Abstract

This guide provides procedures for creating and managing images and instances, and procedures for configuring the Compute service for instance creation.
Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE .......................................................... 5
DIRECT DOCUMENTATION FEEDBACK (DDF) FUNCTION NOT AVAILABLE IN THIS BETA RELEASE .... 6

CHAPTER 1. ABOUT INSTANCES ................................................................. 7
CHAPTER 2. INSTANCE BOOT SOURCE ...................................................... 8

CHAPTER 3. TYPES OF INSTANCE STORAGE ............................................. 9
3.1. INSTANCE DISK ............................................................................... 9
3.2. INSTANCE EPHEMERAL STORAGE .................................................. 9
3.3. INSTANCE SWAP STORAGE ......................................................... 9
3.4. INSTANCE BLOCK STORAGE ...................................................... 9
3.5. CONFIG DRIVE ......................................................................... 10

CHAPTER 4. FLAVORS FOR INSTANCES .................................................. 11

CHAPTER 5. CREATING AN INSTANCE ...................................................... 12
5.1. PREREQUISITES ......................................................................... 12
5.2. CREATING AN INSTANCE FROM AN IMAGE .................................. 12
5.3. CREATING AN INSTANCE FROM A BOOTABLE VOLUME ................. 13
5.4. CREATING AN INSTANCE WITH A SR-IOV NETWORK INTERFACE ...... 14
5.5. ADDITIONAL RESOURCES ....................................................... 15

CHAPTER 6. CREATING AN INSTANCE WITH A GUARANTEED MINIMUM BANDWIDTH QOS .......... 16
6.1. REMOVING A GUARANTEED MINIMUM BANDWIDTH QOS FROM AN INSTANCE .... 17

CHAPTER 7. UPDATING AN INSTANCE ..................................................... 18
7.1. ATTACHING A NETWORK TO AN INSTANCE .................................. 18
7.2. DETACHING A NETWORK FROM AN INSTANCE ................................ 18
7.3. ATTACHING A PORT TO AN INSTANCE .......................................... 19
7.4. DETACHING A PORT FROM AN INSTANCE ...................................... 20
7.5. ATTACHING A VOLUME TO AN INSTANCE ...................................... 20
7.6. VIEWING THE VOLUMES ATTACHED TO AN INSTANCE ................. 20
7.7. DETACHING A VOLUME FROM AN INSTANCE ................................... 21
7.8. ATTACHING A VOLUME TO MULTIPLE INSTANCES ......................... 21

CHAPTER 8. PROVIDING PUBLIC ACCESS TO AN INSTANCE ....................... 24
8.1. PREREQUISITES ......................................................................... 24
8.2. SECURING INSTANCE ACCESS WITH SECURITY GROUPS AND KEY PAIRS .......... 24
8.2.1. Creating a security group ......................................................... 24
8.2.2. Updating security group rules ................................................. 25
8.2.3. Deleting security group rules .................................................. 26
8.2.4. Adding a security group to a port .......................................... 27
8.2.5. Removing a security group from a port .................................... 27
8.2.6. Deleting a security group ...................................................... 27
8.2.7. Generating a new SSH key pair ............................................. 28
8.2.8. Importing an existing SSH key pair ....................................... 28
8.2.9. Additional resources ............................................................ 29
8.3. ASSIGNING A FLOATING IP ADDRESS TO AN INSTANCE ............... 29
8.4. DISASSOCIATING A FLOATING IP ADDRESS FROM AN INSTANCE .......... 30
8.5. CREATING AN INSTANCE WITH SSH ACCESS ................................ 31
8.6. ADDITIONAL RESOURCES ....................................................... 32
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.2.2. Importing image data to multiple stores</td>
<td>72</td>
</tr>
<tr>
<td>13.2.3. Importing image data to multiple stores without failure</td>
<td>72</td>
</tr>
<tr>
<td>13.2.4. Importing image data to a single store</td>
<td>73</td>
</tr>
<tr>
<td>13.2.5. Checking the progress of the image import operation</td>
<td>74</td>
</tr>
<tr>
<td>13.3. COPY AN EXISTING IMAGE TO MULTIPLE STORES</td>
<td>75</td>
</tr>
<tr>
<td>13.3.1. Copying an image to all stores</td>
<td>75</td>
</tr>
<tr>
<td>13.3.2. Copying an image to specific stores</td>
<td>76</td>
</tr>
<tr>
<td>13.4. DELETING AN IMAGE FROM A SPECIFIC STORE</td>
<td>76</td>
</tr>
<tr>
<td>13.5. UNDERSTANDING THE LOCATIONS OF IMAGES</td>
<td>77</td>
</tr>
<tr>
<td><strong>APPENDIX A. IMAGE CONFIGURATION PARAMETERS</strong></td>
<td>79</td>
</tr>
</tbody>
</table>
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.
DIRECT DOCUMENTATION FEEDBACK (DDF) FUNCTION NOT AVAILABLE IN THIS BETA RELEASE

The Direct Documentation Feedback (DDF) function allows users to enter feedback directly on documentation pages on fully supported Red Hat documentation products. The DDF function is not available in this Red Hat OpenStack platform 16.2 beta documentation set.
CHAPTER 1. ABOUT INSTANCES

Instances are the individual virtual machines that run on physical Compute nodes inside the cloud. To launch an instance, you need a flavor and either an image or a bootable volume. When you use an image to launch an instance, the provided image becomes the base image that contains a virtual disk installed with a bootable operating system. Each instance requires a root disk, which we refer to as the instance disk. The Compute service (nova) resizes the instance disk to match the specifications of the flavor that you specified for the instance.

Images are managed by the Image Service (glance). The Image Service image store contains a number of predefined images. The Compute nodes provide the available vCPU, memory, and local disk resources for instances. The Block Storage service (cinder) provides predefined volumes. Instance disk data is stored either in ephemeral storage, which is deleted when you delete the instance, or in a persistent volume provided by the Block Storage service.

The Compute service is the central component that provides instances on demand. The Compute service creates, schedules, and manages instances, and interacts with the Identity service for authentication, the Image service for the images used to launch instances, and the Dashboard service (horizon) for the user and administrative interface. As a cloud user, you interact with the Compute service when you create and manage your instances. You can create and manage your instances by using the OpenStack CLI or the Dashboard.
CHAPTER 2. INSTANCE BOOT SOURCE

The boot source for an instance can be an image or a bootable volume. The instance disk of an instance that you boot from an image is controlled by the Compute service and deleted when the instance is deleted. The instance disk of an instance that you boot from a volume is controlled by the Block Storage service and is stored remotely.

An image contains a bootable operating system. The Image Service (glance) controls image storage and management. You can launch any number of instances from the same base image. Each instance runs from a copy of the base image. Any changes that you make to the instance do not affect the base image.

A bootable volume is a block storage volume created from an image that contains a bootable operating system. The instance can use the bootable volume to persist instance data when the instance is deleted. You can use an existing persistent root volume when you launch an instance. You can also create persistent storage when you launch an instance from an image, so that you can save the instance data when the instance is deleted. A new persistent storage volume is created automatically when you create an instance from a volume snapshot.

The following diagram shows the instance disks and storage that you can create when you launch an instance. The actual instance disks and storage created depend on the boot source and flavor used.
CHAPTER 3. TYPES OF INSTANCE STORAGE

The virtual storage that is available to an instance is defined by the flavor used to launch the instance. The following virtual storage resources can be associated with an instance:

- Instance disk
- Ephemeral storage
- Swap storage
- Persistent block storage volumes
- Config drive

3.1. INSTANCE DISK

The instance disk created to store instance data depends on the boot source that you use to create the instance. The instance disk of an instance that you boot from an image is controlled by the Compute service and deleted when the instance is deleted. The instance disk of an instance that you boot from a volume is a persistent volume provided by the Block Storage service.

3.2. INSTANCE EPHEMERAL STORAGE

You can specify that an ephemeral disk is created for the instance by choosing a flavor that configures an ephemeral disk. This ephemeral storage is an empty additional disk that is available to an instance. This storage value is defined by the instance flavor. The default value is 0, meaning that no secondary ephemeral storage is created.

The ephemeral disk appears in the same way as a plugged-in hard drive or thumb drive. It is available as a block device, which you can check using the lsblk command. You can mount it and use it however you normally use a block device. You cannot preserve or reference that disk beyond the instance it is attached to.

**NOTE**

Ephemeral storage data is not included in instance snapshots, and is not available on instances that are shelved and then unshelved.

3.3. INSTANCE SWAP STORAGE

You can specify that a swap disk is created for the instance by choosing a flavor that configures a swap disk. This swap storage is an additional disk that is available to the instance for use as swap space for the running operating system.

3.4. INSTANCE BLOCK STORAGE

A block storage volume is persistent storage that is available to an instance regardless of the state of the running instance. You can attach multiple block devices to an instance, one of which can be a bootable volume.
NOTE
When you use a block storage volume for your instance disk data, the block storage volume persists for any instance rebuilds, even when an instance is rebuilt with a new image that requests that a new volume is created.

3.5. CONFIG DRIVE

You can attach a config drive to an instance when it boots. The config drive is presented to the instance as a read-only drive. The instance can mount this drive and read files from it. You can use the config drive as a source for cloud-init information. Config drives are useful when combined with cloud-init for server bootstrapping, and when you want to pass large files to your instances. For example, you can configure cloud-init to automatically mount the config drive and run the setup scripts during the initial instance boot. Config drives are created with the volume label of config-2, and attached to the instance when it boots. The contents of any additional files passed to the config drive are added to the user_data file in the openstack/{version}/ directory of the config drive. cloud-init retrieves the user data from this file.
CHAPTER 4. FLAVORS FOR INSTANCES

An instance flavor is a resource template that specifies the virtual hardware profile for the instance. You select a flavor when you launch instances to specify the virtual resources to allocate to the instance. Flavors define the number of virtual CPUs, the amount of RAM, the size of the root disk, and the size of the virtual storage, including secondary ephemeral storage and swap disk, to create the instance with. You select the flavor from the set of available flavors defined for your project within the cloud.
CHAPTER 5. CREATING AN INSTANCE

Before you can create an instance, other Red Hat OpenStack Platform (RHOSP) components must be available, such as the flavor, boot source, network, key pair, and security group. These components are used in the creation of an instance and are not available by default.

