property Injection 插件可能会向镜像注入元数据属性。这些注入的属性决定了镜像实例在其上启动哪些 Compute 节点。

流程

- 创建镜像并指定要导入的镜像的 URI：

```
$ glance image-create-via-import \
   --container-format <container-format> \
   --disk-format <disk-format> \
   --name <name> \
   --import-method web-download \
   --uri <uri>
```

  - 将 **<container-format** > 替换为您要为镜像设置的容器格式(None, ami, ari, aki, bare, ovf, ova, docker)。
  - 将 **<disk-format** > 替换为您要为镜像设置的磁盘格式(None, ami, ari, aki, vhd, vhdx, vmdk, raw, qcow2, vdi, iso, ploop)。
  - 将 **<name** > 替换为您的镜像的描述性名称。
  - 将 **<uri** > 替换为您的镜像的 URI。

验证

- 检查镜像的可用性：

```
$ glance image-show <image-id>
```

  - 将 **<image-id** > 替换为您在创建镜像过程中提供的 ID。

3.2.2. 从本地卷导入镜像

glance-direct 镜像导入方法会创建一个镜像记录，它会生成镜像 ID。当您从本地卷将镜像上传到镜像服务(glance)时，该镜像存储在暂存区域中，并在通过任何配置的检查后激活。在高可用性(HA)配置中使用时，**glance-direct** 方法需要一个共享暂存区域。

注意

如果您使用 **glance-direct** 导入方法上传镜像，则上传可能会在 HA 环境中失败（如果不存在通用暂存区域）。在 HA 主动环境中，API 调用被分发到镜像服务控制器。下载 API 调用可以发送到与 API 调用不同的控制器，以上传镜像。

glance-direct 镜像导入方法使用三个不同的调用来导入镜像：

- **glance image-create**
- **glance image-stage**
- **glance image-import**

流程

- 使用 **glance image-create-via-import** 命令在一个命令中执行所有三个 **glance-direct** 调用：

```
$ glance image-create-via-import \
  --container-format <container-format> \
  --disk-format <disk-format> \
  --name <name> \
  --file </path/to/image>
```

  - 将 **<container-format>** 、**<disk-format>** 和 **</path/to/image>** 替换为您的镜像的相关值。
    当镜像从暂存区域移到后端位置时，会列出该镜像。但是，可能需要过些时间，镜像才会变为活动状态。

验证

- 检查镜像的可用性：

```
$ glance image-show <image-id>
```

  - 将 **<image-id>** 替换为您在创建镜像过程中提供的 ID。

3.3. 更新镜像属性

更新存储在 Red Hat OpenStack Platform (RHOSP)镜像服务(glance)中的镜像属性。

流程

- 使用 **glance image-update** 命令和 **property** 选项来更新镜像。
  例如：

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

  - 有关 **glance image-update** 命令选项列表，请参阅 镜像服务(glance)命令选项。
  - 有关属性键的列表，请参阅 镜像配置参数。

3.4. 启用镜像转换

您可以通过启用 **GlanceImageImportPlugins** 参数，将 QCOW2 镜像上传到镜像服务(glance)。然后，您可以将 QCOW2 镜像转换为 RAW 格式。

注意

当使用 Red Hat Ceph Storage RADOS 块设备(RBD)来存储镜像和引导 Nova 实例时，镜像转换会自动启用。

要启用镜像转换，请创建一个包含以下参数值的环境文件。在 **openstack overcloud deploy** 命令中使用 **-e** 选项的新环境文件：

```
parameter_defaults:
  GlanceImageImportPlugins:'image_conversion'
```

使用镜像服务命令行客户端进行镜像管理。

3.4.1. 将镜像转换为 RAW 格式

Red Hat Ceph Storage 可以存储，但不支持使用 QCOW2 镜像来托管虚拟机(VM)磁盘。

当您上传 QCOW2 镜像并从中创建虚拟机时，计算节点会下载镜像，将镜像转换为 RAW，然后将其上传到 Ceph，然后使用它。这个过程会影响创建虚拟机所需的时间，特别是在并行虚拟机创建过程中。

例如，当您同时创建多个虚拟机时，上传转换的镜像到 Ceph 集群可能会影响已在运行的工作负载。上传过程可能会使 IOPS 的工作负载以及存储响应速度导致。

要在 Ceph 中更有效地引导虚拟机（临时后端或从卷引导），glance 镜像格式必须是 RAW。

流程

1. 将镜像转换为 RAW 可能会产生大于原始 QCOW2 镜像文件的镜像。在转换前运行以下命令，以确定最终 RAW 镜像大小：

```
$ qemu-img info <image>.qcow2
```

2. 将镜像从 QCOW2 转换为 RAW 格式：

```
$ qemu-img convert -p -f qcow2 -O raw <original qcow2 image>.qcow2 <new raw image>.raw
```

3.4.2. 使用 GlanceDiskFormats 参数配置磁盘格式

您可以使用 **GlanceDiskFormats** 参数配置镜像服务(glance)以启用或禁用磁盘格式。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 提供 undercloud 凭证文件：

```
$ source ~/stackrc
```

3. 在环境文件中包含 **GlanceDiskFormats** 参数，如 **glance_disk_formats.yaml** ：

```
parameter_defaults:
  GlanceDiskFormats:
    - <disk_format>
```

- 例如，使用以下配置只启用 RAW 和 ISO 磁盘格式：

```
parameter_defaults:
  GlanceDiskFormats:
  - raw
  - iso
```

- 使用以下示例配置拒绝 QCOW2 磁盘镜像：

```
parameter_defaults:
  GlanceDiskFormats:
  - raw
  - iso
  - aki
  - ari
  - ami
```

