



Red Hat OpenStack Platform 17.1

Creating and managing images

Create and manage images in Red Hat OpenStack Platform by using the Image service (glance)

Red Hat OpenStack Platform 17.1 Creating and managing images

Create and manage images in Red Hat OpenStack Platform by using the Image service (glance)

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Copyright © 2024 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This guide provides procedures for creating and managing images, and procedures for configuring the Image service (glance).

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE	4
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	5
CHAPTER 1. THE IMAGE SERVICE (GLANCE)	6
1.1. VIRTUAL MACHINE IMAGE FORMATS	6
1.2. SUPPORTED IMAGE SERVICE BACK ENDS	7
1.3. IMAGE SIGNING AND VERIFICATION	8
1.4. IMAGE FORMAT CONVERSION	8
1.5. IMPROVING SCALABILITY WITH IMAGE SERVICE CACHING	9
1.6. IMAGE PRE-CACHING	9
1.6.1. Configuring the default interval for periodic image pre-caching	9
1.6.2. Preparing to use a periodic job to pre-cache an image	10
1.6.3. Using a periodic job to pre-cache an image	12
1.6.4. Image caching command options	12
1.7. USING THE IMAGE SERVICE API TO ENABLE SPARSE IMAGE UPLOAD	13
1.8. SECURE METADEF APIS	15
1.9. ENABLING METADEF API ACCESS FOR CLOUD USERS	15
CHAPTER 2. CREATING RHEL KVM OR RHOSP-COMPATIBLE IMAGES	18
2.1. CREATING RHEL KVM IMAGES	18
2.1.1. Using a RHEL KVM instance image with Red Hat OpenStack Platform	18
2.1.2. Creating a RHEL-based root partition image for bare-metal instances	18
2.1.3. Creating a RHEL-based whole-disk user image for bare-metal instances	20
2.2. CREATING INSTANCE IMAGES WITH RHEL OR WINDOWS ISO FILES	21
2.2.1. Prerequisites	21
2.2.2. Creating a Red Hat Enterprise Linux 9 image	21
2.2.3. Creating a Red Hat Enterprise Linux 8 image	24
2.2.4. Creating a Windows image	27
2.3. CREATING AN IMAGE FOR UEFI SECURE BOOT	28
2.4. METADATA PROPERTIES FOR VIRTUAL HARDWARE	29
CHAPTER 3. MANAGING IMAGES, IMAGE PROPERTIES, AND IMAGE FORMATS	30
3.1. UPLOADING IMAGES TO THE IMAGE SERVICE	30
3.2. IMAGE SERVICE IMAGE IMPORT METHODS	30
3.2.1. Importing an image from a remote URI	31
3.2.2. Importing an image from a local volume	31
3.3. UPDATING IMAGE PROPERTIES	32
3.4. ENABLING IMAGE CONVERSION	33
3.4.1. Converting an image to RAW format	33
3.4.2. Configuring disk formats with the GlanceDiskFormats parameter	34
3.4.3. Storing an image in RAW format	35
3.5. HIDING OR UNHIDING IMAGES	35
3.6. DELETING IMAGES FROM THE IMAGE SERVICE	36
CHAPTER 4. CONFIGURING THE IMAGE SERVICE IMAGE IMPORT METHOD	37
4.1. CONFIGURING THE GLANCE-DIRECT IMAGE IMPORT METHOD	37
4.2. CONTROLLING IMAGE WEB-IMPORT SOURCES	38
4.2.1. Image import allowlist example	39
4.2.2. Default image import blocklist and allowlist settings	39
4.3. INJECTING METADATA ON IMAGE IMPORT TO CONTROL WHERE INSTANCES LAUNCH	40
CHAPTER 5. IMAGE SERVICE WITH MULTIPLE STORES	41

5.1. IMAGE COPIES ON MULTIPLE STORES	41
5.2. REQUIREMENTS OF STORAGE EDGE ARCHITECTURE	41
5.3. IMPORTING AN IMAGE TO MULTIPLE STORES	41
5.3.1. Managing image import failures	42
5.3.2. Importing image data to multiple stores	42
5.3.3. Importing image data to multiple stores without failure	43
5.3.4. Importing image data to a single store	43
5.4. CHECKING THE PROGRESS OF THE IMAGE IMPORT OPERATION	44
5.5. COPYING AN EXISTING IMAGE TO MULTIPLE STORES	45
5.5.1. Copying an image to all stores	46
5.5.2. Copying an image to specific stores	46
5.6. DELETING AN IMAGE FROM A SPECIFIC STORE	46
5.7. LISTING IMAGE LOCATIONS AND LOCATION PROPERTIES	47
APPENDIX A. IMAGE SERVICE COMMAND OPTIONS	49
APPENDIX B. IMAGE CONFIGURATION PARAMETERS	51

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see [our CTO Chris Wright's message](#).

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Tell us how we can make it better.

Providing documentation feedback in Jira

Use the [Create Issue](#) form to provide feedback on the documentation. The Jira issue will be created in the Red Hat OpenStack Platform Jira project, where you can track the progress of your feedback.

1. Ensure that you are logged in to Jira. If you do not have a Jira account, create an account to submit feedback.
2. Click the following link to open a the **Create Issue** page: [Create Issue](#)
3. Complete the **Summary** and **Description** fields. In the **Description** field, include the documentation URL, chapter or section number, and a detailed description of the issue. Do not modify any other fields in the form.
4. Click **Create**.

CHAPTER 1. THE IMAGE SERVICE (GLANCE)

The Image service (glance) provides discovery, registration, and delivery services for disk and server images. It provides the ability to copy or snapshot a server image, and immediately store it. You can use stored images as templates to commission new servers quickly and more consistently than installing a server operating system and individually configuring services.

1.1. VIRTUAL MACHINE IMAGE FORMATS

A virtual machine (VM) image is a file that contains a virtual disk with a bootable OS installed. Red Hat OpenStack Platform (RHOSP) supports VM images in different formats.

The disk format of a VM image is the format of the underlying disk image. The container format indicates if the VM image is in a file format that also contains metadata about the VM.

When you add an image to the Image service (glance), you can set the disk or container format for your image to any of the values in the following tables by using the **--disk-format** and **--container-format** command options with the **glance image-create**, **glance image-create-via-import**, and **glance image-update** commands. If you are not sure of the container format of your VM image, you can set it to **bare**.

Table 1.1. Disk image formats

Format	Description
aki	Indicates an Amazon kernel image that is stored in the Image service.
ami	Indicates an Amazon machine image that is stored in the Image service.
ari	Indicates an Amazon ramdisk image that is stored in the Image service.
iso	Sector-by-sector copy of the data on a disk, stored in a binary file. Although an ISO file is not normally considered a VM image format, these files contain bootable file systems with an installed operating system, and you use them in the same way as other VM image files.
ploop	A disk format supported and used by Virtuozzo to run OS containers.
qcow2	Supported by QEMU emulator. This format includes QCOW2v3 (sometimes referred to as QCOW3), which requires QEMU 1.1 or higher.
raw	Unstructured disk image format.
vdi	Supported by VirtualBox VM monitor and QEMU emulator.
vhd	Virtual Hard Disk. Used by VM monitors from VMware, VirtualBox, and others.
vhdx	Virtual Hard Disk v2. Disk image format with a larger storage capacity than VHD.
vmdk	Virtual Machine Disk. Disk image format that allows incremental backups of data changes from the time of the last backup.

Table 1.2. Container image formats

Format	Description
aki	Indicates an Amazon kernel image that is stored in the Image service.
ami	Indicates an Amazon machine image that is stored in the Image service.
ari	Indicates an Amazon ramdisk image that is stored in the Image service.
bare	Indicates there is no container or metadata envelope for the image.
docker	Indicates a TAR archive of the file system of a Docker container that is stored in the Image service.
ova	Indicates an Open Virtual Appliance (OVA) TAR archive file that is stored in the Image service. This file is stored in the Open Virtualization Format (OVF) container file.
ovf	OVF container file format. Open standard for packaging and distributing virtual appliances or software to be run on virtual machines.

1.2. SUPPORTED IMAGE SERVICE BACK ENDS

The following Image service (glance) back-end scenarios are supported:

- RADOS Block Device (RBD) is the default back end when you use Ceph.
- RBD multi-store.
- Object Storage (swift). The Image service uses the Object Storage type and back end as the default.
- Block Storage (cinder).
- NFS

Important

Although NFS is a supported Image service deployment option, more robust options are available.

NFS is not native to the Image service. When you mount an NFS share on the Image service, the Image service does not manage the operation. The Image service writes data to the file system but is unaware that the back end is an NFS share.

In this type of deployment, the Image service cannot retry a request if the share fails. This means that when a failure occurs on the back end, the store might enter read-only mode, or it might continue to write data to the local file system, in which case you risk data loss. To recover from this situation, you must ensure that the share is mounted and in sync, and then restart the Image service. For these reasons, Red Hat does not recommend NFS as an Image service back end.