When you create an instance, you choose a boot source that has the bootable operating system that you require for your instance, a flavor that has the hardware profile you require for your instance, the network you want to connect your instance to, and any additional storage you need, such as data volumes and ephemeral storage.

NOTE

When you create an instance, the Compute (nova) service uses the name you give the instance to generate a valid hostname for the instance in the metadata service and config drive. The Compute service removes all periods (.) and any non-alphanumeric characters, except dashes (-), from the instance name to generate a valid DNS label. This results in a hostname that consists only of alphanumeric characters and dashes (-). Therefore, it is not possible to specify the instance name as a FQDN as the Compute service replaces the periods (.) when generating the hostname.

5.1. PREREQUISITES

- The required image or volume is available as the boot source:
  - For more information about how to create an image, see Creating an image.
  - For more information about how to create a volume, see Creating Block Storage volumes in the Storage Guide.
  - For more information about the options available for the boot source of an instance, see Instance boot source.

- A flavor is available that specifies the required number of CPUs, memory, and storage capacity. The flavor settings must meet the minimum requirements for disk and memory size specified by your chosen image, otherwise the instance will fail to launch.

- The required network is available. For more information about how to create a network, see Creating a network in the Networking Guide.

5.2. CREATING AN INSTANCE FROM AN IMAGE

You can create an instance by using an image as the boot source.

Procedure

1. Retrieve the name or ID of the flavor that has the hardware profile that your instance requires:

   ```shell
   $ openstack flavor list
   ```

   NOTE

   Choose a flavor with sufficient size for the image to successfully boot, otherwise the instance will fail to launch.
2. Retrieve the name or ID of the image that has the software profile that your instance requires:

```
$ openstack image list
```

If the image that you require is not available, you can download or create a new image. For information about how to create or download cloud images, see Image service.

3. Retrieve the name or ID of the network that you want to connect your instance to:

```
$ openstack network list
```

4. Create your instance:

```
$ openstack server create --flavor <flavor> --image <image> --network <network> --wait myInstanceFromImage
```

- Replace `<flavor>` with the name or ID of the flavor that you retrieved in step 1.
- Replace `<image>` with the name or ID of the image that you retrieved in step 2.
- Replace `<network>` with the name or ID of the network that you retrieved in step 3. You can use the `--network` option more than once to connect your instance to several networks, as required.

### 5.3. CREATING AN INSTANCE FROM A BOOTABLE VOLUME

You can create an instance by using a bootable volume as the boot source. Boot your instance from a volume when you need to improve the availability of the instance data in the event of a failure.

**NOTE**

When you use a block storage volume for your instance disk data, the block storage volume persists for any instance rebuilds, even when an instance is rebuilt with a new image that requests that a new volume is created.

**Procedure**

1. Retrieve the name or ID of the image that has the software profile that your instance requires:

```
$ openstack image list
```

If the image that you require is not available, you can download or create a new image. For information about how to create or download cloud images, see Image service.

2. Create a bootable volume from the image:

```
$ openstack volume create --image <image> --size <size_gb> --bootable myBootableVolume
```

- Replace `<image>` with the name or ID of the image to write to the volume, retrieved in step 1.
- Replace `<size_gb>` with the size of the volume in GB.
3. Retrieve the name or ID of the flavor that has the hardware profile that your instance requires:

   $ openstack flavor list

4. Retrieve the name or ID of the network that you want to connect your instance to:

   $ openstack network list

5. Create an instance with the bootable volume:

   $ openstack server create --flavor <flavor> \
   --volume myBootableVolume --network <network> \
   --wait myInstanceFromVolume

   - Replace `<flavor>` with the name or ID of the flavor that you retrieved in step 3.
   - Replace `<network>` with the name or ID of the network that you retrieved in step 4. You can use the `--network` option more than once to connect your instance to several networks, as required.

### 5.4. CREATING AN INSTANCE WITH A SR-IOV NETWORK INTERFACE

To create an instance with a single root I/O virtualization (SR-IOV) network interface you need to create the required SR-IOV port.

**Procedure**

1. Retrieve the name or ID of the flavor that has the hardware profile that your instance requires:

   $ openstack flavor list

   **NOTE**

   Choose a flavor with sufficient size for the image to successfully boot, otherwise the instance will fail to launch.

   **TIP**

   You can specify the NUMA affinity policy that is applied to your instance for PCI passthrough devices and SR-IOV interfaces, by selecting a flavor that has the policy you require. For more information on the available policies, see *Instance PCI NUMA affinity policy* in *Flavor metadata*. If you choose a flavor with a NUMA affinity policy, then the image that you use must have either the same NUMA affinity policy or no NUMA affinity policy.

2. Retrieve the name or ID of the image that has the software profile that your instance requires:

   $ openstack image list

   If the image that you require is not available, you can download or create a new image. For information about how to create or download cloud images, see *Image service*.
TIP

You can specify the NUMA affinity policy that is applied to your instance for PCI passthrough devices and SR-IOV interfaces, by selecting an image that has the policy you require. For more information on the available policies, see Instance PCI NUMA affinity policy in Flavor metadata. If you choose an image with a NUMA affinity policy, then the flavor that you use must have either the same NUMA affinity policy or no NUMA affinity policy.

3. Retrieve the name or ID of the network that you want to connect your instance to:

   $ openstack network list

4. Create the type of port that you require for your SR-IOV interface:

   $ openstack port create --network <network> \
   --vnic-type <vnic_type> mySriovPort

   - Replace <network> with the name or ID of the network you retrieved in step 3.
   - Replace <vnic_type> with the one of the following values:
     - direct: Creates a direct mode SR-IOV virtual function (VF) port.
     - direct-physical: Creates a direct mode SR-IOV physical function (PF) port.
     - macvtap: Creates an indirect mode SR-IOV VF port that uses MacVTap to expose the virtio interface to the instance.

5. Create your instance:

   $ openstack server create --flavor <flavor> \
   --image <image> --port <port> \
   --wait mySriovInstance

   - Replace <flavor> with the name or ID of the flavor that you retrieved in step 1.
   - Replace <image> with the name or ID of the image that you retrieved in step 2.
   - Replace <port> with the name or ID of the port that you created in step 4.

5.5. ADDITIONAL RESOURCES

- Customizing an instance by using user data
CHAPTER 6. CREATING AN INSTANCE WITH A GUARANTEED MINIMUM BANDWIDTH QOS

You can create instances that request a guaranteed minimum bandwidth by using a Quality of Service (QoS) policy.

QoS policies with a guaranteed minimum bandwidth rule are assigned to ports on a specific physical network. When you create an instance that uses the configured port, the Compute scheduling service selects a host for the instance that satisfies this request. The Compute scheduling service checks the Placement service for the amount of bandwidth reserved by other instances on each physical interface, before selecting a host to deploy an instance on.

Limitations/Restrictions

- You can only assign a guaranteed minimum bandwidth QoS policy when creating a new instance. You cannot assign a guaranteed minimum bandwidth QoS policy to instances that are already running, as the Compute service only updates resource usage for an instance in placement during creation or move operations, which means the minimum bandwidth available to the instance cannot be guaranteed.

- You cannot live migrate, unshelf or evacuate an instance that uses a port that has resource requests, such as a guaranteed minimum bandwidth QoS policy. Run the following command to check if a port has resource requests:

  $ openstack port show <port_name/port_id>

Prerequisites

- A QoS policy is available that has a minimum bandwidth rule. For more information, see Configuring Quality of Service (QoS) policies in the Networking Guide.

Procedure

1. List the available QoS policies:

   (overcloud)$ openstack network qos policy list

   ┌───────────────┬──────────┬──────┬───────┬──────────────┬───────────┐
   │ ID            │ Name     │ Shared| Default| Project       │          │
   │ 6d771447-3cf4-4ef1-b613-945e990fa59f | policy2  | True  | False  |              |          │
   | ba4de51bf7694228a350dd22b7a3dc24 |          |       |        |              |          │
   | 78a24462-e3c1-4e66-a042-71131a7daed5 | policy1  | True  | False  |              |          │
   | ba4de51bf7694228a350dd22b7a3dc24 |          |       |        |              |          │
   | b80acc64-4fc2-41f2-a346-520d7cfe02b | policy0  | True  | False  |              |          │
   | ba4de51bf7694228a350dd22b7a3dc24 |          |       |        |              |          │
   └───────────────┴──────────┴──────┴───────┴──────────────┴───────────┘

2. Check the rules of each of the available policies to determine which has the required minimum bandwidth:

   (overcloud)$ openstack network qos policy show policy0
3. Create a port from the appropriate policy:

   (overcloud)$ openstack port create port-normal-qos --network net0 --qos-policy policy0

4. Create an instance, specifying the NIC port to use:

   $ openstack server create --flavor cirros256 --image cirros-0.3.5-x86_64-disk --nic port-id=port-normal-qos --wait qos_instance

   An "ACTIVE" status in the output indicates that you have successfully created the instance on a host that can provide the requested guaranteed minimum bandwidth.

6.1. REMOVING A GUARANTEED MINIMUM BANDWIDTH QOS FROM AN INSTANCE

If you want to lift the guaranteed minimum bandwidth QoS policy restriction from an instance, you can detach the interface.

Procedure

- To detach the interface, enter the following command:

   $ openstack server remove port <vm_name|vm_id> <port_name|port_id>
CHAPTER 7. UPDATING AN INSTANCE

You can add and remove additional resources from running instances, such as persistent volume storage, a network interface, or a public IP address. You can also update instance metadata and the security groups that the instance belongs to.

7.1. ATTACHING A NETWORK TO AN INSTANCE

You can attach a network to a running instance. When you attach a network to the instance, the Compute service creates the port on the network for the instance. Use a network to attach the network interface to an instance when you want to use the default security group and there is only one subnet on the network.

Procedure

1. Identify the available networks and note the name or ID of the network that you want to attach to your instance:

   (overcloud)$ openstack network list

   If the network that you need is not available, create a new network:

   (overcloud)$ openstack network create <network>

2. Attach the network to your instance:

   $ openstack server add network <instance> <network>

   Replace <instance> with the name or ID of the instance that you want to attach the network to.

   Replace <network> with the name or ID of the network that you want to attach to the instance.