4. 在 **openstack overcloud deploy** 命令中包含新配置的环境文件以及与您环境相关的任何其他环境文件：

```
$ openstack overcloud deploy --templates \
  -e <overcloud_environment_files> \
  -e <new_environment_file> \
  …
```

- 将 **<overcloud_environment_files** > 替换为属于部署的环境文件列表。
- 将 **<new_environment_file** > 替换为包含新配置的环境文件。

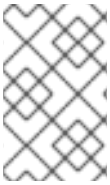
有关 RHOSP 中可用磁盘格式的更多信息，请参阅 镜像配置参数。

3.4.3. 以 RAW 格式存储镜像

启用 **GlanceImageImportPlugins** 参数后，运行以下命令以 RAW 格式存储之前创建的镜像：

```
$ glance image-create-via-import \
    --disk-format qcow2 \
    --container-format bare \
    --name <name> \
    --visibility public \
    --import-method web-download \
    --uri <http://server/image.qcow2>
```

- 将 &**lt**;name> 替换为镜像的名称；这是 **glance image-list** 中显示的名称。
- 将 **&lt**;http://server/image.qcow2> 替换为 QCOW2 镜像的位置和文件名。



注意

此命令示例创建镜像记录并使用 **web-download** 方法导入它。glance-api 在导入过程中从 **--uri** 位置下载镜像。如果 **web-download** 不可用，**glanceclient** 无法自动下载镜像数据。运行 **glance import-info** 命令来列出可用的镜像导入方法。

3.5. 隐藏或取消隐藏镜像

您可以从向用户呈现的普通列表中隐藏公共镜像。例如，您可以隐藏过时的 CentOS 7 镜像，并只显示最新版本来简化用户体验。用户可以发现和使用隐藏的镜像。

要创建隐藏的镜像，请将 **--hidden** 参数添加到 **glance image-create** 命令。

流程

- 隐藏镜像：

```
$ glance image-update <image_id> --hidden 'true'
```

- 解译镜像：

```
$ glance image-update <image_id> --hidden 'false'
```

- 列出隐藏的镜像：

```
$ glance image-list --hidden 'true'
```

3.6. 从镜像服务中删除镜像

使用 **glance image-delete** 命令删除不需要存储在镜像服务(glance)中的一个或多个镜像。

流程

- 删除一个或多个镜像：

```
$ glance image-delete <image-id> [<image-id> ...]
```

  - 将 **<image-id** > 替换为您要删除的镜像的 ID。

警告

glance image-delete 命令永久删除镜像以及镜像的所有副本，以及镜像实例和元数据。

第 4 章 配置镜像导入方法和共享暂存

Red Hat OpenStack Platform (RHOSP)镜像服务(glance)的默认设置由安装 RHOSP 时使用的 heat 模板决定。Image service heat 模板为 **deployment/glance/glance-api-container-puppet.yaml**。

您可以使用 **web-download** 方法或 **glance-direct** 方法导入镜像。有关这些导入方法的更多信息，请参阅镜像服务镜像导入方法。

4.1. 创建和部署 GLANCE-SETTINGS.YAML 文件

使用自定义环境文件配置导入参数。这些参数覆盖核心 heat 模板集合中存在的默认值。示例环境内容包含可组合镜像导入的参数。

```
parameter_defaults:
  # Configure NFS backend
  GlanceBackend: file
  GlanceNfsEnabled: true
  GlanceNfsShare: 192.168.122.1:/export/glance

  # Enable glance-direct import method
  GlanceEnabledImportMethods: glance-direct,web-download

  # Configure NFS staging area (required for glance-direct import method)
  GlanceStagingNfsShare: 192.168.122.1:/export/glance-staging
```

GlanceBackend、**GlanceNfsEnabled** 和 **GlanceNfsShare** 参数在 *Overcloud 参数* 指南中定义。

使用两个新参数进行可互操作的镜像导入，以定义导入方法和共享 NFS 暂存区域。

GlanceEnabledImportMethods

定义可用的导入方法、web-download （默认）和 glance-direct。只有在您希望在 web-download 外启用附加方法时，才需要此参数。

GlanceStagingNfsShare

配置 glance-direct 导入方法使用的 NFS 暂存区域。此空间可以在高可用性集群配置中的节点之间共享。如果要使用此参数，还必须将 **GlanceNfsEnabled** 参数设置为 **true**。

流程

1. 创建新文件，如 **glance-settings.yaml**。使用示例中的语法来填充此文件。
2. 在 **openstack overcloud deploy** 命令中包含 **glance-settings.yaml** 文件，以及与部署相关的任何其他环境文件：

```
$ openstack overcloud deploy --templates -e glance-settings.yaml
```

有关使用环境文件的更多信息，*请参阅使用 director 安装和管理 Red Hat OpenStack Platform* 指南。

4.2. 控制镜像 WEB-IMPORT 源

您可以通过将 URI blocklist 和 allowlists 添加到可选的 **glance-image-import.conf** 文件中来限制 web-import 镜像下载源。

您可以在三个级别允许或阻止镜像源 URI：

- scheme (allowed_schemes, disallowed_schemes)
- host (allowed_hosts, disallowed_hosts)
- port (allowed_ports, disallowed_ports)

如果您在任何级别上同时指定了 allowlist 和 blocklist，则会遵守允许列表，并忽略 blocklist。

Image 服务(glance)应用以下决策逻辑来验证镜像源 URI：

1. 检查方案。
   a. 缺少方案： reject
   b. 如果存在允许列表，并且 allowlist: reject 中没有方案。否则，跳过 C 并继续 2。
   c. 如果存在黑名单，且 blocklist: reject 中存在方案。
2. 检查主机名。
   a. 缺少主机名： reject
   b. 如果存在允许列表，并且 allowlist: reject 中没有主机名。否则，跳过 C 并继续 3。
   c. 如果存在黑名单，并且 blocklist: reject 中存在主机名。
3. 如果 URI 中存在端口，则会检查端口。
   a. 如果存在允许列表，并且 allowlist: reject 中没有端口。否则，跳过 B 并继续 4。
   b. 如果存在黑名单，并且该端口存在于 blocklist: reject 中。
4. URI 被接受为 valid。