However, if you do choose to use NFS as an Image service back end, some of the following best practices can help to mitigate risks:

- Use a reliable production-grade NFS back end.
- Ensure that you have a strong and reliable connection between Controller nodes and the NFS back end: Layer 2 (L2) network connectivity is recommended.
- Include monitoring and alerts for the mounted share.
- Set underlying file system permissions. Write permissions must be present in the shared file system that you use as a store.
- Ensure that the user and the group that the glance-api process runs on do not have write permissions on the mount point at the local file system. This means that the process can detect possible mount failure and put the store into read-only mode during a write attempt.

1.3. IMAGE SIGNING AND VERIFICATION

Image signing and verification protects image integrity and authenticity by enabling deployers to sign images and save the signatures and public key certificates as image properties.



NOTE

Image signing and verification is not supported if Nova is using RADOS Block Device (RBD) to store virtual machines disks.

For information on image signing and verification, see [Validating Image service \(glance\) images](#) in the *Managing secrets with the Key Manager service* guide.

1.4. IMAGE FORMAT CONVERSION

You can convert images to a different format by activating the image conversion plugin when you import images to the Image service (glance).

You can activate or deactivate the image conversion plugin based on your Red Hat OpenStack Platform (RHOSP) deployment configuration. The deployer configures the preferred format of images for the deployment.

Internally, the Image service receives the bits of the image in a particular format and stores the bits in a temporary location. The Image service triggers the plugin to convert the image to the target format and move the image to a final destination. When the task is finished, the Image service deletes the temporary location. The Image service does not retain the format that was uploaded initially.

You can trigger image conversion only when importing an image. It does not run when uploading an image.

Use the Image service command-line client for image management.

For example:

```
$ glance image-create-via-import \  
--disk-format qcow2 \  

```

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

- Replace **<name>** with the name of your image.

1.5. IMPROVING SCALABILITY WITH IMAGE SERVICE CACHING

Use the Image service (glance) API caching mechanism to store copies of images on Image service API servers and retrieve them automatically to improve scalability. With Image service caching, you can run glance-api on multiple hosts. This means that it does not need to retrieve the same image from back-end storage multiple times. Image service caching does not affect any Image service operations.

Configure Image service caching with the Red Hat OpenStack Platform director (tripleo) heat templates:

Procedure

1. In an environment file, set the value of the **GlanceCacheEnabled** parameter to **true**, which automatically sets the **flavor** value to **keystone+cachemanagement** in the **glance-api.conf** heat template:

```
parameter_defaults:
  GlanceCacheEnabled: true
```

2. Include the environment file in the **openstack overcloud deploy** command when you redeploy the overcloud.
3. Optional: Tune the **glance_cache_pruner** to an alternative frequency when you redeploy the overcloud. The following example shows a frequency of 5 minutes:

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

Adjust the frequency according to your needs to avoid file system full scenarios. Include the following elements when you choose an alternative frequency:

- The size of the files that you want to cache in your environment.
- The amount of available file system space.
- The frequency at which the environment caches images.

1.6. IMAGE PRE-CACHING

You can use Red Hat OpenStack Platform (RHOSP) director to pre-cache images as part of the **glance-api** service.

Use the Image service (glance) command-line client for image management.

1.6.1. Configuring the default interval for periodic image pre-caching

The Image service (glance) pre-caching periodic job runs every 300 seconds (5 minutes default time) on each controller node where the **glance-api** service is running. To change the default time, you can set the **cache_prefetcher_interval** parameter under the **Default** section in the `glance-api.conf` environment file.

Procedure

1. Add a new interval with the **ExtraConfig** parameter in an environment file on the undercloud according to your requirements:

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

- Replace **<300>** with the number of seconds that you want as an interval to pre-cache images.
2. After you adjust the interval in the environment file in `/home/stack/templates/`, log in as the **stack** user and deploy the configuration:

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

- Replace `<env_file>` with the name of the environment file that contains the **ExtraConfig** settings that you added.



IMPORTANT

If you passed any extra environment files when you created the overcloud, pass them again here by using the **-e** option to avoid making undesired changes to the overcloud.

Additional resources

For more information about the **openstack overcloud deploy** command, see [Deployment command](#) in the *Installing and managing Red Hat OpenStack Platform with director* guide.

1.6.2. Preparing to use a periodic job to pre-cache an image

To use a periodic job to pre-cache an image, you must use the **glance-cache-manage** command connected directly to the node where the **glance-api** service is running. Do not use a proxy, which hides the node that answers a service request. Because the undercloud might not have access to the network where the **glance-api** service is running, run commands on the first overcloud node, which is called **controller-0** by default.

Complete the following prerequisite procedure to ensure that you run commands from the correct host, have the necessary credentials, and are also running the **glance-cache-manage** commands from inside the **glance-api** container.

Procedure

1. Log in to the undercloud as the stack user and identify the provisioning IP address of **controller-0**:

```
(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. To authenticate to the overcloud, copy the credentials that are stored in **/home/stack/overcloudrc**, by default, to **controller-0**:

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

3. Connect to **controller-0**:

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

4. On **controller-0** as the **tripleo-admin** user, identify the IP address of the **glance_api** service. In the following example, the IP address is **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. Because the **glance-cache-manage** command is only available in the **glance_api** container, create a script to exec into that container where the environment variables to authenticate to the overcloud are already set. Create a script called **glance_pod.sh** in **/home/tripleo-admin** on **controller-0** with the following contents:

```
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 the **overcloudrc** file and run the **glance_pod.sh** script to exec into the **glance_api** container with the necessary environment variables to authenticate to the overcloud Controller node.

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

7. Use a command such as **glance image-list** to verify that the container can run authenticated commands against the overcloud.

```
()[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. Using a periodic job to pre-cache an image

When you have completed the prerequisite procedure in [Section 1.6.2, "Preparing to use a periodic job to pre-cache an image"](#), you can use a periodic job to pre-cache an image.

Procedure

1. As the admin user, queue an image to cache:

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

- Replace **<host_ip>** with the IP address of the Controller node where the **glance-api** container is running.
- Replace **<image_id>** with the ID of the image that you want to queue. When you have queued the images that you want to pre-cache, the **cache_images** periodic job prefetches all queued images concurrently.



NOTE

Because the image cache is local to each node, if your Red Hat OpenStack Platform (RHOSP) deployment is HA, with 3, 5, or 7 Controllers, then you must specify the host address with the **--host** option when you run the **glance-cache-manage** command.

2. Run the following command to view the images in the image cache:

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

- Replace **<host_ip>** with the IP address of the host in your environment.

1.6.4. Image caching command options

You can use the following **glance-cache-manage** command options to queue images for caching and manage cached images:

- **list-cached** to list all images that are currently cached.
- **list-queued** to list all images that are currently queued for caching.
- **queue-image** to queue an image for caching.
- **delete-cached-image** to purge an image from the cache.
- **delete-all-cached-images** to remove all images from the cache.
- **delete-queued-image** to delete an image from the cache queue.
- **delete-all-queued-images** to delete all images from the cache queue.

1.7. USING THE IMAGE SERVICE API TO ENABLE SPARSE IMAGE UPLOAD

With the Image service (glance) API, you can use sparse image upload to reduce network traffic and save storage space. This feature is particularly useful in distributed compute node (DCN) environments. With a sparse image file, the Image service does not write null byte sequences. The Image service writes data with a given offset. Storage back ends interpret these offsets as null bytes that do not actually consume storage space.

Use the Image service command-line client for image management.

Limitations

- Sparse image upload is supported only with Ceph RADOS Block Device (RBD).
- Sparse image upload is not supported for file systems.
- Sparseness is not maintained during the transfer between the client and the Image service API. The image is sparsed at the Image service API level.

Prerequisites

- Your Red Hat OpenStack Platform (RHOSP) deployment uses RBD for the Image service back end.

Procedure

1. Log in to the undercloud node as the **stack** user.
2. Source the **stackrc** credentials file:

```
$ source stackrc
```

3. Create an environment file with the following content:

```
parameter_defaults:
  GlanceSparseUploadEnabled: true
```

4. Add your new environment file to the stack with your other environment files and deploy the overcloud:

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

For more information about uploading images, see [Uploading images to the Image service](#).

Verification

You can import an image and check its size to verify sparse image upload.

The following procedure uses example commands. Replace the values with those from your environment where appropriate.

1. Download the image file locally:

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

- Replace **<file_location>** with the location of the file.
- Replace **<file_name>** with the name of the file.
For example:

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

2. Check the disk size and the virtual size of the image to be uploaded:

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

For example:

```
$ 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. Import the image:

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

4. Record the image ID. It is required in a subsequent step.

5. Verify that the image is imported and in an active state:

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

6. From a Ceph Storage node, verify that the size of the image is less than the virtual size from the output of step 1:

```
$ sudo rbd -p images diff <image_id> | awk '{ SUM += $2 } END { print SUM/1024/1024/1024
" GB" }'
1.03906 GB
```

7. Optional: You can confirm that **rbd_thin_provisioning** is configured in the Image service configuration file on the Controller nodes:

- a. Use SSH to access a Controller node:

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

- b. Confirm that **rbd_thin_provisioning** equals **True** on that Controller node:

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

1.8. SECURE METADEF APIS

In Red Hat OpenStack Platform (RHOSP), cloud administrators can define key value pairs and tag metadata with metadata definition (metadef) APIs. There is no limit on the number of metadef namespaces, objects, properties, resources, or tags that cloud administrators can create.