Additional resources

- openstack network create command in the Command Line Interface Reference.
- Creating a network in the Networking Guide.

7.2. DETACHING A NETWORK FROM AN INSTANCE

You can detach a network from an instance.

NOTE

Detaching the network detaches all network ports. If the instance has multiple ports on a network and you want to detach only one of those ports, follow the Detaching a port from an instance procedure to detach the port.

Procedure

1. Identify the network that is attached to the instance:
2. Detach the network from the instance:

$ openstack server remove network <instance> <network>

- Replace `<instance>` with the name or ID of the instance that you want to remove the network from.
- Replace `<network>` with the name or ID of the network that you want to remove from the instance.

### 7.3. ATTACHING A PORT TO AN INSTANCE

You can attach a network interface to a running instance by using a port. You can attach a port to only one instance at a time. Use a port to attach the network interface to an instance when you want to use a custom security group, or when there are multiple subnets on the network.

**TIP**

If you attach the network interface by using a network, the port is created automatically. For more information, see *Attaching a network to an instance*.

**NOTE**

You cannot attach a port with an SR-IOV vNIC to an instance, or a port with a guaranteed minimum bandwidth QoS policy.

**Procedure**

1. Identify the available ports and note the name or ID of the port that you want to attach to your instance:

   (overcloud)$ openstack port list

   If the port that you need is not available, create a new port:

   (overcloud)$ openstack port create --network <network> <port>

   - Replace `<network>` with the name or ID of the network to create the port on.
   - Replace `<port>` with the name or ID of the port that you want to attach to the instance.

2. Attach the port to your instance:

   $ openstack server add port <instance> <port>

   - Replace `<instance>` with the name or ID of the instance that you want to attach the port to.
   - Replace `<port>` with the name or ID of the port that you want to attach to the instance.

**Additional resources**
7.4. DETACHING A PORT FROM AN INSTANCE

You can detach a port from an instance.

Procedure

1. Identify the port that is attached to the instance:

   (overcloud)$ openstack server show <instance>

2. Detach the port from the instance:

   $ openstack server remove port <instance> <port>

   - Replace `<instance>` with the name or ID of the instance that you want to remove the port from.
   - Replace `<port>` with the name or ID of the port that you want to remove from the instance.

7.5. ATTACHING A VOLUME TO AN INSTANCE

You can attach a volume to an instance for persistent storage. You can attach a volume to only one instance at a time, unless the volume has been configured as a multiattach volume. For more information about creating multiattach-capable volumes, see Attach a volume to multiple instances.

Procedure

1. Identify the available volumes and note the name or ID of the volume that you want to attach to your instance:

   (overcloud)$ openstack volume list

2. Attach the volume to your instance:

   $ openstack server add volume <instance> <volume>

   - Replace `<instance>` with the name or ID of the instance that you want to attach the volume to.
   - Replace `<volume>` with the name or ID of the volume that you want to attach to the instance.

   NOTE

   Specify `--os-compute-api-version 2.20` or higher to add a volume to an instance with status SHELVED or SHELVED_OFFLOADED.

7.6. VIEWING THE VOLUMES ATTACHED TO AN INSTANCE
You can view the volumes attached to a particular instance.

Prerequisites

- You are using `python-openstackclient 5.5.0`.

Procedure

- List the volumes attached to an instance:

  ```
  $ openstack server volume list <instance>
  +---------------------+----------+---------------------+-----------------------+
  | ID                  | Device   | Server ID           | Volume ID             |
  +---------------------+----------+---------------------+-----------------------+
  | 1f9dcb02-9a20-4a4b-9f25-c7846a1ce9e8 | /dev/vda | ab96b635-1e63-4487-c7846a1ce9e8 | 25-c7846a1ce9e8       |
  +---------------------+----------+---------------------+-----------------------+
  ```

7.7. DETACHING A VOLUME FROM AN INSTANCE

You can detach a volume from an instance.

**NOTE**

Detaching the network detaches all network ports. If the instance has multiple ports on a network and you want to detach only one of those ports, follow the [Detaching a port from an instance](#detaching-a-port-from-an-instance) procedure to detach the port.

Procedure

1. Identify the volume that is attached to the instance:

   ```
   (overcloud)$ openstack server show <instance>
   ```

2. Detach the volume from the instance:

   ```
   $ openstack server remove volume <instance> <volume>
   ```

   - Replace `<instance>` with the name or ID of the instance that you want to remove the volume from.
   - Replace `<volume>` with the name or ID of the volume that you want to remove from the instance.

**NOTE**

Specify `--os-compute-api-version 2.20` or higher to remove a volume from an instance with status `SHELVED` or `SHELVED_OFFLOADED`.

7.8. ATTACHING A VOLUME TO MULTIPLE INSTANCES

You can attach multiple instances to the same Block Storage (cinder) volume when the volume is
configured as multi-attach. Multi-attach gives multiple instances simultaneous read/write access to a single Block Storage volume. This is useful for deploying clustered database systems, such as MSSQL and Oracle. Multi-attach is supported by the Compute service (nova) libvirt driver, and the following Block Storage drivers:

- Scaleio
- Infinidat
- Solidfire
- Zfssaiscsi

Prerequisites

- You must use Compute API version 2.60 or later. If you are using an earlier version of the API you will see the following error:

  Multiattach volumes are only supported starting with compute API version 2.60. (HTTP 400)
  (Request-ID: req-3a969c31-e360-4c79-a403-75cc6053c9e5)

  You can either set the environment variable `OS_COMPUTE_API_VERSION=2.72`, or run the command again using the `--os-compute-api-version` argument:

    $ openstack --os-compute-api-version 2.72 server add volume <instance_name> <volume_name>

- Multi-attach volumes are supported in your project.

Procedure

1. Identify the available volumes and note the name or ID of the volume that you want to attach to your instances:

   (overcloud)$ openstack volume list

2. Ensure that your chosen volume is multi-attach capable:

   (overcloud)$ openstack volume show volMultiattach | grep multiattach
   multiattach | True |
   type        | multiattach |

3. Attach the volume to an instance:

   $ openstack server add volume <instance> <volume>
   - Replace `<instance>` with the name or ID of the instance that you want to attach the volume to.
   - Replace `<volume>` with the name or ID of the volume that you want to attach to the instance.

4. Repeat steps 2 and 3 until all instances that you want to attach to the volume are attached.
5. Verify the status of the volume and confirm that the instances are attached:

```bash
$ openstack volume list
```

```
+-----------------------------------------------------+---------------------+---------+-----+----------------+
| ID                                                             | Name                | Status | Size| Attached to     |
+-----------------------------------------------------+---------------------+---------+-----+----------------+
| f3fb92f6-c77b-429f-871d-65b1e3afa750 | volMultiattach       | in-use | 50  | Attached to instance1 on /dev/vdb Attached to instance2 on /dev/vdb |
```

---

CHAPTER 7. UPDATING AN INSTANCE

---

23
CHAPTER 8. PROVIDING PUBLIC ACCESS TO AN INSTANCE

New instances automatically receive a port with a fixed IP address on the network that the instance is assigned to. This IP address is private and is permanently associated with the instance until the instance is deleted. The fixed IP address is used for communication between instances.

You can connect a public instance directly to a shared external network where a public IP address is directly assigned to the instance. This is useful if you are working in a private cloud.

You can also provide public access to an instance through a project network that has a routed connection to an external provider network. This is the preferred method if you are working in a public cloud, or when public IP addresses are limited. To provide public access through the project network, the project network must be connected to a router with the gateway set to the external network. For external traffic to reach the instance, the cloud user must associate a floating IP address with the instance.

To provide access to and from an instance, whether it is connected to a shared external network or a routed provider network, you must configure security group rules for the required protocols, such as SSH, ICMP, or HTTP. You must also pass a key pair to the instance during creation, so that you can access the instance remotely.

8.1. PREREQUISITES

- The external network must have a subnet to provide the floating IP addresses.
- The project network must be connected to a router that has the external network configured as the gateway.

8.2. SECURING INSTANCE ACCESS WITH SECURITY GROUPS AND KEY PAIRS

Security groups are sets of IP filter rules that control network and protocol access to and from instances, such as ICMP to allow you to ping an instance, and SSH to allow you to connect to an instance. The security group rules are applied to all instances within a project.

All projects have a default security group called default, which is used when you do not specify a security group for your instances. By default, the default security group allows all outgoing traffic and denies all incoming traffic from any source other than instances in the same security group. You can either add rules to the default security group or create a new security group for your project. You can apply one or more security groups to an instance during instance creation. To apply a security group to a running instance, apply the security group to a port attached to the instance.

Key pairs are SSH or x509 credentials that are injected into an instance when it is launched to enable remote access to the instance. You can create new key pairs in RHOSP, or import existing key pairs. Each user should have at least one key pair. The key pair can be used for multiple instances.

**NOTE**

You cannot share key pairs between users in a project because each key pair belongs to the individual user that created or imported the key pair, rather than to the project.

8.2.1. Creating a security group

You can create a new security group to apply to instances and ports within a project.
Procedure

1. Optional: To ensure the security group you need does not already exist, review the available security groups and their rules:

   $ openstack security group list
   $ openstack security group rule list <sec_group>

   - Replace `<sec_group>` with the name or ID of the security group that you retrieved from the list of available security groups.

2. Create your security group:

   $ openstack security group create mySecGroup

3. Add rules to your security group:

   $ openstack security group rule create --protocol <protocol> \
   [---dst-port <port-range>] \
   [---remote-ip <ip-address> | --remote-group <group>] \
   [---ingress | --egress] mySecGroup

   - Replace `<protocol>` with the name of the protocol you want to allow to communicate with your instances.

   - Optional: Replace `<port-range>` with the destination port or port range to open for the protocol. Required for IP protocols TCP, UDP, and SCTP. Set to `-1` to allow all ports for the specified protocol.

   - Optional: You can allow access only from specified IP addresses by using --remote-ip to specify the remote IP address block, or --remote-group to specify that the rule only applies to packets from interfaces that are a member of the remote group. If using --remote-ip, replace `<ip-address>` with the remote IP address block. You can use CIDR notation. If using --remote-group, replace `<group>` with the name or ID of the existing security group. If neither option is specified, then access is allowed to all addresses, as the remote IP access range defaults (IPv4 default: `0.0.0.0/0`; IPv6 default: `::/0`).