如果您允许方案，可以将它添加到允许列表中，或者不将其添加到黑名单中，则允许将这个方案使用默认端口的任何 URI 添加到该方案中。如果在 URI 中包含端口，则 URI 会根据默认的决策逻辑进行验证。

4.2.1. 镜像导入允许列表示例

在本例中，FTP 的默认端口为 21。

由于 **ftp** 位于 **allowed_schemes** 的列表中，允许此 URL 到 image 资源 ： ftp://example.org/some/resource。

但是，由于 21 不在 **allowed_ports** 的列表中，因此同一镜像资源的 URL 将被拒绝： ftp://example.org:21/some/resource。

```
allowed_schemes = [http,https,ftp]
disallowed_schemes = []
allowed_hosts = []
disallowed_hosts = []
allowed_ports = [80,443]
disallowed_ports = []
```

4.2.2. 默认镜像导入 blocklist 和 allowlist 设置

glance-image-import.conf 文件是一个可选文件，包含以下默认选项：

- allowed_schemes - [*http*, *https*]
- disallowed_schemes - empty list
- allowed_hosts - empty list
- disallowed_hosts - empty list
- allowed_ports - [80, 443]
- disallowed_ports - empty list

如果使用默认值，最终用户只能使用 **http** 或 **https** 方案访问 URI。用户只能指定的端口是 **80** 和 **443。**用户不必指定端口，但是如果这样做，它必须是 **80** 或 **443。**

您可以在镜像服务源代码树的 **etc/** 子目录中找到 **glance-image-import.conf** 文件。确保您正在寻找 Red Hat OpenStack Platform 发行版本的正确分支。

4.3. 在镜像导入时注入元数据，以控制实例启动位置

云用户可以将镜像上传到镜像服务(glance)，并使用这些镜像启动实例。云用户必须在特定的一组 Compute 节点上启动这些镜像。您可以使用镜像元数据属性来控制实例分配给 Compute 节点。

Image Property Injection 插件在导入过程中将元数据属性注入镜像。通过编辑 **glance-image-import.conf** 文件的 [image_import_opts] 和 [inject_metadata_properties] 部分来指定属性。

要启用 Image Property Injection 插件，请在 **[image_import_opts]** 部分添加以下行：

```
[image_import_opts]
image_import_plugins = [inject_image_metadata]
```

要将元数据注入限制到特定用户提供的镜像，请设置 **ignore_user_roles** 参数。例如，使用以下配置将 **property1** 的值和 **property2** 的两个值注入任何非 admin 用户下载的镜像中。

```
[DEFAULT]
[image_conversion]
[image_import_opts]
image_import_plugins = [inject_image_metadata]
[import_filtering_opts]
[inject_metadata_properties]
ignore_user_roles = admin
inject = PROPERTY1:value,PROPERTY2:value;another value
```

参数 **ignore_user_roles** 是插件忽略的身份服务(keystone)角色的逗号分隔列表。这意味着，如果进行镜像导入调用的用户具有任何这些角色，则插件不会将任何属性注入到镜像中。

参数 **注入** 是一个以逗号分隔的属性和值列表，注入导入的镜像记录中。每个属性和值都必须用冒号 **(':')** 分隔。

您可以在镜像服务源代码树的 **etc/** 子目录中找到 **glance-image-import.conf** 文件。确保您正在寻找 Red Hat OpenStack Platform (RHOSP)发行版本的正确分支。

第 5 章 具有多个存储的镜像服务

Red Hat OpenStack Platform (RHOSP)镜像服务(glance)支持使用带有分布式边缘架构的多个存储，以便您可以在每个边缘站点都有镜像池。

5.1. 在多个存储上复制镜像

当您将多个存储与分布式边缘架构搭配使用时，您可以在每个边缘站点都有一个镜像池。您可以在中央站点（也称为 hub 站点）和边缘站点之间复制镜像。

镜像元数据包含每个副本的位置。例如，两个边缘站点上的镜像公开为具有三个位置的单一 UUID：中央站点加上两个边缘站点。这意味着，您可以在多个存储上共享单个 UUID 的镜像数据副本。有关位置的更多信息，请参阅了解镜像的位置。

在每个边缘站点中使用 RADOS 块设备(RBD)镜像池，您可以使用 Ceph RBD 写时复制(COW)和快照分层技术快速引导虚拟机(VM)。这意味着，您可以从卷引导虚拟机并具有实时迁移。有关使用 Ceph RBD 分层的更多信息，请参阅*块设备指南*中的 Ceph 块设备层。

当您在边缘站点启动实例时，所需的镜像会自动复制到本地镜像服务(glance)存储中。但是，您可以使用 glance multistore 在实例启动期间，将镜像从中央镜像存储复制到边缘站点。

5.2. 存储边缘架构的要求

请参考以下要求来将镜像与边缘站点搭配使用：

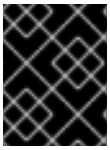
- 每个镜像的副本必须存在于中央位置的 Image 服务(glance)中。
- 您必须将镜像从边缘站点复制到中央位置，然后才能将其复制到其他边缘站点。
- 在使用 Red Hat Ceph Storage 部署分布式 Compute 节点(DCN)架构时，您必须使用原始镜像。
- RADOS 块设备(RBD)必须是镜像、计算和块存储服务的存储驱动程序。
- 对于每个站点，您必须将相同的值分配给 **NovaComputeAvailabilityZone** 和 **CinderStorageAvailabilityZone** 参数。