Image service policies control metadef APIs. By default, only cloud administrators can create, update, or delete (CUD) metadef APIs. This limitation prevents metadef APIs from exposing information to unauthorized users and mitigates the risk of a malicious user filling the Image service (glance) database with unlimited resources, which can create a Denial of Service (DoS) style attack. However, cloud administrators can override the default policy.

1.9. ENABLING METADEF API ACCESS FOR CLOUD USERS

Cloud administrators with users who depend on write access to metadata definition (metadef) APIs can make those APIs accessible to all users by overriding the default admin-only policy. In this type of configuration, however, there is the potential to unintentionally leak sensitive resource names, such as customer names and internal projects. Administrators must audit their systems to identify previously created resources that might be vulnerable even if only read-access is enabled for all users.

Procedure

1. As a cloud administrator, log in to the undercloud and create a file for policy overrides. For example:

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

2. Configure the policy override file to allow metadef API read-write access to all users:

```

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' }
}

```



NOTE

You must configure all metadef policies to use **rule:metadef_default**. For information about policies and policy syntax, see this [Policies](#) chapter.

3. Include the new policy file in the deployment command with the **-e** option when you deploy the overcloud:

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

■

CHAPTER 2. CREATING RHEL KVM OR RHOSP-COMPATIBLE IMAGES

To create images that you can manage in the Red Hat OpenStack Platform (RHOSP) Image service (glance), you can use Red Hat Enterprise Linux (RHEL) Kernel-based Virtual Machine (KVM) instance images, or you can manually create RHOSP-compatible images in the QCOW2 format by using RHEL ISO files or Windows ISO files.

2.1. CREATING RHEL KVM IMAGES

Use Red Hat Enterprise Linux (RHEL) Kernel-based Virtual Machine (KVM) instance images to create images that you can manage in the Red Hat OpenStack Platform (RHOSP) Image service (glance).

2.1.1. Using a RHEL KVM instance image with Red Hat OpenStack Platform

You can use one of the following Red Hat Enterprise Linux (RHEL) Kernel-based Virtual Machine (KVM) instance images with Red Hat OpenStack Platform (RHOSP):

- [Red Hat Enterprise Linux 9 KVM Guest Image](#)
- [Red Hat Enterprise Linux 8 KVM Guest Image](#)

These QCOW2 images are configured with **cloud-init** and must have EC2-compatible metadata services for provisioning Secure Shell (SSH) keys to function correctly.

Ready Windows KVM instance images in QCOW2 format are not available.



NOTE

For KVM instance images:

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

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

For a RHOSP instance, generate an SSH keypair from the RHOSP dashboard or command line, and use that key combination to perform an SSH public authentication to the instance as root user.

When you launch the instance, this public key is injected to it. You can then authenticate by using the private key that you download when you create the keypair.

2.1.2. Creating a RHEL-based root partition image for bare-metal instances

To create a custom root partition image for bare-metal instances, download the base Red Hat Enterprise Linux KVM instance image, and then upload the image to the Image service (glance).

Procedure

1. Download the base Red Hat Enterprise Linux KVM instance image from the [Customer Portal](#).
2. Define **DIB_LOCAL_IMAGE** as the downloaded image:

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

- Replace **<ver>** with the RHEL version number of the image.

3. Set your registration information depending on your method of registration:

- Red Hat Customer Portal:

```
$ 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>
```

- Replace values in angle brackets **<>** with the correct values for your Red Hat Customer Portal or Red Hat Satellite registration.

4. Optional: If you have any offline repositories, you can define **DIB_YUM_REPO_CONF** as a local repository configuration:

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

- Replace **<file-path>** with the path to your local repository configuration file.

5. Use the **diskimage-builder** tool to extract the kernel as **rhel-image.vmlinuz** and the initial RAM disk as **rhel-image.initrd**:

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

6. Upload the images to the Image service:

```
$ 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. Creating a RHEL-based whole-disk user image for bare-metal instances

To create a whole-disk user image for bare-metal instances, download the base Red Hat Enterprise Linux KVM instance image, and then upload the image to the Image service (glance).

Procedure

1. Download the base Red Hat Enterprise Linux KVM instance image from the [Customer Portal](#).
2. Define **DIB_LOCAL_IMAGE** as the downloaded image:

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

- Replace **<ver>** with the RHEL version number of the image.

3. Set your registration information depending on your method of registration:

- Red Hat Customer Portal:

```
$ 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>
```

- Replace values in angle brackets **<>** with the correct values for your Red Hat Customer Portal or Red Hat Satellite registration.

4. Optional: If you have any offline repositories, you can define **DIB_YUM_REPO_CONF** as a local repository configuration:

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

- Replace **<file-path>** with the path to your local repository configuration file.

5. Upload the image to the Image service:

```
$ openstack image create \
--file rhel-image.qcow2 --public \
```



```
--container-format bare \
--disk-format qcow2 \
rhel-whole-disk-bare-metal-image
```

2.2. CREATING INSTANCE IMAGES WITH RHEL OR WINDOWS ISO FILES

You can create custom Red Hat Enterprise Linux (RHEL) or Windows images in QCOW2 format from ISO files, and upload these images to the Red Hat OpenStack Platform (RHOSP) Image service (glance) for use when creating instances.

2.2.1. Prerequisites

- A Linux host machine to create an image. This can be any machine on which you can install and run the Linux packages, except for the undercloud or the overcloud.
- The **advanced-virt** repository is enabled:

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

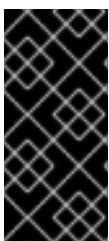
- The **virt-manager** application is installed to have all packages necessary to create a guest operating system:

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

- The **libguestfs-tools** package is installed to have a set of tools to access and modify virtual machine images:

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

- A RHEL 9 or 8 ISO file or a Windows ISO file. For more information about RHEL ISO files, see [RHEL 9.0 Binary DVD](#) or [RHEL 8.6 Binary DVD](#). If you do not have a Windows ISO file, see the [Microsoft Evaluation Center](#) to download an evaluation image.
- A text editor, if you want to change the **kickstart** files (RHEL only).



IMPORTANT

If you install the **libguestfs-tools** package on the undercloud, deactivate **iscsid.socket** to avoid port conflicts with the **tripleo_iscsid** service on the undercloud:

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

When you have the prerequisites in place, you can proceed to create a RHEL or Windows image:

- [Create a Red Hat Enterprise Linux 9 image](#)
- [Create a Red Hat Enterprise Linux 8 image](#)
- [Create a Windows image](#)

2.2.2. Creating a Red Hat Enterprise Linux 9 image

You can create a Red Hat OpenStack Platform (RHOSP) image in QCOW2 format by using a Red Hat Enterprise Linux (RHEL) 9 ISO file.

Procedure

1. Log on to your host machine as the **root** user.
2. Start the installation by using **virt-install**:

```
[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>
```

- Replace the values in angle brackets **<>** with the correct values for your RHEL 9 image. This command launches an instance and starts the installation process.



NOTE

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

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

3. Configure the instance:
 - a. At the initial Installer boot menu, select **Install Red Hat Enterprise Linux 9**
 - b. Choose the appropriate **Language** and **Keyboard** options.
 - c. When prompted about which type of devices your installation uses, select **Auto-detected installation media**.
 - d. When prompted about which type of installation destination, select **Local Standard Disks**. For other storage options, select **Automatically configure partitioning**.
 - e. In the **Which type of installation would you like?** window, choose the **Basic Server** install, which installs an SSH server.
 - f. For network and host name, select **eth0** for network and choose a host name for your device. The default host name is **localhost.localdomain**.
 - g. Enter a password in the **Root Password** field and enter the same password again in the **Confirm** field.
4. When the on-screen message confirms that the installation is complete, reboot the instance and log in as the root user.

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

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

- Reboot the machine.
- Register the machine with the Content Delivery Network.

```
# 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
```

- Replace **pool-id** with a valid pool ID. You can see a list of available pool IDs by running the **subscription-manager list --available** command.

- Update the system:

```
# dnf -y update
```

- Install the **cloud-init** packages:

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

- Edit the `/etc/cloud/cloud.cfg` configuration file and add the following content under **cloud_init_modules**:

```
- resolv-conf
```

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

- Add the following line to `/etc/sysconfig/network` to avoid issues when accessing the EC2 metadata service:

```
NOZEROCONF=yes
```

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

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

- Run the **grub2-mkconfig** command:

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

The output is as follows:

```
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-229.9.2.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. Deregister the instance so that the resulting image does not contain the subscription details for this instance:

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

15. Power off the instance:

```
# poweroff
```

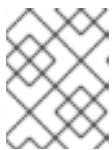
16. Reset and clean the image by using the **virt-sysprep** command so that it can be used to create instances without issues:

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

17. Reduce the image size by converting any free space in the disk image back to free space in the host:

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

This command creates a new **<rhel9-cloud.qcow2>** file in the location from where the command is run.



NOTE

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

The **<rhel9-cloud.qcow2>** image file is ready to be uploaded to the Image service. For more information about uploading this image to your RHOSP deployment, see [Uploading images to the Image service](#).