   - Specify the direction of network traffic the protocol rule applies to, either incoming (ingress) or outgoing (egress). If not specified, defaults to ingress.

4. Repeat step 3 until you have created rules for all the protocols that you want to allow to access your instances. The following example creates a rule to allow SSH connections to instances in the security group `mySecGroup`:

   $ openstack security group rule create --protocol tcp \
   --dst-port 22 mySecGroup

8.2.2. Updating security group rules

You can update the rules of any security group that you have access to.

Procedure

1. Retrieve the name or ID of the security group that you want to update the rules for:
2. Determine the rules that you need to apply to the security group.

3. Add rules to your security group:

```
$ openstack security group rule create --protocol <protocol> \
    [--dst-port <port-range>] \
    [--remote-ip <ip-address> | --remote-group <group>] \
    [--ingress | --egress] <group_name>
```

- Replace `<protocol>` with the name of the protocol you want to allow to communicate with your instances.

- Optional: Replace `<port-range>` with the destination port or port range to open for the protocol. Required for IP protocols TCP, UDP, and SCTP. Set to -1 to allow all ports for the specified protocol.

- Optional: You can allow access only from specified IP addresses by using `--remote-ip` to specify the remote IP address block, or `--remote-group` to specify that the rule only applies to packets from interfaces that are a member of the remote group. If using `--remote-ip`, replace `<ip-address>` with the remote IP address block. You can use CIDR notation. If using `--remote-group`, replace `<group>` with the name or ID of the existing security group. If neither option is specified, then access is allowed to all addresses, as the remote IP access range defaults (IPv4 default: 0.0.0.0/0; IPv6 default: ::/0).

- Specify the direction of network traffic the protocol rule applies to, either incoming (ingress) or outgoing (egress). If not specified, defaults to ingress.

- Replace `<group_name>` with the name or ID of the security group that you want to apply the rule to.

4. Repeat step 3 until you have created rules for all the protocols that you want to allow to access your instances. The following example creates a rule to allow SSH connections to instances in the security group `mySecGroup`:

```
$ openstack security group rule create --protocol tcp \
    --dst-port 22 mySecGroup
```

### 8.2.3. Deleting security group rules

You can delete rules from a security group.

**Procedure**

1. Identify the security group that the rules are applied to:

```
$ openstack security group list
```

2. Retrieve IDs of the rules associated with the security group:

```
$ openstack security group show <sec-group>
```

3. Delete the rule or rules:
$ openstack security group rule delete <rule> [<rule> ...]

Replace `<rule>` with the ID of the rule to delete. You can delete more than one rule at a time by specifying a space-delimited list of the IDs of the rules to delete.

### 8.2.4. Adding a security group to a port

The **default** security group is applied to instances that do not specify an alternative security group. You can apply an alternative security group to a port on a running instance.

**Procedure**

1. Determine the port on the instance that you want to apply the security group to:

   $ openstack port list --server myInstanceWithSSH

2. Apply the security group to the port:

   $ openstack port set --security-group <sec_group> <port>

   Replace `<sec_group>` with the name or ID of the security group you want to apply to the port on your running instance. You can use the `--security-group` option more than once to apply multiple security groups, as required.

### 8.2.5. Removing a security group from a port

To remove a security group from a port you need to first remove all the security groups, then re-add the security groups that you want to remain assigned to the port.

**Procedure**

1. List all the security groups associated with the port and record the IDs of the security groups that you want to remain associated with the port:

   $ openstack port show <port>

2. Remove all the security groups associated with the port:

   $ openstack port set --no-security-group <port>

3. Re-apply the security groups to the port:

   $ openstack port set --security-group <sec_group> <port>

   Replace `<sec_group>` with the ID of the security group that you want to re-apply to the port on your running instance. You can use the `--security-group` option more than once to apply multiple security groups, as required.

### 8.2.6. Deleting a security group

You can delete security groups that are not associated with any ports.

**Procedure**
Procedure

1. Retrieve the name or ID of the security group that you want to delete:

   $ openstack security group list

2. Retrieve a list of the available ports:

   $ openstack port list

3. Check each port for an associated security group:

   $ openstack port show <port-uuid> -c security_group_ids

   If the security group you want to delete is associated with any of the ports, then you must first remove the security group from the port. For more information, see Removing a security group from a port.

4. Delete the security group:

   $ openstack security group delete <group> [...]

   Replace `<group>` with the ID of the group that you want to delete. You can delete more than one group at a time by specifying a space-delimited list of the IDs of the groups to delete.

8.2.7. Generating a new SSH key pair

You can create a new SSH key pair for use within your project.

NOTE

Use a x509 certificate to create a key pair for a Windows instance.

Procedure

1. Create the key pair and save the private key in your local `.ssh` directory:

   $ openstack keypair create <keypair> > ~/.ssh/<keypair>.pem

   Replace `<keypair>` with the name of your new key pair.

2. Protect the private key:

   $ chmod 600 ~/.ssh/<keypair>.pem

8.2.8. Importing an existing SSH key pair

You can import an SSH key to your project that you created outside of the Red Hat OpenStack Platform (RHOSP) by providing the public key file when you create a new key pair.

Procedure
1. Create the key pair from the existing key file and save the private key in your local .ssh directory:

- To import the key pair from an existing public key file, enter the following command:

  `openstack keypair create --public-key ~/.ssh/<public_key>.pub <keypair> > ~/.ssh/<keypair>.pem`

  - Replace `<public_key>` with the name of the public key file that you want to use to create the key pair.
  - Replace `<keypair>` with the name of your new key pair.

- To import the key pair from an existing private key file, enter the following command:

  `openstack keypair create --private-key ~/.ssh/<private_key> <keypair> > ~/.ssh/<keypair>.pem`

  - Replace `<private_key>` with the name of the public key file that you want to use to create the key pair.
  - Replace `<keypair>` with the name of your new key pair.

2. Protect the private key:

   `chmod 600 ~/.ssh/<keypair>.pem`

8.2.9. Additional resources

- Security groups in the Networking Guide.

- Project security management in the Users and Identity Management Guide.

8.3. ASSIGNING A FLOATING IP ADDRESS TO AN INSTANCE

You can assign a public floating IP address to an instance to enable communication with networks outside the cloud, including the Internet. The cloud administrator configures the available pool of floating IP addresses for an external network. You can allocate a floating IP address from this pool to your project, then associate the floating IP address with your instance.

Projects have a limited quota of floating IP addresses that can be used by instances in the project, 50 by default. Therefore, release IP addresses for reuse when you no longer need them.

Prerequisites

- The instance must be on an external network, or on a project network that is connected to a router that has the external network configured as the gateway.

- The external network that the instance will connect to must have a subnet to provide the floating IP addresses.

Procedure

1. Check the floating IP addresses that are allocated to the current project:
$ openstack floating ip list

If there are no floating IP addresses available that you want to use, allocate a floating IP address to the current project from the external network allocation pool:

$ openstack floating ip create <provider-network>

Replace `<provider-network>` with the name or ID of the external network that you want to use to provide external access.

**TIP**

By default, a floating IP address is randomly allocated from the pool of the external network. A cloud administrator can use the `--floating-ip-address` option to allocate a specific floating IP address from an external network.

2. Assign the floating IP address to an instance:

   $ openstack server add floating ip [--fixed-ip-address <ip_address>] <instance> <floating_ip>

   - Replace `<instance>` with the name or ID of the instance that you want to provide public access to.

   - Replace `<floating_ip>` with the floating IP address that you want to assign to the instance.

   - Optional: Replace `<ip_address>` with the IP address of the interface that you want to attach the floating IP to. By default, this attaches the floating IP address to the first port.

3. Verify that the floating IP address has been assigned to the instance:

   $ openstack server show <instance>

Additional resources

- Creating floating IP pools in the *Networking Guide*.

**8.4. DISASSOCIATING A FLOATING IP ADDRESS FROM AN INSTANCE**

When the instance no longer needs public access, disassociate it from the instance and return it to the allocation pool.

**Procedure**

1. Disassociate the floating IP address from the instance:

   $ openstack server remove floating ip <instance> <ip_address>

   - Replace `<instance>` with the name or ID of the instance that you want to remove public access from.

   - Replace `<floating_ip>` with the floating IP address that is assigned to the instance.
2. Release the floating IP address back into the allocation pool:

   $ openstack floating ip delete <ip_address>

3. Confirm the floating IP address is deleted and is no longer available for assignment:

   $ openstack floating ip list

### 8.5. CREATING AN INSTANCE WITH SSH ACCESS

You can provide SSH access to an instance by specifying a key pair when you create the instance. Key pairs are SSH or x509 credentials that are injected into an instance when it is launched. Each project should have at least one key pair. A key pair belongs to an individual user, not to a project.

**NOTE**

You cannot associate a key pair with an instance after the instance has been created.

**Prerequisites**

- A key pair is available that you can use to SSH into your instances. For more information, see [Generating a new SSH key pair](#).
- The network that you plan to create your instance on must be an external network, or a project network connected to a router that has the external network configured as the gateway. For more information, see [Adding a router](#) in the Networking Guide.
- The external network that the instance connects to must have a subnet to provide the floating IP addresses.
- The security group allows SSH access to instances. For more information, see [Securing instance access with security groups and key pairs](#).
- The image that the instance is based on contains the `cloud-init` package to inject the SSH public key into the instance.
- A floating IP address is available to assign to your instance. For more information, see [Assigning a floating IP address to an instance](#).

**Procedure**

1. Retrieve the name or ID of the flavor that has the hardware profile that your instance requires:

   $ openstack flavor list

   **NOTE**

   Choose a flavor with sufficient size for the image to successfully boot, otherwise the instance will fail to launch.

2. Retrieve the name or ID of the image that has the software profile that your instance requires:

   $ openstack image list
If the image you require is not available, you can download or create a new image. For information about creating or downloading cloud images, see Image service.