5.3. 将镜像导入到多个存储

使用可互操作的镜像导入工作流，将镜像数据导入到多个 Red Hat Ceph Storage 集群。您可以将镜像导入到本地文件系统上或通过 Web 服务器可用的镜像服务(glance)。

如果从 web 服务器导入镜像，则可以一次将镜像导入到多个存储中。如果 web 服务器上没有镜像，您可以将镜像从本地文件系统导入到中央存储中，然后将其复制到其他存储中。如需更多信息，请参阅 将现有镜像复制到多个存储。

使用镜像服务命令行客户端进行镜像管理。



重要

始终在中央站点上存储镜像副本，即使中央位置上没有使用该镜像的实例。有关将镜像导入到镜像服务的更多信息，请参阅部署分布式计算节点架构指南。

5.3.1. 管理镜像导入失败

您可以使用 **--allow-failure** 参数管理镜像导入操作的失败：

- 如果 **--allow-failure** 参数的值为 **true**，则第一次存储成功导入数据后镜像状态将 变为**活动状态**。这是默认设置。您可以使用 **os_glance_failed_import** 镜像属性查看无法导入镜像数据的存储列表。
- 如果将 **--allow-failure** 参数的值设置为 **false**，则镜像状态仅在所有指定存储成功导入数据后 变为**活动状态**。导入镜像数据的任何存储失败会导致镜像状态 **失败**。镜像没有导入到任何指定存储中。

5.3.2. 将镜像数据导入到多个存储

由于 **--allow-failure** 参数的默认设置是 **true**，因此如果某些存储可以接受，则不需要在命令中包含该参数，无法导入镜像数据。

注意

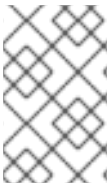
此流程不需要所有存储才能成功导入镜像数据。

流程

- 将镜像数据导入多个指定的存储：

```
$ glance image-create-via-import \
--container-format bare \
--name <image-name> \
--import-method web-download \
--uri <uri> \
--stores <store-1>,<store-2>,<store-3>
```

  - 将 **<image-name** > 替换为您要导入的镜像的名称。
  - 将 <uri> 替换为镜像的 URI。
  - 将 **<store** -1& **gt; ,** &lt **;store-2> , 和 <store-3** > 替换为您要导入镜像数据的存储名称。
  - 或者，将 **--stores** 替换为 **--all-stores true**，以将镜像上传到所有存储。



注意

glance image-create-via-import 命令会自动将 QCOW2 镜像转换为 RAW 格式，仅适用于 **web-download** 方法。**glance-direct** 方法可用，但仅适用于带有配置的共享文件系统的部署。

5.3.3. 在不失败的情况下将镜像数据导入到多个存储中

此流程要求所有存储成功导入镜像数据。

流程

1. 将镜像数据导入多个指定的存储：

```
$ glance image-create-via-import \
--container-format bare \
```

```
--name <image-name> \
--import-method web-download \
--uri <uri> \
--stores <store-1>,<store-2>,<store-3>
```

- 将 **<image-name** > 替换为您要导入的镜像的名称。
- 将 **<uri** > 替换为镜像的 URI。
- 将 **<store** -1& **gt; , < store-2> , 和 <store-3** > 替换为您要复制镜像数据的存储名称。
- 或者，将 **--stores** 替换为 **--all-stores true**，以将镜像上传到所有存储。

注意

将 **--allow-failure** 参数设置为 **false** 时，镜像服务(glance)不会忽略无法导入镜像数据的存储。您可以使用镜像属性 **os_glance_failed_import** 来查看失败的存储列表。如需更多信息，请参阅 第 5.4 节 “检查镜像导入操作的进度”。

2. 验证镜像数据是否已添加到特定存储中：

```
$ glance image-show <image-id> | grep stores
```

将 **<image-id** > 替换为原始现有镜像的 ID。

输出显示以逗号分隔的存储列表。

5.3.4. 将镜像数据导入到单个存储中

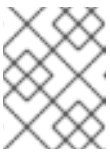
您可以使用镜像服务(glance)将镜像数据导入到单个存储中。

流程

1. 将镜像数据导入到单个存储中：

```
$ glance image-create-via-import \
--container-format bare \
--name <image-name> \
--import-method web-download \
--uri <uri> \
--store <store>
```

- 将 **<image-name** > 替换为您要导入的镜像的名称。
- 将 **<uri** > 替换为镜像的 URI。
- 将 **&** lt;store> 替换为您要复制镜像数据的存储名称。



注意

如果您没有在命令中包括 **--stores**、**--all-stores** 或 **--store** 的选项，则镜像服务会在中央存储中创建镜像。

2. 验证镜像数据是否已添加到特定的存储中：

```
$ glance image-show <image-id> | grep stores
```

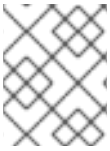
- 将 **<image-id** > 替换为原始现有镜像的 ID。
  输出显示以逗号分隔的存储列表。

5.4. 检查镜像导入操作的进度

可组合镜像导入工作流将镜像数据按顺序导入存储。镜像的大小、存储数量以及中央站点和边缘站点之间的网络速度会影响镜像导入操作所需的时间。

您可以通过查看两个镜像属性来跟踪镜像导入的进度，这些属性会出现在镜像导入操作过程中发送的通知中：

- **os_glance_importing_to_stores** 属性列出了尚未导入镜像数据的存储。在导入开始时，所有请求的存储都显示在列表中。每次存储成功导入镜像数据时，镜像服务会从列表中删除存储。
- **os_glance_failed_import** 属性列出了无法导入镜像数据的存储。此列表在镜像导入操作开始时为空。