2.2.3. Creating a Red Hat Enterprise Linux 8 image

You can create a Red Hat OpenStack Platform (RHOSP) image in QCOW2 format by using a Red Hat Enterprise Linux (RHEL) 8 ISO file.

Procedure

1. Log on to your host machine as the **root** user.

2. Start the installation by using **virt-install**:

```
[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>
```

- Replace the values in angle brackets `<>` with the correct values for your RHEL image. This command launches an instance and starts the installation process.



NOTE

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

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

3. Configure the instance:
 - a. At the initial Installer boot menu, select **Install Red Hat Enterprise Linux 8**
 - b. Choose the appropriate **Language** and **Keyboard** options.
 - c. When prompted about which type of devices your installation uses, select **Basic Storage Devices**.
 - d. Choose a host name for your device. The default host name is **localhost.localdomain**.
 - e. Set the **timezone** and **root** password.
 - f. In the **Which type of installation would you like?** window, choose the **Basic Server** install, which installs an SSH server.
4. When the on-screen message confirms that the installation is complete, reboot the instance and log in as the root user.
5. Update the `/etc/sysconfig/network-scripts/ifcfg-eth0` file so that it contains only the following values:

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

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

```
# 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
```

- Replace **pool-id** with a valid pool ID. You can see a list of available pool IDs by running the **subscription-manager list --available** command.

8. Update the system:

```
# dnf -y update
```

9. Install the **cloud-init** packages:

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

10. Edit the **/etc/cloud/cloud.cfg** configuration file and add the following content under **cloud_init_modules**.

```
- resolv-conf
```

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

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

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

This prevents the **/etc/udev/rules.d/70-persistent-net.rules** file from being created. If the **/etc/udev/rules.d/70-persistent-net.rules** file is created, networking might not function correctly when you boot from snapshots because the network interface is created as **eth1** instead of **eth0** and the IP address is not assigned.

12. Add the following line to **/etc/sysconfig/network** to avoid issues when accessing the EC2 metadata service:

```
NOZEROCONF=yes
```

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

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

14. Deregister the instance so that the resulting image does not contain the same subscription details for this instance:

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

15. Power off the instance:

```
# poweroff
```

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

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

17. Reduce the image size by converting any free space in the disk image back to free space in the host:

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

This command creates a new **<rhel86-cloud.qcow2>** file in the location from where the command is run.



NOTE

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

The **<rhel86-cloud.qcow2>** image file is ready to be uploaded to the Image service. For more information about uploading this image to your RHOSP deployment, see [Uploading images to the Image service](#).

2.2.4. Creating a Windows image

You can create a Red Hat OpenStack Platform (RHOSP) image in QCOW2 format by using a Windows ISO file.

Procedure

1. Log on to your host machine as the **root** user.
2. Start the installation by using **virt-install**:

```
[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>
```

- Replace the values in angle brackets **<>** with the correct values for your Windows image.

**NOTE**

The `--os-type=windows` parameter ensures that the clock is configured correctly for the Windows instance and enables its Hyper-V enlightenment features. You must also set `os_type=windows` in the image metadata before uploading the image to the Image service (glance).

- The `virt-install` command saves the instance image as `/var/lib/libvirt/images/<windows-image>.qcow2` by default. If you want to keep the instance image elsewhere, change the parameter of the `--disk` option:

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

- Replace `<file-name>` with the name of the file that stores the instance image, and optionally its path. For example, `path=win8.qcow2,size=8` creates an 8 GB file named `win8.qcow2` in the current working directory.

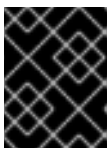
**NOTE**

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

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

For more information about how to install Windows, see the Microsoft documentation.

- To allow the newly-installed Windows system to use the virtualized hardware, you might need to install VirtIO drivers. For more information, see [Installing KVM paravirtualized drivers for Windows virtual machines](#) in *Configuring and managing virtualization*.
- To complete the configuration, download and run [Cloudbase-Init](#) on the Windows system. At the end of the installation of Cloudbase-Init, select the **Run Sysprep** and **Shutdown** checkboxes. The **Sysprep** tool makes the instance unique by generating an OS ID, which is used by certain Microsoft services.

**IMPORTANT**

Red Hat does not provide technical support for Cloudbase-Init. If you encounter an issue, see [Contact Cloudbase Solutions](#).

When the Windows system shuts down, the `<windows-image.qcow2>` image file is ready to be uploaded to the Image service. For more information about uploading this image to your RHOSP deployment, see [Uploading images to the Image service](#).

2.3. CREATING AN IMAGE FOR UEFI SECURE BOOT

When the overcloud contains UEFI Secure Boot Compute nodes, you can create a Secure Boot instance image that cloud users can use to launch Secure Boot instances.

Procedure

- Create a new image for UEFI Secure Boot:


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

- Replace **<base_image_file>** with an image file that supports UEFI and the GUID Partition Table (GPT) standard, and includes an EFI system partition.

2. If the default machine type is not **q35**, then set the machine type to **q35**:

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

3. Specify that the instance must be scheduled on a UEFI Secure Boot host:

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

2.4. METADATA PROPERTIES FOR VIRTUAL HARDWARE

The Compute service (nova) has deprecated support for using **libosinfo** data to set default device models. Instead, use the following image metadata properties to configure the optimal virtual hardware for an instance:

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

For more information about these metadata properties, see [Image configuration parameters](#).

CHAPTER 3. MANAGING IMAGES, IMAGE PROPERTIES, AND IMAGE FORMATS

Manage images and the properties and formats of images that you upload, import, or store in the Red Hat OpenStack Platform (RHOSP) Image service (glance).

3.1. UPLOADING IMAGES TO THE IMAGE SERVICE

You can upload an image to the Red Hat OpenStack Platform (RHOSP) Image service (glance) by using the **glance image-create** command with the **--property** option.

For a list of **glance image-create** command options, see [Image service command options](#). For a list of property keys, see [Image configuration parameters](#).

Procedure

- Use the **glance image-create** command with the **property** option to upload an image. For example:

```
$ glance image-create --name <name> \
  --is-public true --disk-format <qcow2> \
  --container-format <bare> \
  --file </path/to/image> \
  --property <os_version>=<11.10>
```

- Replace **<name>** with a descriptive name for your image.
- Replace **<disk-format>** with one of the following disk formats: None, ami, ari, aki, vhd, vhdx, vmdk, raw, qcow2, vdi, iso, ploop.
- Replace **<container-format>** with one of the following container formats: None, ami, ari, aki, bare, ovf, ova, docker.
- Replace **</path/to/image>** with the file path to your image file.
- Replace **<os_version>** and **<11.10>** with the key-value pair of the property you want to associate to your image. You can use the **--property** option multiple times with different key-value pairs you want to associate to your image.

3.2. IMAGE SERVICE IMAGE IMPORT METHODS

You can import images to the Red Hat OpenStack Platform (RHOSP) Image service (glance) by using the following methods:

- Use the **web-download** (default) method to import images from a URI.
- Use the **glance-direct** method to import images from a local file system.
- Use the **copy-image** method to copy an existing image to other Image service back ends that are in your deployment. Use this import method only if multiple Image service back ends are enabled in your deployment.

The **web-download** method is enabled by default, but the cloud administrator configures other import methods. You can run the **glance import-info** command to list available import options.

3.2.1. Importing an image from a remote URI

You can use the **web-download** image import method to copy an image from a remote URI to the Red Hat OpenStack Platform (RHOSP) Image service (glance).

The Image service **web-download** method uses a two-stage process to perform the import:

1. The **web-download** method creates an image record.
2. The **web-download** method retrieves the image from the specified URI.

The URI is subject to optional **denylist** and **allowlist** filtering.

The image property injection plugin may inject metadata properties to the image. These injected properties determine which Compute nodes the image instances are launched on.

Procedure

- Create an image and specify the URI of the image to import:

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

- Replace **<container-format>** with one of the following container formats: None, ami, ari, aki, bare, ovf, ova, docker
- Replace **<disk-format>** with one of the following disk formats: None, ami, ari, aki, vhd, vhdx, vmdk, raw, qcow2, vdi, iso, ploop.
- Replace **<name>** with a descriptive name for your image.
- Replace **<uri>** with the URI of your image.

Verification

- Check the availability of the image:

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

- Replace **<image-id>** with the image ID you provided during image creation.

3.2.2. Importing an image from a local volume

The **glance-direct** image import method creates an image record, which generates an image ID. When you upload an image to the Image service (glance) from a local volume, the image is stored in a staging area and becomes active when it passes any configured checks.



NOTE

The **glance-direct** method requires a shared staging area when used in a highly available (HA) configuration. If you upload images by using the **glance-direct** import method, the upload can fail in a HA environment if a shared staging area is not present. In a HA active-active environment, API calls are distributed to the Image service controllers. The download API call can be sent to a different controller than the API call to upload the image.

The **glance-direct** image import method uses three different calls to import an image:

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

You can use the **glance image-create-via-import** command to perform all three of the **glance-direct** calls in one command.