3. Retrieve the name or ID of the network that you want to connect your instance to:

   $ openstack network list

4. Retrieve the name of the key pair that you want to use to access your instance remotely:

   $ openstack keypair list

5. Create your instance with SSH access:

   $ openstack server create --flavor <flavor> \\  
   --image <image> --network <network> \\  
   [--security-group <secgroup>] \\  
   --key-name <keypair> --wait myInstancewithSSH

   - Replace <flavor> with the name or ID of the flavor that you retrieved in step 1.
   - Replace <image> with the name or ID of the image that you retrieved in step 2.
   - Replace <network> with the name or ID of the network that you retrieved in step 3. You can use the --network option more than once to connect your instance to several networks, as required.
   - Optional: The default security group is applied to instances that do not specify an alternative security group. You can apply an alternative security group directly to the instance during instance creation, or to a port on the running instance. Use the --security-group option to specify an alternative security group when creating the instance. For information on adding a security group to a port on a running instance, see Adding a security group to a port.
   - Replace <keypair> with the name or ID of the key pair that you retrieved in step 4.

6. Assign a floating IP address to the instance:

   $ openstack server add floating ip myInstancewithSSH <floating_ip>

   Replace <floating_ip> with the floating IP address that you want to assign to the instance.

7. Use the automatically created cloud-user account to verify that you can log in to your instance by using SSH:

   $ ssh -i ~/.ssh/<keypair>.pem cloud-user@<floatingIP>
   [cloud-user@demo-server1 ~]$
You can access an instance from a location external to the cloud by using a remote shell such as SSH or WinRM, when you have allowed the protocol in the instance security group rules. You can also connect directly to the console of an instance, so that you can debug even if the network connection fails.

**NOTE**

If you did not provide a key pair to the instance, or allocate a security group to the instance, you can access the instance only from inside the cloud by using VNC. You cannot ping the instance.

### 9.1. ACCESSING AN INSTANCE CONSOLE

You can connect directly to the VNC console for an instance by entering the VNC console URL in a browser.

**Procedure**

1. To display the VNC console URL for an instance, enter the following command:

   ```bash
   $ openstack console url show <vm_name>
   +-------+------------------------------------------------------+
   | Field | Value                   |                                  |
   +-------+------------------------------------------------------+
   | type  | novnc                    |                                  |
   +-------+------------------------------------------------------+
   
   2. To connect directly to the VNC console, enter the displayed URL in a browser.

### 9.2. LOGGING IN TO AN INSTANCE

You can log in to public instances remotely.

**Prerequisites**

- You have the key pair certificate for the instance. The certificate is downloaded when the key pair is created. If you did not create the key pair yourself, ask your administrator.
- The instance is configured as a public instance. For more information on the requirements of a public instance, see [Providing public access to an instance](#).
- You have a cloud user account.

**Procedure**

1. Retrieve the floating IP address of the instance you want to log in to:

   ```bash
   $ openstack server show <instance>
   ```

   Replace `<instance>` with the name or ID of the instance that you want to connect to.
2. Use the automatically created **cloud-user** account to log in to your instance:

```
$ ssh -i ~/.ssh/<keypair>.pem cloud-user@<floatingIP>
[cloud-user@demo-server1 ~]$
```

- Replace `<keypair>` with the name of the key pair.
- Replace `<floating_ip>` with the floating IP address of the instance.

**TIP**

You can use the following command to log in to an instance without the floating IP address:

```
$ openstack server ssh --login cloud-user \ 
--identity ~/.ssh/<keypair>.pem --private <instance>
```

- Replace `<keypair>` with the name of the key pair.
- Replace `<instance>` with the name or ID of the instance that you want to connect to.
CHAPTER 10. MANAGING AN INSTANCE

You can perform management operations on an instance, such as resizing the instance or shelving the instance. For a complete list of management operations, see Instance management operations.

10.1. RESIZING AN INSTANCE

You can resize an instance if you need to increase or decrease the memory or CPU count of the instance. To resize an instance, select a new flavor for the instance that has the required capacity. Resizing an instance rebuilds and restarts the instance.

Procedure

1. Retrieve the name or ID of the instance that you want to resize:

   $ openstack server list

2. Retrieve the name or ID of the flavor that you want to use to resize the instance:

   $ openstack flavor list

3. Resize the instance:

   $ openstack server resize --flavor <flavor> \
   | --wait <instance>

   - Replace <flavor> with the name or ID of the flavor that you retrieved in step 2.
   - Replace <instance> with the name or ID of the instance that you are resizing.

   NOTE

   Resizing can take time. The operating system on the instance performs a controlled shutdown before the instance is powered off and the instance is resized. During this time, the instance status is RESIZE:

   $ openstack server list
   +----------------------+----------------+--------+----------------------------+
   | ID                   | Name           | Status | Networks                                |
   +----------------------+----------------+--------+----------------------------+
   | 67bc9a9a-5928-47c... | myCirrosServer | RESIZE | admin_internal_net=192.168.111.139 |
   +----------------------+----------------+--------+----------------------------+

4. When the resize completes, the instance status changes to VERIFY_RESIZE. You must now either confirm or revert the resize:

   - To confirm the resize, enter the following command:

     $ openstack server resize confirm <instance>

   - To revert the resize, enter the following command:
$ openstack server resize revert <instance>

The instance is reverted to the original flavor and the status is changed to **ACTIVE**.

**NOTE**

The cloud might be configured to automatically confirm instance resizes if you do not confirm or revert within a configured time frame.

### 10.2. CREATING AN INSTANCE SNAPSHOT

A snapshot is an image that captures the state of the running disk of an instance. You can take a snapshot of an instance to create an image that you can use as a template to create new instances. Snapshots allow you to create new instances from another instance, and restore the state of an instance. If you delete an instance on which a snapshot is based, you can use the snapshot image to create a new instance to the same state as the snapshot.

**Procedure**

1. Retrieve the name or ID of the instance that you want to take a snapshot of:

   ```
   $ openstack server list
   ```

2. Create the snapshot:

   ```
   $ openstack server image create --name <image_name> <instance>
   ```

   - Replace `<image_name>` with a name for the new snapshot image.
   - Replace `<instance>` with the name or ID of the instance that you want to create the snapshot from.

3. Optional: To ensure that the disk state is consistent when you use the instance snapshot as a template to create new instances, enable the QEMU guest agent and specify that the filesystem must be quiesced during snapshot processing by adding the following metadata to the snapshot image:

   ```
   $ openstack image set --property hw_qemu_guest_agent=yes \ --property os_require_quiesce=yes <image_name>
   ```

   The QEMU guest agent is a background process that helps management applications execute instance OS level commands. Enabling this agent adds another device to the instance, which consumes a PCI slot, and limits the number of other devices you can allocate to the instance. It also causes Windows instances to display a warning message about an unknown hardware device.

### 10.3. RESCUING AN INSTANCE

In an emergency such as a system failure or access failure, you can put an instance in rescue mode. This shuts down the instance, reboots it with a new instance disk, and mounts the original instance disk and config drive as a volume on the rebooted instance. You can connect to the rebooted instance to view the original instance disk to repair the system and recover your data.
1. Perform the instance rescue:

```
$ openstack server rescue [-image <image>] <instance>
```

- Optional: By default, the instance is booted from a rescue image provided by the cloud admin, or a fresh copy of the original instance image. Use the `-image` option to specify an alternative image to use when rebooting the instance in rescue mode.

- Replace `<instance>` with the name or ID of the instance that you want to rescue.

2. Connect to the rescued instance to fix the issue.

3. Restart the instance from the normal boot disk:

```
$ openstack server unrescue <instance>
```

10.4. SHELVING AN INSTANCE

Shelving is useful if you have an instance that you are not using, but that you do not want to delete. When you shelve an instance, you retain the instance data and resource allocations, but clear the instance memory. Depending on the cloud configuration, shelved instances are moved to the `SHELVED_OFFLOADED` state either immediately or after a timed delay. When `SHELVED_OFFLOADED`, the instance data and resource allocations are deleted.

When you shelve an instance, the Compute service generates a snapshot image that captures the state of the instance, and allocates a name to the image in the following format: `<instance>-shelved`. This snapshot image is deleted when the instance is unshelved or deleted.

If you no longer need a shelved instance, you can delete it. You can shelve more than one instance at a time.

Procedure

1. Retrieve the name or ID of the instance or instances that you want to shelve:

```
$ openstack server list
```

2. Shelve the instance or instances:

```
$ openstack server shelve <instance> [<instance> ...]
```

Replace `<instance>` with the name or ID of the instance that you want to shelve. You can specify more than one instance to shelve, as required.

3. Verify that the instance has been shelved:

```
$ openstack server list
```

Shelved instances have status `SHELVED_OFFLOADED`.

10.5. INSTANCE MANAGEMENT OPERATIONS
After you create an instance, you can perform the following management operations.

Table 10.1. Management operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop an instance</td>
<td>Stops the instance.</td>
<td><code>openstack server stop</code></td>
</tr>
<tr>
<td>Start an instance</td>
<td>Starts a stopped instance.</td>
<td><code>openstack server start</code></td>
</tr>
<tr>
<td>Pause a running instance</td>
<td>Immediately pause a running instance. The state of the instance is stored in memory (RAM). The paused instance continues to run in a frozen state. You are not prompted to confirm the pause action.</td>
<td><code>openstack server pause</code></td>
</tr>
<tr>
<td>Resume running of a paused instance</td>
<td>Immediately resume a paused instance. You are not prompted to confirm the resume action.</td>
<td><code>openstack server unpause</code></td>
</tr>
<tr>
<td>Suspend a running instance</td>
<td>Immediately suspend a running instance. The state of the instance is stored on the instance disk. You are not prompted to confirm the suspend action.</td>
<td><code>openstack server suspend</code></td>
</tr>
<tr>
<td>Resume running of a suspended instance</td>
<td>Immediately resume a suspended instance. The state of the instance is stored on the instance disk. You are not prompted to confirm the resume action.</td>
<td><code>openstack server resume</code></td>
</tr>
</tbody>
</table>
## Operation

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete an instance</td>
<td>Permanently destroy the instance. You are not prompted to confirm the destroy action. Deleted instances are not recoverable unless the cloud has been configured to enable soft delete.</td>
<td><code>openstack server delete</code></td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deleting an instance does not delete its attached volumes. You must delete attached volumes separately. For more information, see Delete a Volume in the Storage Guide.</td>
<td></td>
</tr>
<tr>
<td>Edit the instance metadata</td>
<td>You can use instance metadata to specify the properties of an instance. For more information, see Creating a customized instance.</td>
<td><code>openstack server set --property &lt;key=value&gt; [--property &lt;key=value&gt;] &lt;instance&gt;</code></td>
</tr>
<tr>
<td>Add security groups</td>
<td>Adds the specified security group to the instance.</td>
<td><code>openstack server add security group</code></td>
</tr>
<tr>
<td>Remove security groups</td>
<td>Removes the specified security group from the instance.</td>
<td><code>openstack remove security group</code></td>
</tr>
<tr>
<td>Rescue an instance</td>
<td>In an emergency such as a system failure or access failure, you can put an instance in rescue mode. This shuts down the instance and mounts the root disk to a temporary server. You can connect to the temporary server to repair the system and recover your data. It is also possible to reboot a running instance into rescue mode. For example, this operation might be required if a filesystem of an instance becomes corrupted.</td>
<td><code>openstack server rescue</code></td>
</tr>
<tr>
<td>Operation</td>
<td>Description</td>
<td>Command</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Restore a rescued instance</td>
<td>Reboots the rescued instance.</td>
<td><code>openstack server unrescue</code></td>
</tr>
<tr>
<td>View instance logs</td>
<td>View the most recent section of the instance console log.</td>
<td><code>openstack console log show</code></td>
</tr>
<tr>
<td>Shelve an instance</td>
<td>When you shelve an instance you retain the instance data and resource allocations, but clear the instance memory. Depending on the cloud configuration, shelved instances are moved to the <code>SHELVED_OFFLOADED</code> state either immediately or after a timed delay. When an instance is in the <code>SHELVED_OFFLOADED</code> state, the instance data and resource allocations are deleted. The state of the instance is stored on the instance disk. If the instance was booted from volume, it goes to <code>SHELVED_OFFLOADED</code> immediately. You are not prompted to confirm the shelve action.</td>
<td><code>openstack server shelve</code></td>
</tr>
<tr>
<td>Unshelve an instance</td>
<td>Restores the instance using the disk image of the shelved instance.</td>
<td><code>openstack server unshelve</code></td>
</tr>
<tr>
<td>Lock an instance</td>
<td>Lock an instance to prevent non-admin users from executing actions on the instance.</td>
<td><code>openstack server lock</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>openstack server unlock</code></td>
</tr>
<tr>
<td>Soft reboot an instance</td>
<td>Gracefully stop and restart the instance. A soft reboot attempts to gracefully shut down all processes before restarting the instance. By default, when you reboot an instance it is a soft reboot.</td>
<td><code>openstack server reboot --soft &lt;server&gt;</code></td>
</tr>
<tr>
<td>Hard reboot an instance</td>
<td>Stop and restart the instance. A hard reboot shuts down the power to the instance and then turns it back on.</td>
<td><code>openstack server reboot --hard &lt;server&gt;</code></td>
</tr>
</tbody>
</table>
Rebuild an instance

Use new image and disk-partition options to rebuild the instance, which involves an instance shut down, re-image, and reboot. Use this option if you encounter operating system issues, rather than terminating the instance and starting over.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rebuild an instance</td>
<td>Use new image and disk-partition options to rebuild the instance, which involves an instance shut down, re-image, and reboot. Use this option if you encounter operating system issues, rather than terminating the instance and starting over.</td>
<td>openstack server rebuild</td>
</tr>
</tbody>
</table>
CHAPTER 11. CREATING A CUSTOMIZED INSTANCE

Cloud users can specify additional data to use when they launch an instance, such as a shell script that the instance runs on boot. The cloud user can use the following methods to pass data to instances:

User data

Use to include instructions in the instance launch command for cloud-init to execute.

Instance metadata

A list of key-value pairs that you can specify when you create or update an instance. You can access the additional data passed to the instance by using a config drive or the metadata service.

Config drive

You can attach a config drive to an instance when it boots. The config drive is presented to the instance as a read-only drive. The instance can mount this drive and read files from it. You can use the config drive as a source for cloud-init information. Config drives are useful when combined with cloud-init for server bootstrapping, and when you want to pass large files to your instances. For example, you can configure cloud-init to automatically mount the config drive and run the setup scripts during the initial instance boot. Config drives are created with the volume label of config-2, and attached to the instance when it boots. The contents of any additional files passed to the config drive are added to the user_data file in the openstack/{version}/ directory of the config drive. cloud-init retrieves the user data from this file.

Metadata service

Provides a REST API to retrieve data specific to an instance. Instances access this service at 169.254.169.254 or at fe80::a9fe:a9fe.

cloud-init can use both a config drive and the metadata service to consume the additional data for customizing an instance. The cloud-init package supports several data input formats. Shell scripts and the cloud-config format are the most common input formats:

- Shell scripts: The data declaration begins with #! or Content-Type: text/x-shellscript. Shell scripts are invoked last in the boot process.

- cloud-config format: The data declaration begins with #cloud-config or Content-Type: text/cloud-config. cloud-config files must be valid YAML to be parsed and executed by cloud-init.

NOTE

cloud-init has a maximum user data size of 16384 bytes for data passed to an instance. You cannot change the size limit, therefore use gzip compression when you need to exceed the size limit.

Vendor-specific data

The RHOSP administrator can also pass data to instances when they are being created. This data may not be visible to you as the cloud user, for example, a cryptographic token that registers the instance with Active Directory.

The RHOSP administrator uses the vendordata feature to pass data to instances. Vendordata configuration is read only, and is located in one of the following files:

- /openstack/{version}/vendor_data.json
11.1. CUSTOMIZING AN INSTANCE BY USING USER DATA

You can use user data to include instructions in the instance launch command. \texttt{cloud-init} executes these commands to customize the instance as the last step in the boot process.

**Procedure**

1. Create a file with instructions for \texttt{cloud-init}. For example, create a bash script that installs and enables a web server on the instance:

   ```bash
   $ vim /home/scripts/install_httpd
   #!/bin/bash
   yum -y install httpd python-psycopg2
   systemctl enable httpd --now
   ```

2. Launch an instance with the \texttt{--user-data} option to pass the bash script:

   ```bash
   $ openstack server create \n   --image rhel8 \n   --flavor default \n   --nic net-id=web-server-network \n   --security-group default \n   --key-name web-server-keypair \n   --user-data /home/scripts/install_httpd \n   --wait web-server-instance
   ```

3. When the instance state is active, attach a floating IP address:

   ```bash
   $ openstack floating ip create web-server-network
   $ openstack server add floating ip web-server-instance 172.25.250.123
   ```

4. Log in to the instance with SSH:

   ```bash
   $ ssh -i ~/.ssh/web-server-keypair cloud-user@172.25.250.123
   ```

5. Check that the customization was successfully performed. For example, to check that the web server has been installed and enabled, enter the following command:

   ```bash
   $ curl http://localhost | grep Test
   <title>Test Page for the Apache HTTP Server on Red Hat Enterprise Linux</title>
   <h1>Red Hat Enterprise Linux <strong>Test Page</strong></h1>
   ```

6. Review the \texttt{/var/log/cloud-init.log} file for relevant messages, such as whether or not the \texttt{cloud-init} executed:
11.2. CUSTOMIZING AN INSTANCE BY USING METADATA

You can use instance metadata to specify the properties of an instance in the instance launch command.

Procedure

1. Launch an instance with the `--property <key=value>` option. For example, to mark the instance as a webserver, set the following property:

   ```
   $ openstack server create \
   --image rhel8 \
   --flavor default \
   --property role=webservers \
   --wait web-server-instance
   ```

2. Optional: Add an additional property to the instance after it is created, for example:

   ```
   $ openstack server set \
   --property region=emea \
   --wait web-server-instance
   ```

11.3. CUSTOMIZING AN INSTANCE BY USING A CONFIG DRIVE

You can create a config drive for an instance that is attached during the instance boot process. You can pass content to the config drive that the config drive makes available to the instance.

Procedure

1. Enable the config drive, and specify a file that contains content that you want to make available in the config drive. For example, the following command creates a new instance named `config-drive-instance` and attaches a config drive that contains the contents of the file `my-user-data.txt`:

   ```
   (overcloud)$ openstack server create --flavor m1.tiny \
   --config-drive true \
   --user-data ./my-user-data.txt \
   --image cirros config-drive-instance
   ```

   This command creates the config drive with the volume label of `config-2`, which is attached to the instance when it boots, and adds the contents of `my-user-data.txt` to the `user_data` file in the `openstack/{version}/` directory of the config drive.

2. Log in to the instance.

3. Mount the config drive:

   - If the instance OS uses `udev`:
# mkdir -p /mnt/config
# mount /dev/disk/by-label/config-2 /mnt/config

- If the instance OS does not use **udev**, you need to first identify the block device that corresponds to the config drive:

  # blkid -t LABEL="config-2" -odevice
  /dev/vdb
  # mkdir -p /mnt/config
  # mount /dev/vdb /mnt/config
CHAPTER 12. IMAGE SERVICE

Manage images and storage in Red Hat OpenStack Platform (RHOSP).

A virtual machine image is a file that contains a virtual disk with a bootable operating system installed. Virtual machine images are supported in different formats. The following formats are available in RHOSP:

- **RAW** - Unstructured disk image format.
- **QCOW2** - Disk format supported by QEMU emulator. This format includes QCOW2v3 (sometimes referred to as QCOW3), which requires QEMU 1.1 or higher.
- **ISO** - Sector-by-sector copy of the data on a disk, stored in a binary file.
- **AKI** - Indicates an Amazon Kernel Image.
- **AMI** - Indicates an Amazon Machine Image.
- **ARI** - Indicates an Amazon RAMDisk Image.
- **VDI** - Disk format supported by VirtualBox virtual machine monitor and the QEMU emulator.
- **VHD** - Common disk format used by virtual machine monitors from VMware, VirtualBox, and others.
- **VMDK** - Disk format supported by many common virtual machine monitors.
- **PLOOP** - A disk format supported and used by Virtuozzo to run OS containers.
- **OVA** - Indicates that what is stored in the Image service (glance) is an OVA tar archive file.
- **DOCKER** - Indicates that what is stored in the Image service (glance) is a Docker tar archive of the container file system.

Although ISO is not normally considered a virtual machine image format, because ISOs contain bootable filesystems with an installed operating system, you use them in the same way as other virtual machine image files.

To download the official Red Hat Enterprise Linux cloud images, your account must have a valid Red Hat Enterprise Linux subscription:

- **Red Hat Enterprise Linux 8 KVM Guest Image**
- **Red Hat Enterprise Linux 7 KVM Guest Image**
- **Red Hat Enterprise Linux 6 KVM Guest Image**

If you are not logged in to the Customer Portal, a prompt opens where you must enter your Red Hat account credentials.

12.1. UNDERSTANDING THE IMAGE SERVICE

Red Hat OpenStack Platform (RHOSP) Image service (glance) features.

12.1.1. Supported Image service (glance) back ends
The following Image service (glance) back end scenarios are supported:

- RBD is the default back end when you use Ceph. For more information, see Configuring Ceph Storage in the Advanced Overcloud Customization guide.

- RBD multi-store. For more information, see Deploying the central site in the Distributed compute node and storage deployment guide.

- Object Storage (swift). For more information, see Using an External Object Storage Cluster in the Advanced Overcloud Customization guide.

- Block Storage (cinder). For more information, see Configuring cinder back end for the Image service in the Advanced Overcloud Customization guide. The Image service uses the Block Storage type and back end as the default.

- NFS. For more information, see Configuring NFS Storage in the Advanced Overcloud Customization guide.

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, L2 is recommended.

- Include monitoring and alerts for the mounted share.

- Set underlying FS permissions.
  
  - 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.
  
  - The write permissions must be present in the shared file system that you use as a store.

12.1.2. 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.
By taking advantage of this feature, you can:

- Sign an image using your private key and upload the image, the signature, and a reference to your public key certificate (the verification metadata). The Image service then verifies that the signature is valid.
- Create an image in the Compute service, have the Compute service sign the image, and upload the image and its verification metadata. The Image service again verifies that the signature is valid.
- Request a signed image in the Compute service. The Image service provides the image and its verification metadata, allowing the Compute service to validate the image before booting it.

For information on image signing and verification, see Validating Image Service (glance) images in the Manage Secrets with OpenStack Key Manager guide.

### 12.1.3. Image conversion

Image conversion converts images by calling the task API while importing an image.

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

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

For more information about image conversion, see Enabling image conversion.

**NOTE**

The conversion can be triggered only when importing an image. It does not run when uploading an image. 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
```