注意

在以下步骤中，环境有三个 Red Hat Ceph Storage 集群： **中央存储** 和两个存储在边缘、**dcn0** 和 **dcn1**。

流程

1. 验证镜像数据是否已添加到特定存储中：

```
$ glance image-show <image-id>
```

- 将 **<image-id** > 替换为原始现有镜像的 ID。
  输出显示以逗号分隔的存储列表，类似以下示例片断：

```
| os_glance_failed_import      |
| os_glance_importing_to_stores | central,dcn0,dcn1
| status                       | importing
```

2. 监控镜像导入操作的状态。当您使用 **watch** 命令前有一个命令时，命令输出每两秒钟刷新一次。

```
$ watch glance image-show <image-id>
```

- 将 **<image-id** > 替换为原始现有镜像的 ID。
  当镜像导入操作进行时，操作的状态会改变：

```
| os_glance_failed_import      |
| os_glance_importing_to_stores | dcn0,dcn1
| status                       | importing
```

显示镜像无法导入的输出结果类似以下示例：

```
| os_glance_failed_import      | dcn0
| os_glance_importing_to_stores | dcn1
| status                       | importing
```

-

操作完成后，状态会变为 active ：

```
| os_glance_failed_import       | dcn0
| os_glance_importing_to_stores |
| status                        | active
```

5.5. 将现有镜像复制到多个存储中

此功能允许您使用 Red Hat OpenStack Image 服务(glance)镜像数据使用 Red Hat OpenStack Image 服务(glance)镜像数据复制到边缘的多个 Red Hat Ceph Storage 存储中，方法是使用可组合的镜像导入工作流。

注意

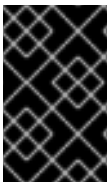
镜像必须在中央站点上存在，然后才能将其复制到任何边缘站点。只有镜像所有者或管理员可以将现有镜像复制到新添加的存储中。

您可以通过将 **--all-stores** 设置为 **true** 或指定要接收镜像数据的特定存储来复制现有镜像数据。

- **--all-stores** 选项的默认设置是 **false**。如果 **--all-stores** 为 **false**，则必须使用 --stores <store-1>,& **lt;store-2> 指定哪个存储接收镜像数据**。如果任何指定存储中已存在镜像数据，则请求会失败。
- 如果将 **all-stores** 设置为 **true**，并且某些存储中已存在镜像数据，则这些存储将不包括在列表中。

指定存储接收镜像数据后，镜像服务(glance)将数据从中央站点复制到暂存区域。然后，镜像服务使用可组合的镜像导入工作流导入镜像数据。如需更多信息，请参阅 将镜像导入到多个存储。

使用镜像服务命令行客户端进行镜像管理。



重要

红帽建议管理员要仔细避免时间非常接近的镜像复制请求。对同一镜像进行两个密切的 copy-image 操作会导致竞争条件和意外的结果。现有镜像数据会保持原样，但将数据复制到新存储中会失败。

5.5.1. 将镜像复制到所有存储中

使用以下步骤将镜像数据复制到所有可用存储中。

流程

1. 将镜像数据复制到所有可用存储中：

```
$ glance image-import <image-id> \
--all-stores true \
--import-method copy-image
```

   - 将 **<image-id** > 替换为您要复制的镜像的名称。

2. 确认镜像数据成功复制到所有可用存储：

-

```
$ glance image-list --include-stores
```

有关如何检查镜像导入操作状态的详情，请参考 第 5.4 节 “检查镜像导入操作的进度”。

5.5.2. 将镜像复制到特定存储中

使用以下步骤将镜像数据复制到特定的存储中。

流程

1. 将镜像数据复制到特定的存储中：

```
$ glance image-import <image-id> \
--stores <store-1>,<store-2> \
--import-method copy-image
```

   - 将 **<image-id** > 替换为您要复制的镜像的名称。
   - 将 **< store-1>** 和 <store-2> 替换为您要复制镜像数据的存储名称。

2. 确认镜像数据成功复制到指定的存储中：

```
$ glance image-list --include-stores
```

有关如何检查镜像导入操作状态的详情，请参考 第 5.4 节 “检查镜像导入操作的进度”。

5.6. 从特定存储中删除镜像

使用 Red Hat OpenStack Platform (RHOSP)镜像服务(glance)删除特定存储上的现有镜像副本。

使用镜像服务命令行客户端进行镜像管理。

流程

- 从特定存储中删除镜像：

```
$ glance stores-delete --store <store-id> <image-id>
```

- 将 **<store-id** > 替换为应删除镜像副本的存储名称。
- 将 **<image-id** > 替换为您要删除的镜像的 ID。

警告

glance image-delete 命令在所有站点间永久删除镜像。所有镜像副本都将被删除，以及镜像实例和元数据。

5.7. 列出镜像位置和位置属性

虽然镜像可以存在于多个站点上，但给定镜像只有一个通用唯一标识符(UUID)。镜像元数据包含每个副本的位置。例如，两个边缘站点中存在的镜像公开为具有三个位置的单个 UUID：中央站点和两个边缘站点。

注意

使用 Image 服务(glance)命令行客户端，而不是 OpenStack 命令行客户端进行镜像管理。但是，使用 **openstack image show** 命令列出镜像位置属性。**glance image-show** 命令输出不包含位置。

流程

1. 显示存在镜像副本的站点：

```
$ glance image-show ID | grep "stores"

| stores |  default_backend,dcn1,dcn2
```

在示例中，镜像存在于中央站点、**default_backend**，以及两个边缘站点 **dcn1** 和 **dcn2** 上。

2. 或者，您可以使用 **--include-stores** 选项运行 **glance image-list** 命令，以查看镜像所在的站点：

```
$ glance image-list --include-stores

| ID                                   | Name    | Stores

| 2bd882e7-1da0-4078-97fe-f1bb81f61b00 | cirros | default_backend,dcn1,dcn2
```