Procedure

1. Source your credentials file.
2. Use the **glance image-create-via-import** command to import a local image:

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

- Replace **<container-format>** with one of the following container formats: None, ami, ari, aki, bare, ovf, ova, docker
- Replace **<disk-format>** with one of the following disk formats: None, ami, ari, aki, vhd, vhdx, vmdk, raw, qcow2, vdi, iso, ploop.
- Replace **<name>** with a descriptive name for your image.
- Replace **</path/to/image>** with the file path to your image file.
When the image moves from the staging area to the back-end storage location, the image is listed. However, it might take some time for the image to become active.

Verification

- Check the availability of the image:

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

- Replace **<image-id>** with the image ID you provided during image creation.

3.3. UPDATING IMAGE PROPERTIES

Update the properties of images that you have stored in the Red Hat OpenStack Platform (RHOSP) Image service (glance).

Procedure

- Use the **glance image-update** command with the **property** option to update an image. For example:

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

- For a list of **glance image-update** command options, see [Image service \(glance\) command options](#).
- For a list of property keys, see [Image configuration parameters](#).

3.4. ENABLING IMAGE CONVERSION

You can upload a QCOW2 image to the Image service (glance) by enabling the **GlanceImageImportPlugins** parameter. You can then convert the QCOW2 image to RAW format.



NOTE

Image conversion is automatically enabled when you use Red Hat Ceph Storage RADOS Block Device (RBD) to store images and boot Nova instances.

To enable image conversion, create an environment file that contains the following parameter value. Include the new environment file with the **-e** option in the **openstack overcloud deploy** command:

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

Use the Image service command-line client for image management.

3.4.1. Converting an image to RAW format

Red Hat Ceph Storage can store, but does not support using, QCOW2 images to host virtual machine (VM) disks.

When you upload a QCOW2 image and create a VM from it, the compute node downloads the image, converts the image to RAW, and uploads it back into Ceph, which can then use it. This process affects the time it takes to create VMs, especially during parallel VM creation.

For example, when you create multiple VMs simultaneously, uploading the converted image to the Ceph cluster might impact already running workloads. The upload process can starve those workloads of IOPS and impede storage responsiveness.

To boot VMs in Ceph more efficiently (ephemeral back end or boot from volume), the glance image format must be RAW.

Procedure

1. Converting an image to RAW might yield an image that is larger in size than the original QCOW2 image file. Run the following command before the conversion to determine the final RAW image size:

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

2. Convert an image from QCOW2 to RAW format:

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

3.4.2. Configuring disk formats with the `GlanceDiskFormats` parameter

You can configure the Image service (glance) to enable or reject disk formats by using the `GlanceDiskFormats` parameter.

Procedure

1. Log in to the undercloud host as the `stack` user.
2. Source the undercloud credentials file:

```
$ source ~/stackrc
```

3. Include the `GlanceDiskFormats` parameter in an environment file, for example, `glance_disk_formats.yaml`:

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

- For example, use the following configuration to enable only RAW and ISO disk formats:

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

- Use the following example configuration to reject QCOW2 disk images:

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

4. Include the environment file that contains your new configuration in the `openstack overcloud deploy` command with any other environment files that are relevant to your environment:

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

```
-e <new_environment_file> \  
...
```

- Replace **<overcloud_environment_files>** with the list of environment files that are part of your deployment.
- Replace **<new_environment_file>** with the environment file that contains your new configuration.

For more information about the disk formats available in RHOSP, see [Image configuration parameters](#).

3.4.3. Storing an image in RAW format

With the **GlanceImageImportPlugins** parameter enabled, run the following command to store a previously created image in RAW format:

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

- Replace **<name>** with the name of the image; this is the name that will appear in **glance image-list**.
- Replace **<http://server/image.qcow2>** with the location and file name of the QCOW2 image.



NOTE

This command example creates the image record and imports it by using the **web-download** method. The glance-api downloads the image from the **--uri** location during the import process. If **web-download** is not available, **glanceclient** cannot automatically download the image data. Run the **glance import-info** command to list the available image import methods.

3.5. HIDING OR UNHIDING IMAGES

You can hide public images from normal listings presented to users. For example, you can hide obsolete CentOS 7 images and show only the latest version to simplify the user experience. Users can discover and use hidden images.

To create a hidden image, add the **--hidden** argument to the **glance image-create** command.

Procedure

- Hide an image:

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

- Unhide an image:

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

- List hidden images:

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

3.6. DELETING IMAGES FROM THE IMAGE SERVICE

Use the **glance image-delete** command to delete one or more images that you do not need to store in the Image service (glance).

Procedure

- Delete one or more images:

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

- Replace **<image-id>** with the ID of the image you want to delete.



WARNING

The **glance image-delete** command permanently deletes the image and all copies of the image, as well as the image instance and metadata.

CHAPTER 4. CONFIGURING THE IMAGE SERVICE IMAGE IMPORT METHOD

The default settings for the Image service (glance) are determined by the Orchestration service (heat) templates that you use when you install Red Hat OpenStack Platform (RHOSP). The Orchestration service template for the Image service is **deployment/glance/glance-api-container-puppet.yaml**.

You can customize aspects of the Image service with a custom environment file, which is a special type of template you can use to customize your Orchestration service templates. For more information about Orchestration service templates and environment files, see [Understanding heat templates](#) in *Installing and managing Red Hat OpenStack Platform with director*.

As a cloud administrator, you can configure an image import workflow for cloud users to upload their own images to the Image service by using the **web-download** or **glance-direct** import methods. You can monitor uploaded images in a staging area before they go active in a storage back end, and you can configure the import workflow to run a set of plugins to make user images discoverable, for example, the image property injection plugin for metadata or the image conversion plugin for image formats.

The **web-download** image import method is enabled by default, but cloud administrators can configure the **glance-direct** method. For further information about the available import methods in Red Hat OpenStack Platform (RHOSP), see [Image service image import methods](#).

4.1. CONFIGURING THE GLANCE-DIRECT IMAGE IMPORT METHOD

When cloud administrators enable the **glance-direct** image import method, cloud users can upload local images to a shared staging area in the OpenStack Image service (glance), a temporary shared storage location common to all Image service API workers.

Procedure

1. Log in to the undercloud host as the **stack** user.
2. Source the **stackrc** undercloud credentials file:

```
$ source ~/stackrc
```

3. Create or open a YAML environment file to configure the import parameters:

Example

```
$ vi /home/stack/templates/<glance-import-settings>.yaml
```

- Replace **<glance-import-settings>** with the name of your file.

4. Configure the NFS back end that is required for shared staging:

```
parameter_defaults:
  GlanceBackend: file
  GlanceNfsEnabled: true
  GlanceNfsShare: 192.168.122.1:/export/glance
```

5. Add **glance-direct** to the **GlanceEnabledImportMethods** parameter to enable the **glance-direct** import method:

```
parameter_defaults:
  [...]
  GlanceEnabledImportMethods: glance-direct,web-download
```

- Configure the NFS staging area that is required for the **glance-direct** import method:

```
parameter_defaults:
  [...]
  GlanceStagingNfsShare: 192.168.122.1:/export/glance-staging
```

The **GlanceEnabledImportMethods** parameter is necessary if you want to enable methods other than **web-download**. For more information about the **GlanceBackend**, **GlanceNfsEnabled**, and **GlanceStagingNfsShare** parameters, see [Image Storage \(glance\) Parameters](#) in *Overcloud parameters*.

- Add your **<glance-import-settings>.yaml** file to the stack with your other environment files and deploy the overcloud:

```
(undercloud)$ openstack overcloud deploy --templates \
-e [your environment files] \
-e /home/stack/templates/<glance-import-settings>.yaml
```

4.2. CONTROLLING IMAGE WEB-IMPORT SOURCES

You can limit the sources of web-import image downloads by adding URI blocklists and allowlists to the optional **glance-image-import.conf** file.

You can allow or block image source URIs at three levels:

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

If you specify both allowlist and blocklist at any level, the allowlist is honored and the blocklist is ignored.

The Image service (glance) applies the following decision logic to validate image source URIs:

- The scheme is checked.
 - Missing scheme: reject
 - If there is an allowlist, and the scheme is not present in the allowlist: reject. Otherwise, skip C and continue on to 2.
 - If there is a blocklist, and the scheme is present in the blocklist: reject.
- The host name is checked.
 - Missing host name: reject
 - If there is an allowlist, and the host name is not present in the allowlist: reject. Otherwise, skip C and continue on to 3.
 - If there is a blocklist, and the host name is present in the blocklist: reject.

3. If there is a port in the URI, the port is checked.
 - a. If there is a allowlist, and the port is not present in the allowlist: reject. Otherwise, skip B and continue on to 4.
 - b. If there is a blacklist, and the port is present in the blacklist: reject.
4. The URI is accepted as valid.

If you allow a scheme, either by adding it to an allowlist or by not adding it to a blacklist, any URI that uses the default port for that scheme by not including a port in the URI is allowed. If it does include a port in the URI, the URI is validated according to the default decision logic.

4.2.1. Image import allowlist example

In this example, the default port for FTP is 21.

Because **ftp** is in the list for **allowed_schemes**, this URL to the image resource is allowed: <ftp://example.org/some/resource>.