### 12.1.4. Image introspection

Every image format comes with a set of metadata embedded inside the image itself. For example, a stream optimized vmdk would contain the following parameters:

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

# Disk DescriptorFile
version=1
CID=d5a0bce5
parentCID=ffffffff
```

48
By introspecting this vmdk, you can easily know that the disk_type is streamOptimized, and the adapter_type is buslogic. These metadata parameters are useful for the consumer of the image. In Compute, the workflow to instantiate a streamOptimized disk is different from the one to instantiate a flat disk. This new feature allows metadata extraction. You can achieve image introspection by calling the task API while importing the image. An administrator can override metadata settings.

12.1.5. Interoperable image import

The interoperable image import workflow enables you to import images in two ways:

- 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.

12.1.6. Improving scalability with Image service caching

Use the glance-api caching mechanism to store copies of images on Image service (glance) API servers and retrieve them automatically to improve scalability. With Image service caching, glance-api can run 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.

12.1.7. Image pre-caching

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

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

The default periodic interval for image pre-caching is 300 seconds. You can increase or decrease the default interval based on your requirements.

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.

For more information about the openstack overcloud deploy command, see Deployment command in the Director Installation and Usage guide.
12.1.7.2. Using a periodic job to pre-cache an image

Prerequisites

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`:

   ```bash
   (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`:

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

3. Connect to `controller-0`:

   ```bash
   $ ssh heat-admin@192.168.24.71
   ```

4. On `controller-0` as the `heat-admin` user, identify the IP address of the `glance_api` service. In the following example, the IP address is `172.25.1.105`:

   ```bash
   (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/heat-admin` on `controller-0` with the following contents:

   ```bash
   sudo podman exec -ti
   -e NOVA_VERSION=$NOVA_VERSION\n   -e COMPUTE_API_VERSION=$COMPUTE_API_VERSION\n   -e OS_USERNAME=$OS_USERNAME\n   -e OS_PROJECT_NAME=$OS_PROJECT_NAME
   ```
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.

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

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

```bash
()
```

Procedure

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

   ```bash
   $ 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 is deployed with 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.

Related information

You can use additional **glance-cache-manage** commands for the following purposes:

- **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.

12.1.8. Secure metadef APIs

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

Metadef APIs can leak information to unauthorized users. A malicious user can exploit the lack of restrictions and fill the Image service (glance) database with unlimited resources, which can create a Denial of Service (DoS) style attack.

Image service policies control metadef APIs. However, the default policy setting for metadef APIs allows all users to create or read the metadef information. Because metadef resources are not isolated to the owner, metadef resources with potentially sensitive names, such as internal infrastructure details or customer names, can expose that information to malicious users.

12.1.8.1. Configuring a policy to restrict metadef APIs

To make the Image service (glance) more secure, restrict metadef modification APIs to admin-only access by default in your Red Hat OpenStack Platform (RHOSP) deployments.

Procedure

1. As a cloud administrator, create a separate heat template environment file, such as **lock-down-glance-metadef-api.yaml**, to contain policy overrides for the Image service metadef API:

   ```yaml
   ...
   parameter_defaults:
     GlanceApiPolicies: {
       glance-metadef_default: { key: 'metadef_default', value: "" },
       glance-metadef_admin: { key: 'metadef_admin', value: 'role:admin' },
       glance-get_metadef_namespace: { key: 'get_metadef_namespace', value: 'rule:metadef_default' },
   ```
2. Include the environment file that contains the policy overrides in the deployment command with the `-e` option when you deploy the overcloud:

```
$ openstack overcloud deploy -e lock-down-glance-metadef-api.yaml
```

### 12.1.8.2. Enabling metadef APIs

If you previously restricted metadata definition (metadef) APIs or want to relax the new defaults, you can override metadef modification policies to allow users to update their respective resources.
IMPORTANT

Cloud administrators with users who depend on write access to the metadef APIs can make those APIs accessible to all users. 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:

   ```yaml
   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' },
   }
   ```
NOTE
You must configure all metadef policies to use `rule:metadef_default`.

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

12.2. MANAGE IMAGES

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 a template to commission new servers quickly and more consistently than installing a server operating system and individually configuring services.

12.2.1. Creating an image

Manually create Red Hat OpenStack Platform (RHOSP) compatible images in the QCOW2 format by using Red Hat Enterprise Linux 7 ISO files, Red Hat Enterprise Linux 6 ISO files, or Windows ISO files.

12.2.1.1. Use a KVM guest image with Red Hat OpenStack Platform

You can use a ready RHEL KVM guest QCOW2 image:

- Red Hat Enterprise Linux 8 KVM Guest Image
- Red Hat Enterprise Linux 7 KVM Guest Image
- Red Hat Enterprise Linux 6 KVM Guest Image

These images are configured with `cloud-init` and must take advantage of ec2-compatible metadata services for provisioning SSH keys to function correctly.

Ready Windows KVM guest QCOW2 images are not available.

NOTE

For the KVM guest images:

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

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

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

If you want to create custom Red Hat Enterprise Linux or Windows images, see Create a Red Hat Enterprise Linux 7 Image, Create a Red Hat Enterprise Linux 6 Image, or Create a Windows Image.

12.2.1.2. Create custom Red Hat Enterprise Linux or Windows images

Prerequisites

- Linux host machine to create an image. This can be any machine on which you can install and run the Linux packages.
- libvirt, virt-manager to install all packages necessary to create a guest operating system:
  
  ```bash
  $ sudo dnf groupinstall -y @virtualization
  
  $ sudo dnf install -y libguestfs-tools-c
  ```
- Libguestfs tools to install a set of tools to access and modify virtual machine images:
  
  ```bash
  $ sudo dnf install -y libguestfs-tools-c
  ```
- A Red Hat Enterprise Linux 7 or 6 ISO file. For more information, see RHEL 7.2 Binary DVD or RHEL 6.8 Binary DVD or a Windows ISO file. If you do not have a Windows ISO file, see Microsoft TechNet Evaluation Center to download an evaluation image.
- Text editor, if you want to change the kickstart files (RHEL only).

**IMPORTANT**

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

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

12.2.1.2.1. Create a Red Hat Enterprise Linux 7 Image

Manually create a Red Hat OpenStack Platform (RHOSP) compatible image in the QCOW2 format by using a Red Hat Enterprise Linux 7 ISO file.

**NOTE**

You must run all commands with the [root@host]# on your host machine.

1. Start the installation by using virt-install:

   ```bash
   [root@host]# qemu-img create -f qcow2 rhel7.qcow2 8G
   [root@host]# virt-install --virt-type kvm --name rhel7 --ram 2048
   --cdrom /tmp/rhel-server-7.2-x86_64-dvd.iso
   ```
This launches an instance and starts the installation process.

**NOTE**

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

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

2. Configure the instance:
   
   a. At the initial Installer boot menu, select **Install Red Hat Enterprise Linux 7**
   
   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. For software selection, select **Minimal Install**.
   
   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.

   **Result**
   
   The installation process completes and the **Complete!** screen is displayed.