3. 列出镜像位置属性，以显示每个位置的详情：

```
$ openstack image show ID -c properties

| properties |

(--- cut ---)
locations='[{'url': 'rbd://79b70c32-df46-4741-93c0-8118ae2ae284/images/2bd882e7-1da0-
4078-97fe-f1bb81f61b00/snap', 'metadata': {'store': 'default_backend'}}, {'url': 'rbd://63df2767-
8ddb-4e06-8186-8c155334f487/images/2bd882e7-1da0-4078-97fe-f1bb81f61b00/snap',
'metadata': {'store': 'dcn1'}}, {'url': 'rbd://1b324138-2ef9-4ef9-bd9e-
aa7e6d6ead78/images/2bd882e7-1da0-4078-97fe-f1bb81f61b00/snap', 'metadata': {'store':
'dcn2'}}]',
(--- cut --)
```

image 属性显示各个镜像位置的不同 Ceph RBD URI。

在示例中，中央镜像位置 URI 是：

```
rbd://79b70c32-df46-4741-93c0-8118ae2ae284/images/2bd882e7-1da0-4078-97fe-
f1bb81f61b00/snap', 'metadata': {'store': 'default_backend'}}
```

URI 由以下数据组成：

- **79b70c32-df46-4741-93c0-8118ae2ae284** 对应于中央 Ceph FSID。每个 Ceph 集群都有唯一的 FSID。
- 所有站点的默认值为 **images**，对应于存储镜像的 Ceph 池。
- **2bd882e7-1da0-4078-97fe-f1bb81f61b00** 对应于镜像 UUID。给定镜像的 UUID 相同，无论其位置如何。
- 元数据显示此位置映射的 glance 存储。在本例中，它映射到 **default_backend**，这是中央 hub 站点。

附录 A. 镜像服务命令选项

您可以将以下可选参数与 **glance image-create** 和 **glance image-update** 命令一起使用。

表 A.1. 命令选项

特定于	选项	描述
All	**--architecture <ARCHITECTURE>**	如 https://docs.openstack.org/glance/latest/user/common-image-properties.html#architecture 中指定的操作系统架构
All	**--protected [True_False]**	如果为 true，则镜像将无法处理。
All	**--name <NAME>**	镜像的描述性名称
All	**--instance-uuid <INSTANCE_UUID>**	可用于记录该镜像关联的实例的元数据。（仅信息，不创建实例快照。）
All	**--min-disk <MIN_DISK>**	引导镜像所需的磁盘空间量（以 GB 为单位）。
All	**--visibility <VISIBILITY>**	镜像可访问性的范围。有效值：public, private, community, shared
All	**--kernel-id <KERNEL_ID>**	镜像服务(glance)中存储的镜像 ID，在引导 AMI 风格的镜像时应用作内核。
All	**--os-version <OS_VERSION>**	由经销商指定的操作系统版本
All	**--disk-format <DISK_FORMAT>**	磁盘格式。有效值： None, ami, ari, aki, vhd, vhdx, vmdk, raw, qcow2, vdi, iso, ploop
All	**--os-distro <OS_DISTRO>**	https://docs.openstack.org/glance/latest/user/common-image-properties.html#os-distro 中指定的操作系统分布的通用名称
All	**--owner <OWNER>**	镜像的所有者
All	**--ramdisk-id <RAMDISK_ID>**	镜像服务中存储的镜像的 ID，在引导 AMI 样式的镜像时应用作 ramdisk。
All	**--min-ram <MIN_RAM>**	引导镜像所需的 RAM 量（以 MB 为单位）。
All	**--Container-format <CONTAINER_FORMAT>**	容器的格式。有效值： None, ami, ari, aki, bare, ovf, ova, docker
All	**--property <key=value>**	与镜像关联的任意属性。可以多次使用。

特定于	选项	描述
glance image-create	**--tags <TAGS> [<TAGS> ...]**	与镜像相关的字符串列表
glance image-create	**--id <ID>**	镜像的标识符
glance image-update	**--remove-property**	从镜像中删除的任意属性的密钥名称。

附录 B. 镜像配置参数

您可以将以下键与 **glance image-create** 和 **glance image-update** 命令的 **property** 选项一起使用。

表 B.1. 属性键

特定于	键	描述	支持的值
All	**架构**	管理程序必须支持的 CPU 架构。例如，**x86_64**、**arm** 或 **ppc64**。运行 **uname -m** 以获取计算机架构。	• **aarch** - ARM 64 位 • **alpha** - DEC 64 位 RISC • **armv7l** - ARM Cortex-A7 MPCore • **CRIS**- Ethernet, Token Ring, AXis-Code Reduced instructions Set • **i686** - Intel 6th-generation x86（P6 微架构） • **ia64** - Itanium • **lm32** - Lattice Micro32 • **m68k** - Motorola 68000 • **microblaze** - Xilinx 32 位 FPGA (Big Endian) • **microblazeel** - Xilinx 32 位 FPGA (Little Endian) • **M** IPS - MIPS 32 位 RISC (Big Endian) • **mipsel** - MIPS 32 位 RISC (Little Endian) • **mips64** - MIPS 64 位 RISC (Big Endian) • **mips64el** - MIPS 64 位 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 （嵌入式 32 位）