However, because 21 is not in the list for **allowed_ports**, this URL to the same image resource is rejected: <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. Default image import blacklist and allowlist settings

The **glance-image-import.conf** file is an optional file that contains the following default options:

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

If you use the defaults, end users can access URIs by using only the **http** or **https** scheme. The only ports that users can specify are **80** and **443**. Users do not have to specify a port, but if they do, it must be either **80** or **443**.

You can find the **glance-image-import.conf** file in the **etc/** subdirectory of the Image service source code tree. Ensure that you are looking in the correct branch for your release of Red Hat OpenStack Platform.

4.3. INJECTING METADATA ON IMAGE IMPORT TO CONTROL WHERE INSTANCES LAUNCH

Cloud users can upload images to the Image service (glance) and use these images to launch instances. Cloud users must launch these images on a specific set of Compute nodes. You can control the assignment of an instance to a Compute node by using image metadata properties.

The image property injection plugin injects metadata properties to images during import. You can specify the properties by editing the **[image_import_opts]** and **[inject_metadata_properties]** sections of the **glance-image-import.conf** file. You can find the **glance-image-import.conf** file in the **etc/** subdirectory of the Image service source code tree. Ensure that you are looking in the correct branch for your release of Red Hat OpenStack Platform (RHOSP).

To enable the image property injection plugin, add the following line to the **[image_import_opts]** section:

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

To limit the metadata injection to images provided by a certain set of users, set the **ignore_user_roles** parameter. For example, use the following configuration to inject one value for **property1** and two values for **property2** into images downloaded by any non-admin user.

```
[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
```

The parameter **ignore_user_roles** is a comma-separated list of the Identity service (keystone) roles that the plugin ignores. This means that if the user that makes the image import call has any of these roles, the plugin does not inject any properties into the image.

The parameter **inject** is a comma-separated list of properties and values that are injected into the image record for the imported image. Each property and value must be quoted and separated by a colon (':').

CHAPTER 5. IMAGE SERVICE WITH MULTIPLE STORES

The Red Hat OpenStack Platform (RHOSP) Image service (glance) supports using multiple stores with distributed edge architecture so that you can have an image pool at every edge site.

5.1. IMAGE COPIES ON MULTIPLE STORES

When you use multiple stores with distributed edge architecture, you can have an image pool at every edge site. You can copy images between the central site, which is also known as the hub site, and the edge sites.

The image metadata contains the location of each copy. For example, an image present on two edge sites is exposed as a single UUID with three locations: the central site plus the two edge sites. This means you can have copies of image data that share a single UUID on many stores. For more information about locations, see [Understanding the location of images](#).

With a RADOS Block Device (RBD) image pool at every edge site, you can boot Virtual Machines (VMs) quickly by using Ceph RBD copy-on-write (COW) and snapshot layering technology. This means that you can boot VMs from volumes and have live migration. For more information about layering with Ceph RBD, see [Ceph block device layering](#) in the *Block Device Guide*.

When you launch an instance at an edge site, the required image is copied to the local Image service (glance) store automatically. However, you can copy images in advance from the central Image store to edge sites by using glance multistore to save time during instance launch.

5.2. REQUIREMENTS OF STORAGE EDGE ARCHITECTURE

Refer to the following requirements to use images with edge sites:

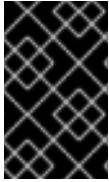
- A copy of each image must exist in the Image service (glance) at the central location.
- You must copy images from an edge site to the central location before you can copy them to other edge sites.
- You must use raw images when deploying a Distributed Compute Node (DCN) architecture with Red Hat Ceph Storage.
- RADOS Block Device (RBD) must be the storage driver for the Image, Compute, and Block Storage services.
- For each site, you must assign the same value to the **NovaComputeAvailabilityZone** and **CinderStorageAvailabilityZone** parameters.

5.3. IMPORTING AN IMAGE TO MULTIPLE STORES

Use the interoperable image import workflow to import image data into multiple Red Hat Ceph Storage clusters. You can import images to the Image service (glance) that are available on the local file system or through a web server.

If you import an image from a web server, the image can be imported into multiple stores at once. If the image is not available on a web server, you can import the image from a local file system into the central store and then copy it to additional stores. For more information, see [Copy an existing image to multiple stores](#).

Use the Image service command-line client for image management.



IMPORTANT

Always store an image copy on the central site, even if there are no instances using the image at the central location. For more information about importing images into the Image service, see the [Deploying a Distributed Compute Node architecture](#) guide.

5.3.1. Managing image import failures

You can manage failures of the image import operation by using the **--allow-failure** parameter:

- If the value of the **--allow-failure** parameter is **true**, the image status becomes **active** after the first store successfully imports the data. This is the default setting. You can view a list of stores that failed to import the image data by using the **os_glance_failed_import** image property.
- If you set the value of the **--allow-failure** parameter to **false**, the image status only becomes **active** after all specified stores successfully import the data. Failure of any store to import the image data results in an image status of **failed**. The image is not imported into any of the specified stores.

5.3.2. Importing image data to multiple stores

Because the default setting of the **--allow-failure** parameter is **true**, you do not need to include the parameter in the command if it is acceptable for some stores to fail to import the image data.



NOTE

This procedure does not require all stores to successfully import the image data.

Procedure

- Import image data to multiple, specified stores:

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

- Replace **<image-name>** with the name of the image you want to import.
- Replace **<uri>** with the URI of the image.
- Replace **<store-1>**, **<store-2>**, and **<store-3>** with the names of the stores to which you want to import the image data.
- Alternatively, replace **--stores** with **--all-stores true** to upload the image to all the stores.



NOTE

The **glance image-create-via-import** command, which automatically converts the QCOW2 image to RAW format, works only with the **web-download** method. The **glance-direct** method is available, but it works only in deployments with a configured shared file system.

5.3.3. Importing image data to multiple stores without failure

This procedure requires all stores to successfully import the image data.

Procedure

1. Import image data to multiple, specified stores:

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

- Replace **<image-name>** with the name of the image you want to import.
- Replace **<uri>** with the URI of the image.
- Replace **<store-1>**, **<store-2>**, and **<store-3>** with the names of stores to which you want to copy the image data.
- Alternatively, replace **--stores** with **--all-stores true** to upload the image to all the stores.



NOTE

With the **--allow-failure** parameter set to **false**, the Image service (glance) does not ignore stores that fail to import the image data. You can view the list of failed stores with the image property **os_glance_failed_import**. For more information, see [Section 5.4, “Checking the progress of the image import operation”](#).

2. Verify that the image data was added to specific stores:

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

Replace **<image-id>** with the ID of the original existing image.

The output displays a comma-delimited list of stores.

5.3.4. Importing image data to a single store

You can use the Image service (glance) to import image data to a single store.

Procedure

1. Import image data to a single store:

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

- Replace **<image-name>** with the name of the image you want to import.
- Replace **<uri>** with the URI of the image.
- Replace **<store>** with the name of the store to which you want to copy the image data.



NOTE

If you do not include the options of **--stores**, **--all-stores**, or **--store** in the command, the Image service creates the image in the central store.

2. Verify that the image data was added to specific store:

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

- Replace **<image-id>** with the ID of the original existing image. The output displays a comma-delimited list of stores.

5.4. CHECKING THE PROGRESS OF THE IMAGE IMPORT OPERATION

The interoperable image import workflow sequentially imports image data into stores. The size of the image, the number of stores, and the network speed between the central site and the edge sites impact how long it takes for the image import operation to complete.

You can follow the progress of the image import by looking at two image properties, which appear in notifications sent during the image import operation:

- The **os_glance_importing_to_stores** property lists the stores that have not imported the image data. At the beginning of the import, all requested stores show up in the list. Each time a store successfully imports the image data, the Image service removes the store from the list.
- The **os_glance_failed_import** property lists the stores that fail to import the image data. This list is empty at the beginning of the image import operation.



NOTE

In the following procedure, the environment has three Red Hat Ceph Storage clusters: the **central** store and two stores at the edge, **dcn0** and **dcn1**.

Procedure

1. Verify that the image data was added to specific stores:

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

- Replace **<image-id>** with the ID of the original existing image. The output displays a comma-delimited list of stores similar to the following example snippet:

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


2. Monitor the status of the image import operation. When you precede a command with **watch**, the command output refreshes every two seconds.

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

- Replace **<image-id>** with the ID of the original existing image.
The status of the operation changes as the image import operation progresses:

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

Output that shows that an image failed to import resembles the following example:

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

After the operation completes, the status changes to active:

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

5.5. COPYING AN EXISTING IMAGE TO MULTIPLE STORES

This feature enables you to copy existing images using Red Hat OpenStack Image service (glance) image data into multiple Red Hat Ceph Storage stores at the edge by using the interoperable image import workflow.



NOTE

The image must be present at the central site before you copy it to any edge sites. Only the image owner or administrator can copy existing images to newly added stores.

You can copy existing image data either by setting **--all-stores** to **true** or by specifying specific stores to receive the image data.

- The default setting for the **--all-stores** option is **false**. If **--all-stores** is **false**, you must specify which stores receive the image data by using **--stores <store-1>,<store-2>**. If the image data is already present in any of the specified stores, the request fails.
- If you set **all-stores** to **true**, and the image data already exists in some of the stores, then those stores are excluded from the list.