特定于	键	描述	支持的值
			• **s390** - IBM Enterprise Systems Architecture/390 • **s390x** - S/390 64-bit • **sh4** - SuperH SH-4 (Little Endian) • **sh4eb** - SuperH SH-4 (Big Endian) • **SPARC -** 可扩展处理器架构, 32 位 • **sparc64** - 可扩展处理器架构, 64 位 • **unicore32** - Microprocessor research and Development Center RISC Unicore32 • **x86_64** - IA-32 的 64 位扩展 • **xtensa** - Tensilica Xtensa 可配置微处理器内核 • **xtensaeb** - Tensilica Xtensa 可配置微处理器内核 (Big Endian)
All	**hypervisor_type**	管理程序类型。	**KVM**,**vmware**
All	**instance_uuid**	对于快照镜像，这是用于创建此镜像的服务器的 UUID。	有效的服务器 UUID
All	**kernel_id**	镜像服务中存储的镜像的 ID，在引导 AMI 风格的镜像时应用作内核。	有效的镜像 ID
All	**os_distro**	操作系统分发的通用名称（小写）。	• **arch** - Arch Linux.不要使用 **archlinux** 或 **org.archlinux**。 • **CentOS** - 社区企业操作系统.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

特定于	键	描述	支持的值
			org.freebsd、freeBSD 或 FreeBSD。 • **gentoo** - Gentoo Linux.不要使用 **Gentoo** 或 **org.gentoo**。 • **mandrake** - Mandrakelinux (MandrakeSoft) distribution. 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 操作系统.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.openactivities**。 • **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 **规范**。 • **Windows** - Microsoft Windows.不要使用 **com.microsoft.server**。
All	**os_version**	由经销商指定的操作系统版本。	版本号（例如："11.10"）

特定于	键	描述	支持的值
All	**ramdisk_id**	镜像服务中存储的镜像的 ID，在引导 AMI 风格的镜像时应用作 ramdisk。	有效的镜像 ID
All	**vm_mode**	虚拟机模式。这代表了用于虚拟机的主机/客户机 ABI（应用程序二进制接口）。	**HVM**- 完全虚拟化.这是 QEMU 和 KVM 使用的模式。
libvirt API 驱动程序	**hw_cdrom_bus**	指定要将 CD-ROM 设备附加到的磁盘控制器类型。	**SCSI**、**virtio**、**ide** 或 **usb**。如果指定了 **iscsi**，您必须将 **hw_scsi_model** 参数设置为 **virtio-scsi**。
libvirt API 驱动程序	**hw_disk_bus**	指定要将磁盘设备附加到的磁盘控制器类型。	**SCSI**、**virtio**、**ide** 或 **usb**。请注意，如果使用 **iscsi**，则需要将 **hw_scsi_model** 设置为 **virtio-scsi**。
libvirt API 驱动程序	**hw_firmware_type**	指定用来引导实例的固件类型。	设置为以下有效值之一： • **BIOS** • **uefi**
libvirt API 驱动程序	**hw_machine_type**	启用使用指定的机器类型引导 ARM 系统。如果使用 ARM 镜像且其机器类型没有被明确指定，则 Compute 将使用 virt 机器类型作为 ARMv7 和 AArch64 的默认设置。	可以使用 **virsh capabilities** 命令查看有效的类型。机器类型显示在机器标签中。
libvirt API 驱动程序	**hw_numa_nodes**	要公开给实例的 NUMA 节点数（不覆盖类别定义）。	整数.
libvirt API 驱动程序	**hw_numa_cpus.0**	vCPU N-M 映射到 NUMA 节点 0 （不覆盖类别定义）。	以逗号分隔的整数列表。
libvirt API 驱动程序	**hw_numa_cpus.1**	vCPU N-M 映射到 NUMA 节点 1（不覆盖类别定义）。	以逗号分隔的整数列表。
libvirt API 驱动程序	**hw_numa_mem.0**	将 N MB RAM 映射到 NUMA 节点 0 （不覆盖类别定义）。	整数

特定于	键	描述	支持的值
libvirt API 驱动程序	**hw_numa_mem.1**	将 N MB RAM 映射到 NUMA 节点 1（不覆盖类别定义）。	整数

libvirt API 驱动程序	**hw_pci_numa_affinity_policy**	指定 PCI 透传设备和 SR-IOV 接口的 NUMA 关联性策略。	设置为以下有效值之一：
			• **必需**：计算服务会创建一个实例，只有在实例的至少一个 NUMA 节点与 PCI 设备关联时才请求 PCI 设备。这个选项提供最佳性能。
			• **preferred**：计算服务会尝试根据 NUMA 关联性选择 PCI 设备。如果关联性不可能，则计算服务会将实例调度到没有与 PCI 设备关联性的 NUMA 节点上。
			• **传统**：（默认）计算服务创建请求 PCI 设备的实例：
			  ○ PCI 设备与至少一个 NUMA 节点关联。
			  ○ PCI 设备不提供有关其 NUMA 事务的信息。

libvirt API 驱动程序	**hw_qemu_guest_agent**	客户机代理支持。如果设置为 **yes**，并且安装了 **qemu-ga**，则可以静止(frozen)和快照自动创建。	**是 / no**
libvirt API 驱动程序	**hw_rng_model**	向使用此镜像启动的实例添加一个随机数生成器(RNG)设备。 instance 类别默认启用 RNG 设备。要禁用 RNG 设备，云管理员必须在类别上将 **hw_rng:allowed** 设置为 **False**。 默认熵源为 **/dev/random**。要指定硬件 RNG 设备，请在 Compute 环境文件中将 **rng_dev_path** 设置为 **/dev/hwrng**。	**VirtIO** 或其他支持的设备。

特定于	键	描述	支持的值
libvirt API 驱动程序	**hw_scsi_model**	启用 VirtIO SCSI (virtio-scsi)来为计算实例提供块设备访问；默认情况下，实例使用 VirtIO 块(virtio-blk)。VirtIO SCSI 是一个半虚拟化 SCSI 控制器设备，可提供更高的可扩展性和性能，并支持高级 SCSI 硬件。	**virtio-scsi**
libvirt API 驱动程序	**hw_tpm_model**	设置为要使用的 TPM 设备模型。如果没有配置 **hw:tpm_version**，则忽略。	• TPM **-tis**：（默认） TPM 接口规格。 • **TPM-crb** ：命令响应缓冲. 仅与 TPM 版本 2.0 兼容。
libvirt API 驱动程序	**hw_tpm_version**	设置为要使用的 TPM 版本。TPM 版本 **2.0** 是唯一受支持的版本。	**2.0**

特定于	键	描述	支持的值
libvirt API 驱动程序	**hw_video_model**	在虚拟机实例中使用的显示设备的视频设备驱动程序。	设置为以下值之一以指定要使用的支持的驱动程序： • **virtio** - （默认）虚拟机显示设备 **的建议** 驱动程序，大多数架构支持。VirtIO GPU 驱动程序包含在 RHEL-7 及更高版本中，Linux 内核版本 4.4 及更新的版本。如果实例内核具有 VirtIO GPU 驱动程序，则实例可以使用所有 VirtIO GPU 功能。如果实例内核没有 VirtIO GPU 驱动程序，则 VirtIO GPU 设备会正常回退到 VGA 兼容模式，该模式为实例提供可正常工作的显示。 • **QXL** - **已弃用** 驱动程序的 Spice 或 noVNC 环境，这些环境不再被维护。 • **Cirrus** - 传统驱动程序，仅支持向后兼容。不要将 用作新实例。 • **VGA** - 将这个驱动程序用于 IBM Power 环境。 • **gop** - 不支持 QEMU/KVM 环境。 • **Xen** - KVM 环境不支持. • **vmvga** - 旧的驱动程序，不使用。 • **none** - 使用这个值在单独配置驱动程序的虚拟 GPU (vGPU)实例中禁用模拟图形或视频。
libvirt API 驱动程序	**hw_video_ram**	视频镜像的最大 RAM。仅在类别 **extra_specs** 和该值高于 **hw_video_ram** 中设置的值时，才使用 **hw_video:ram_max_mb** 值。	以 MB 为单位的整数（例如 *64*）

特定于	键	描述	支持的值
libvirt API 驱动程序	**hw_watchdog_action**	启用在服务器挂起时执行指定操作的虚拟硬件 watchdog 设备。watchdog 使用 i6300esb 设备（模拟 PCI Intel 6300ESB）。如果没有指定 **hw_watchdog_action**，则禁用 watchdog。	• disabled- 该设备没有附加。允许用户禁用镜像的 watchdog，即使它已使用镜像的类别启用。这个参数的默认值被禁用。 • reset-Forcefully 重置 guest。 • poweroff-Forcefully 关闭客户机。 • 暂停客户机。 • none-Only 启用 watchdog；如果服务器挂起，不执行任何操作。
libvirt API 驱动程序	**os_command_line**	**libvirt** 驱动程序使用的内核命令行，而不是默认值。对于 Linux 容器(LXC)，该值用作初始化的参数。这个密钥仅对 Amazon 内核、ramdisk 或机器镜像(aki、ari 或 ami)有效。	
libvirt API 驱动程序	**os_secure_boot**	使用 创建通过 UEFI 安全引导保护的实例。	设置为以下有效值之一： • **必需** ：为使用此镜像启动的实例启用安全引导。只有计算服务找到可以支持安全引导的主机时，才会启动该实例。如果没有找到主机，Compute 服务会返回 "No valid host" 错误。 • **disabled** ：为使用此镜像启动的实例禁用安全引导。默认禁用此选项。 • **可选** ：只有在计算服务确定主机可以支持安全引导机制时，才会为使用此镜像启动的实例启用安全引导。

特定于	键	描述	支持的值
libvirt API 驱动程序和 VMware API 驱动程序	**hw_vif_model**	指定要使用的虚拟网络接口设备模型。	有效选项取决于配置的 hypervisor。 • KVM 和 QEMU：e1000、ne2k_pci、pcnet、rtl8139 和 virtio。 • VMware：e1000、e1000e、VirtualE1000e、VirtualE1000e、VirtualPCNet32, VirtualSriovEthernetCard 和 VirtualVmxnet. • Xen: e1000, netfront, ne2k_pci, pcnet, and rtl8139.
VMware API 驱动程序	**vmware_adaptertype**	管理程序使用的虚拟 SCSI 或 IDE 控制器。	**lsiLogic**、**busLogic** 或 **ide**
VMware API 驱动程序	**vmware_ostype**	描述镜像中安装的操作系统的 VMware GuestID。这个值在创建虚拟机时传递给虚拟机监控程序。如果没有指定，则密钥默认为 **otherGuest**。	如需更多信息，请参阅使用 VMware vSphere 的镜像。
VMware API 驱动程序	**vmware_image_version**	当前未使用。	**1**
XenAPI 驱动程序	**auto_disk_config**	如果为 true，则在实例引导前，磁盘上的根分区会自动调整大小。只有在使用带有 XenAPI 驱动程序的基于 Xen 的 hypervisor 时，Compute 服务才会考虑这个值。只有镜像上只有一个分区，并且仅当分区采用 **ext3** 或 **ext4** 格式时，计算服务才会尝试调整大小。	**true** / **false**

特定于	键	描述	支持的值
libvirt API 驱动程序和 XenAPI 驱动程序	**os_type**	在镜像上安装的操作系统。XenAPI 驱动程序包含根据镜像的 **os_type** 参数的值采取不同的操作的逻辑。例如，对于 **os_type=windows** 镜像，它会创建一个基于 FAT32 的交换分区，而不是 Linux swap 分区，它会将注入的主机名限制为小于 16 个字符。	**linux** 或 **windows**