After you specify which stores receive the image data, the Image service (glance) copies data from the central site to a staging area. Then the Image service imports the image data by using the interoperable image import workflow. For more information, see [Importing an image to multiple stores](#).

Use the Image service command-line client for image management.



IMPORTANT

Red Hat recommends that administrators carefully avoid closely timed image copy requests. Two closely timed copy-image operations for the same image causes race conditions and unexpected results. Existing image data remains as it is, but copying data to new stores fails.

5.5.1. Copying an image to all stores

Use the following procedure to copy image data to all available stores.

Procedure

1. Copy image data to all available stores:

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

- Replace **<image-id>** with the name of the image you want to copy.

2. Confirm that the image data successfully replicated to all available stores:

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

For information about how to check the status of the image import operation, see [Section 5.4, “Checking the progress of the image import operation”](#).

5.5.2. Copying an image to specific stores

Use the following procedure to copy image data to specific stores.

Procedure

1. Copy image data to specific stores:

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

- Replace **<image-id>** with the name of the image you want to copy.
- Replace **<store-1>** and **<store-2>** with the names of the stores to which you want to copy the image data.

2. Confirm that the image data successfully replicated to the specified stores:

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

For information about how to check the status of the image import operation, see [Section 5.4, “Checking the progress of the image import operation”](#).

5.6. DELETING AN IMAGE FROM A SPECIFIC STORE

Delete an existing image copy on a specific store by using the Red Hat OpenStack Platform (RHOSP) Image service (glance).

Use the Image service command-line client for image management.

Procedure

- Delete an image from a specific store:

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

- Replace **<store-id>** with the name of the store on which the image copy should be deleted.
- Replace **<image-id>** with the ID of the image you want to delete.



WARNING

The **glance image-delete** command permanently deletes the image across all the sites. All image copies are deleted, as well as the image instance and metadata.

5.7. LISTING IMAGE LOCATIONS AND LOCATION PROPERTIES

Although an image can be present on multiple sites, there is only a single Universal Unique Identifier (UUID) for a given image. The image metadata contains the locations of each copy. For example, an image present on two edge sites is exposed as a single UUID with three locations: the central site and the two edge sites.



NOTE

Use the Image service (glance) command-line client instead of the OpenStack command-line client for image management. However, use the **openstack image show** command to list image location properties. The **glance image-show** command output does not include locations.

Procedure

1. Show the sites on which a copy of the image exists:

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

```
| stores | default_backend,dcn1,dcn2
```

In the example, the image is present on the central site, the **default_backend**, and on the two edge sites **dcn1** and **dcn2**.

2. Alternatively, you can run the **glance image-list** command with the **--include-stores** option to see the sites where the images exist:

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

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

- List the image location properties to show the details of each location:

```
$ 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 --)
```

The image properties show the different Ceph RBD URIs for the location of each image.

In the example, the central image location URI is:

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

The URI is composed of the following data:

- **79b70c32-df46-4741-93c0-8118ae2ae284** corresponds to the central Ceph FSID. Each Ceph cluster has a unique FSID.
- The default value for all sites is **images**, which corresponds to the Ceph pool on which the images are stored.
- **2bd882e7-1da0-4078-97fe-f1bb81f61b00** corresponds to the image UUID. The UUID is the same for a given image regardless of its location.
- The metadata shows the glance store to which this location maps. In this example, it maps to the **default_backend**, which is the central hub site.

APPENDIX A. IMAGE SERVICE COMMAND OPTIONS

You can use the following optional arguments with the **glance image-create**, **glance image-create-via-import**, and **glance image-update** commands.

Table A.1. Command options

Specific to	Option	Description
All	--architecture <ARCHITECTURE>	Operating system architecture as specified in https://docs.openstack.org/glance/latest/user/common-image-properties.html#architecture
All	--protected [True_False]	If true, image will not be deletable.
All	--name <NAME>	Descriptive name for the image
All	--instance-uuid <INSTANCE_UUID>	Metadata that can be used to record which instance this image is associated with. (Informational only, does not create an instance snapshot.)
All	--min-disk <MIN_DISK>	Amount of disk space (in GB) required to boot image.
All	--visibility <VISIBILITY>	Scope of image accessibility. Valid values: public, private, community, shared
All	--kernel-id <KERNEL_ID>	ID of image stored in the Image service (glance) that should be used as the kernel when booting an AMI-style image.
All	--os-version <OS_VERSION>	Operating system version as specified by the distributor
All	--disk-format <DISK_FORMAT>	Format of the disk. Valid values: None, ami, ari, aki, vhd, vhdx, vmdk, raw, qcow2, vdi, iso, ploop
All	--os-distro <OS_DISTRO>	Common name of operating system distribution as specified in https://docs.openstack.org/glance/latest/user/common-image-properties.html#os-distro
All	--owner <OWNER>	Owner of the image
All	--ramdisk-id <RAMDISK_ID>	ID of image stored in the Image service that should be used as the ramdisk when booting an AMI-style image.
All	--min-ram <MIN_RAM>	Amount of RAM (in MB) required to boot image.

Specific to	Option	Description
All	--container-format <CONTAINER_FORMAT>	Format of the container. Valid values: None, ami, ari, aki, bare, ovf, ova, docker
All	--property <key=value>	Arbitrary property to associate with image. May be used multiple times.
glance image-create	--tags <TAGS> [<TAGS> ...]	List of strings related to the image
glance image-create	--id <ID>	An identifier for the image
glance image-update	--remove-property	Key name of arbitrary property to remove from the image.

APPENDIX B. IMAGE CONFIGURATION PARAMETERS

You can use the following keys with the **--property** option for the **glance image-create**, **glance image-create-via-import**, and **glance image-update** commands.

Table B.1. Property keys

Specific to	Key	Description	Supported values
All	architecture	The CPU architecture that must be supported by the hypervisor. For example, x86_64 , arm , or ppc64 . Run uname -m to get the architecture of a machine.	<ul style="list-style-type: none"> ● aarch - ARM 64-bit ● alpha - DEC 64-bit RISC ● armv7l - ARM Cortex-A7 MPCore ● cris - Ethernet, Token Ring, AXis-Code Reduced Instruction Set ● i686 - Intel sixth-generation x86 (P6 micro architecture) ● ia64 - Itanium ● lm32 - Lattice Micro32 ● m68k - Motorola 68000 ● microblaze - Xilinx 32-bit FPGA (Big Endian) ● microblazeel - Xilinx 32-bit FPGA (Little Endian) ● mips - MIPS 32-bit RISC (Big Endian) ● mipsel - MIPS 32-bit RISC (Little Endian) ● mips64 - MIPS 64-bit RISC (Big Endian) ● mips64el - MIPS 64-bit RISC (Little Endian) ● openrisc - OpenCores RISC ● parisc - HP Precision Architecture RISC ● parisc64 - HP Precision Architecture 64-bit RISC ● ppc - PowerPC 32-bit ● ppc64 - PowerPC 64-bit ● ppcemb - PowerPC (Embedded 32-bit)

Specific to	Key	Description	Supported values
			<ul style="list-style-type: none"> ● 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 Unicores32 ● x86_64 - 64-bit extension of IA-32 ● xtensa - Tensilica Xtensa configurable microprocessor core ● xtensaeb - Tensilica Xtensa configurable microprocessor core (Big Endian)
All	hypervisor_type	The hypervisor type.	kvm, vmware
All	instance_uuid	For snapshot images, this is the UUID of the server used to create this image.	Valid server UUID
All	kernel_id	The ID of an image stored in the Image Service that should be used as the kernel when booting an AMI-style image.	Valid image ID
All	os_distro	The common name of the operating system distribution in lowercase.	<ul style="list-style-type: none"> ● arch - Arch Linux. Do not use archlinux or org.archlinux. ● centos - Community Enterprise Operating System. Do not use org.centos or CentOS. ● debian - Debian. Do not use Debian or org.debian.

Specific to	Key	Description	Supported values
			<ul style="list-style-type: none"> ● fedora - Fedora. Do not use fedora, org.fedora, or org.fedoraproject. ● freebsd - FreeBSD. Do not use org.freebsd, freeBSD, or FreeBSD. ● gentoo - Gentoo Linux. Do not use Gentoo or org.gentoo. ● mandrake - Mandrakelinux (MandrakeSoft) distribution. Do not use mandrakelinux or MandrakeLinux. ● mandriva - Mandriva Linux. Do not use mandrivalinux. ● mes - Mandriva Enterprise Server. Do not use mandrivaent or mandrivaES. ● msdos - Microsoft Disc Operating System. Do not use ms-dos. ● netbsd - NetBSD. Do not use NetBSD or org.netbsd. ● netware - Novell NetWare. Do not use novell or NetWare. ● openbsd - OpenBSD. Do not use OpenBSD or org.openbsd. ● opensolaris - OpenSolaris. Do not use OpenSolaris or org.opensolaris. ● opensuse - openSUSE. Do not use suse, SuSE, or org.opensuse. ● rhel - Red Hat Enterprise Linux. Do not use redhat, RedHat, or com.redhat. ● sled - SUSE Linux Enterprise Desktop. Do not use com.suse. ● ubuntu - Ubuntu. Do not use Ubuntu, com.ubuntu, org.ubuntu, or canonical.

Specific to	Key	Description	Supported values
			<ul style="list-style-type: none"> windows - Microsoft Windows. Do not use <code>com.microsoft.server</code>.
All	os_version	The operating system version as specified by the distributor.	Version number (for example, "11.10")
All	ramdisk_id	The ID of image stored in the Image Service that should be used as the ramdisk when booting an AMI-style image.	Valid image ID
All	vm_mode	The virtual machine mode. This represents the host/guest ABI (application binary interface) used for the virtual machine.	hvm -Fully virtualized. This is the mode used by QEMU and KVM.
libvirt API driver	hw_cdrom_bu s	Specifies the type of disk controller to attach CD-ROM devices to.	scsi, virtio, ide, or usb . If you specify iscsi , you must set the hw_scsi_model parameter to virtio-scsi .
libvirt API driver	hw_disk_bus	Specifies the type of disk controller to attach disk devices to.	scsi, virtio, ide, or usb . Note that if using iscsi , the hw_scsi_model needs to be set to virtio-scsi .
libvirt API driver	hw_firmware_ type	Specifies the type of firmware to use to boot the instance.	Set to one of the following valid values: <ul style="list-style-type: none"> bios uefi
libvirt API driver	hw_machine_t ype	Enables booting an ARM system using the specified machine type. If an ARM image is used and its machine type is not explicitly specified, then Compute uses the virt machine type as the default for ARMv7 and AArch64.	Valid types can be viewed by using the virsh capabilities command. The machine types are displayed in the machine tag.
libvirt API driver	hw_numa_no des	Number of NUMA nodes to expose to the instance (does not override flavor definition).	Integer.

Specific to	Key	Description	Supported values
libvirt API driver	hw_numa_cpus.0	Mapping of vCPUs N-M to NUMA node 0 (does not override flavor definition).	Comma-separated list of integers.
libvirt API driver	hw_numa_cpus.1	Mapping of vCPUs N-M to NUMA node 1 (does not override flavor definition).	Comma-separated list of integers.
libvirt API driver	hw_numa_memory.0	Mapping N MB of RAM to NUMA node 0 (does not override flavor definition).	Integer
libvirt API driver	hw_numa_memory.1	Mapping N MB of RAM to NUMA node 1 (does not override flavor definition).	Integer
libvirt API driver	hw_pci_numa_affinity_policy	Specifies the NUMA affinity policy for PCI passthrough devices and SR-IOV interfaces.	<p>Set to one of the following valid values:</p> <ul style="list-style-type: none"> ● required: The Compute service creates an instance that requests a PCI device only when at least one of the NUMA nodes of the instance has affinity with the PCI device. This option provides the best performance. ● preferred: The Compute service attempts a best effort selection of PCI devices based on NUMA affinity. If affinity is not possible, then the Compute service schedules the instance on a NUMA node that has no affinity with the PCI device. ● legacy: (Default) The Compute service creates instances that request a PCI device in one of the following cases: <ul style="list-style-type: none"> ○ The PCI device has affinity with at least one of the NUMA nodes. ○ The PCI devices do not provide information about their NUMA affinities.

Specific to	Key	Description	Supported values
libvirt API driver	hw_qemu_guest_agent	Guest agent support. If set to yes , and if qemu-ga is also installed, file systems can be quiesced (frozen) and snapshots created automatically.	yes / no
libvirt API driver	hw_rng_model	<p>Adds a random number generator (RNG) device to instances launched with this image.</p> <p>The instance flavor enables the RNG device by default. To disable the RNG device, the cloud administrator must set hw_rng:allowed to False on the flavor.</p> <p>The default entropy source is /dev/random. To specify a hardware RNG device, set rng_dev_path to /dev/hwrng in your Compute environment file.</p>	virtio , or other supported device.
libvirt API driver	hw_scsi_model	Enables the use of VirtIO SCSI (virtio-scsi) to provide block device access for compute instances; by default, instances use VirtIO Block (virtio-blk). VirtIO SCSI is a para-virtualized SCSI controller device that provides improved scalability and performance, and supports advanced SCSI hardware.	virtio-scsi
libvirt API driver	hw_tpm_model	Set to the model of TPM device to use. Ignored if hw:tpm_version is not configured.	<ul style="list-style-type: none"> ● tpm-tis: (Default) TPM Interface Specification. ● tpm-crb: Command-Response Buffer. Compatible only with TPM version 2.0.

Specific to	Key	Description	Supported values
libvirt API driver	hw_tpm_version	Set to the version of TPM to use. TPM version 2.0 is the only supported version.	2.0
libvirt API driver	hw_video_model	The video device driver for the display device to use in virtual machine instances.	<p>Set to one of the following values to specify the supported driver to use:</p> <ul style="list-style-type: none"> ● virtio - (Default) Recommended Driver for the virtual machine display device, supported by most architectures. The VirtIO GPU driver is included in RHEL-7 and later, and Linux kernel versions 4.4 and later. If an instance kernel has the VirtIO GPU driver, then the instance can use all the VirtIO GPU features. If an instance kernel does not have the VirtIO GPU driver, the VirtIO GPU device gracefully falls back to VGA compatibility mode, which provides a working display for the instance. ● qxl - Deprecated Driver for Spice or noVNC environments that is no longer maintained. ● cirrus - Legacy driver, supported only for backward compatibility. Do not use for new instances. ● vga - Use this driver for IBM Power environments. ● gop - Not supported for QEMU/KVM environments. ● xen - Not supported for KVM environments. ● vmvga - Legacy driver, do not use. ● none - Use this value to disable emulated graphics or video in virtual GPU (vGPU) instances where the driver is configured separately.

Specific to	Key	Description	Supported values
libvirt API driver	hw_video_ram	Maximum RAM for the video image. Used only if a hw_video:ram_max_mb value has been set in the flavor's extra_specs and that value is higher than the value set in hw_video_ram .	Integer in MB (for example, 64)
libvirt API driver	hw_watchdog_action	Enables a virtual hardware watchdog device that carries out the specified action if the server hangs. The watchdog uses the i6300esb device (emulating a PCI Intel 6300ESB). If hw_watchdog_action is not specified, the watchdog is disabled.	<ul style="list-style-type: none"> ● disabled-The device is not attached. Allows the user to disable the watchdog for the image, even if it has been enabled using the image's flavor. The default value for this parameter is disabled. ● reset-Forcefully reset the guest. ● poweroff-Forcefully power off the guest. ● pause-Pause the guest. ● none-Only enable the watchdog; do nothing if the server hangs.
libvirt API driver	os_command_line	The kernel command line to be used by the libvirt driver, instead of the default. For Linux Containers(LXC), the value is used as arguments for initialization. This key is valid only for Amazon kernel, ramdisk, or machine images (aki, ari, or ami).	

Specific to	Key	Description	Supported values
libvirt API driver	os_secure_boot	Use to create an instance that is protected with UEFI Secure Boot.	Set to one of the following valid values: <ul style="list-style-type: none"> ● required: Enables Secure Boot for instances launched with this image. The instance is only launched if the Compute service locates a host that can support Secure Boot. If no host is found, the Compute service returns a "No valid host" error. ● disabled: Disables Secure Boot for instances launched with this image. Disabled by default. ● optional: Enables Secure Boot for instances launched with this image only when the Compute service determines that the host can support Secure Boot.
libvirt API driver and VMware API driver	hw_vif_model	Specifies the model of virtual network interface device to use.	The valid options depend on the configured hypervisor. <ul style="list-style-type: none"> ● KVM and QEMU: e1000, ne2k_pci, pcnet, rtl8139, and virtio. ● VMware: e1000, e1000e, VirtualE1000, VirtualE1000e, VirtualPCNet32, VirtualSriovEthernetCard, and VirtualVmxnet. ● Xen: e1000, netfront, ne2k_pci, pcnet, and rtl8139.
VMware API driver	vmware_adaptype	The virtual SCSI or IDE controller used by the hypervisor.	lsiLogic , busLogic , or ide
VMware API driver	vmware_ostype	A VMware GuestID which describes the operating system installed in the image. This value is passed to the hypervisor when creating a virtual machine. If not specified, the key defaults to otherGuest .	For more information, see Images with VMware vSphere .

Specific to	Key	Description	Supported values
VMware API driver	vmware_image_version	Currently unused.	1
XenAPI driver	auto_disk_config	If true, the root partition on the disk is automatically resized before the instance boots. This value is only taken into account by the Compute service when using a Xen-based hypervisor with the XenAPI driver. The Compute service will only attempt to resize if there is a single partition on the image, and only if the partition is in ext3 or ext4 format.	true / false
libvirt API driver and XenAPI driver	os_type	The operating system installed on the image. The XenAPI driver contains logic that takes different actions depending on the value of the os_type parameter of the image. For example, for os_type=windows images, it creates a FAT32-based swap partition instead of a Linux swap partition, and it limits the injected host name to less than 16 characters.	linux or windows