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

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

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

5. Reboot the machine.

6. Register the machine with the Content Delivery Network.

   ```
   # sudo subscription-manager register
   # sudo subscription-manager attach --pool=Valid-Pool-Number-123456
   # sudo subscription-manager repos --enable=rhel-7-server-rpms
   ```
7. Update the system:

```
    # dnf -y update
```

8. Install the `cloud-init` packages:

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

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

```
    - resolv-conf
```

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

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

```
    NOZEROCONF=yes
```

11. To ensure 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"
```

12. Run the `grub2-mkconfig` command:

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

The output is as follows:

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

13. 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
```

14. Power off the instance:

```
    # poweroff
```
15. 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 rhel7
   ```

16. Reduce the image size by converting any free space within the disk image back to free space within the host:

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

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

   The `rhel7-cloud.qcow2` image file is ready to be uploaded to the Image service. For more information about uploading this image to your RHOSP deployment by using the dashboard, see Upload an Image.

### 12.2.1.2.2. Create a Red Hat Enterprise Linux 6 Image

Manually create a Red Hat OpenStack Platform (RHOSP) compatible image in the QCOW2 format by using a Red Hat Enterprise Linux 6 ISO file.

**NOTE**

You must run all commands with the `[root@host]#` on your host machine.

1. Start the installation by using `virt-install`:

   ```
   [root@host]# qemu-img create -f qcow2 rhel6.qcow2 4G
   [root@host]# virt-install --connect=qemu:///system --network=bridge:virbr0 \
   --name=rhel6 --os-type linux --os-variant rhel6 \
   --disk path=rhel6.qcow2,format=qcow2,size=10,cache=none \
   --ram 4096 --vcpus=2 --check-cpu --accelerate \
   --hvm --cdrom=rhel-server-6.8-x86_64-dvd.iso
   ```

   This launches an instance and starts the installation process.

   **NOTE**

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

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

2. Configure the instances:

   a. At the initial Installer boot menu, select Install or upgrade an existing system and follow the installation prompts. Accept the defaults.

      The disk installer provides an option to test your installation media before installation. Select OK to run the test or Skip to proceed without testing.

   b. Choose the appropriate Language and Keyboard options.

   c. When prompted about which type of devices your installation uses, 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. Based on the space on the disk, choose the type of installation you want from the options in the Which type of installation would you like? window.

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

h. The installation process completes and the Congratulations, your Red Hat Enterprise Linux installation is complete screen is displayed.

3. Reboot the instance and log in as the root user.

4. 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
   ```

5. Reboot the machine.

6. Register the machine with the Content Delivery Network:

   ```
   # sudo subscription-manager register
   # sudo subscription-manager attach --pool=Valid-Pool-Number-123456
   # sudo subscription-manager repos --enable=rhel-6-server-rpms
   ```

7. Update the system:

   ```
   # dnf -y update
   ```

8. Install the cloud-init packages:

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

9. 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 configuration file when an instance boots for the first time. This file contains information related to the instance such as nameservers, domain, and other options.

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

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

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

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

```
NOZEROCONF=yes
```

12. To ensure 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:

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

13. Deregister the virtual machine 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
```

14. Power off the instance:

```
# poweroff
```

15. 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 rhel6
```

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

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

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

**NOTE**

You 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 `rhel6-cloud.qcow2` image file is ready to be uploaded to the Image service. For more information about uploading this image to your RHOSP deployment by using the dashboard, see [Upload an Image](#).

### 12.2.1.2.3. Create a Windows image

Manually create a Red Hat OpenStack Platform (RHOSP) compatible image in the QCOW2 format by using a Windows ISO file.

**NOTE**

You must run all commands with the `[root@host]#` on your host machine.
Procedure

1. Start the installation by using `virt-install`:

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

   Replace the following values of the `virt-install` parameters:
   - `<name>` – the name that the Windows instance has.
   - `<size>` – disk size in GB.
   - `<path>` – the path to the Windows installation ISO file.
   - `<RAM>` – the requested amount of RAM in MB.

   **NOTE**
   The `--os-type=windows` parameter ensures that the clock is configured correctly for the Windows guest, 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.

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

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

   Replace `<filename>` 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.

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

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

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

3. To allow the newly installed Windows system to use the virtualized hardware, you might need to install VirtIO drivers. To do so, first install the image, which you must attach as a CD-ROM drive to the Windows instance. To install the `virtio-win` package you must add the VirtIO ISO image to the instance, and install the VirtIO drivers. For more information, see Installing KVM paravirtualized drivers for Windows virtual machines in the Configuring and managing virtualization guide.

4. To complete the configuration, download and execute `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 guest unique by generating an OS ID, which is used by certain Microsoft services.

**IMPORTANT**

Red Hat does not provide technical support for Cloudbase-Init. If you encounter an issue, see [contact Cloudbase Solutions](https://www.cloudbase.net).

When the Windows system shuts down, the `<name>.qcow2` image file is ready to be uploaded to the Image service. For more information about uploading this image to your RHOSP deployment by using the dashboard or the command line, see [Upload an Image](https://www.redhat.com!

**NOTE**

**libosinfo data**

The Compute Service 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 [Appendix A, Image Configuration Parameters](https://www.redhat.com).

### 12.2.2. Upload an image

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

2. Click **Create Image**.

3. Fill out the values, and click **Create Image**.

**Table 12.1. Image options**

<table>
<thead>
<tr>
<th>Field</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name for the image. The name must be unique within the project.</td>
</tr>
<tr>
<td>Field</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Description</td>
<td>Brief description to identify the image.</td>
</tr>
<tr>
<td>Image Source</td>
<td>Image source: <strong>Image Location</strong> or <strong>Image File</strong>. Based on your selection, the next field is displayed.</td>
</tr>
</tbody>
</table>
| Image Location or Image File | - Select **Image Location** option to specify the image location URL.  
- Select **Image File** option to upload an image from the local disk.                                                                                      |
| Format                       | Image format (for example, qcow2).                                                                                                                                                                  |
| Architecture                 | Image architecture. For example, use i686 for a 32-bit architecture or x86_64 for a 64-bit architecture.                                                                                           |
| Minimum Disk (GB)            | Minimum disk size required to boot the image. If this field is not specified, the default value is 0 (no minimum).                                                                               |
| Minimum RAM (MB)             | Minimum memory size required to boot the image. If this field is not specified, the default value is 0 (no minimum).                                                                               |
| Public                       | If selected, makes the image public to all users with access to the project.                                                                                                                        |
| Protected                    | If selected, ensures only users with specific permissions can delete this image.                                                                                                                     |

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

**NOTE**

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

### 12.2.3. Update an image

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

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

3. Update the fields and click Update Image. You can update the following values - name, description, kernel ID, ramdisk ID, architecture, format, minimum disk, minimum RAM, public, protected.

4. Click the drop-down menu and select Update Metadata option.

5. Specify metadata by adding items from the left column to the right one. In the left column, there are metadata definitions from the Image Service Metadata Catalog. Select Other to add metadata with the key of your choice and click Save when finished.

NOTE

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

12.2.4. Import an image

You can import images into the Image service (glance) using web-download to import an image from a URI and glance-direct to import an image from a local file system. The web-download method is enabled by default.

Import methods are configured by the cloud administrator. Run the glance import-info command to list available import options.

12.2.4.1. Import from a remote URI

You can use the web-download method to copy an image from a remote URI.

1. Create an image and specify the URI of the image to import.

   glance image-create --uri <URI>

2. You can monitor the image availability by using the glance image-show <image_id> command. Replace <image_id> with the ID you provided during image creation.

The Image service web download method uses a two-stage process to perform the import. First, it creates an image record. Second, it retrieves the image the specified URI. This method provides a more secure way to import images than the deprecated copy-from method used in Image API v1.

The URI is subject to optional denylist and allowlist filtering as described in the Advanced Overcloud Customization guide.

The Image Property Injection plugin may inject metadata properties to the image as described in the Advanced Overcloud Customization guide. These injected properties determine which compute nodes the image instances are launched on.

12.2.4.2. Import from a local volume
The **glance-direct** method creates an image record, which generates an image ID. After the image is uploaded to the service from a local volume, it is stored in a staging area and is made active after it passes any configured checks. The **glance-direct** method requires a shared staging area when used in a highly available (HA) configuration.

**NOTE**

Image uploads that use the **glance-direct** method fail in an HA environment if a common staging area is not present. In an 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. For more information about configuring the staging area, see Storage Configuration section in the Advanced Overcloud Customization Guide.

The glance-direct 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 these calls in one command. In the example below, replace uppercase words with the appropriate options.

```
[ ]
glance image-create-via-import --container-format FORMAT --disk-format DISKFORMAT --name NAME --file /PATH/TO/IMAGE
```

After the image moves from the staging area to the back end location, the image is listed. However, it might take some time for the image to become active.

You can monitor the image availability by using the **glance image-show <image_id>** command. Replace `<image_id>` with the ID you provided during image creation.

### 12.2.5. Delete an image

1. In the dashboard, select Project > Compute > Images
2. Select the image you want to delete and click Delete Images.

### 12.2.6. Hide or unhide an image

You can hide public images from normal listings presented to users. For instance, 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 hide an image:

```
[ ]
glance image-update <image_id> --hidden 'true'
```

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

To unhide an image:
12.2.7. Show hidden images

To list hidden images:

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

12.2.8. Enabling image conversion

With the `GlanceImageImportPlugins` parameter enabled, you can upload a QCOW2 image, and the Image service will convert it to RAW.

**NOTE**

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

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

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

12.2.9. 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
   ```
12.2.9.1. Configuring disk formats in the Image service (glance)

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`:

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

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

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

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

     ```yaml
     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 \n   -e <overcloud_environment_files> \n   -e <new_environment_file> \n   ...
   ```

   - 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 service*.

12.2.10. Storing an image in RAW format
With the GlanceImageImportPlugins parameter enabled, run the following command to store a previously created image in RAW format:

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

- For **--name**, replace **NAME** with the name of the image; this is the name that will appear in `glance image-list`.
- For **--uri**, 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.
CHAPTER 13. IMAGE SERVICE WITH MULTIPLE STORES

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

13.1. REQUIREMENTS OF STORAGE EDGE ARCHITECTURE

- A copy of each image must exist in the Image service at the central location.
- Prior to creating an instance at an edge site, you must have a local copy of the image at that edge site.
- Images uploaded to an edge site must be copied to the central location before they can be copied to other edge sites.
- Use the Image service RBD driver for all edge sites. Mixed architecture is not supported.
- 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.

13.2. IMPORT AN IMAGE TO MULTIPLE STORES

Use the interoperable image import workflow to import image data into multiple Ceph Storage clusters. You can import images into the Image service 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.

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 Distributed compute node and storage deployment guide.

13.2.1. Manage 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.

### 13.2.2. 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**

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 STORE1,STORE2,STORE3
   ```

   - Replace `IMAGE-NAME` with the name of the image you want to import.
   - Replace `URI` with the URI of the image.
   - Replace `STORE1`, `STORE2`, and `STORE3` 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. For more information, see [Storing an image in RAW format](#).

### 13.2.3. 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 \n--name IMAGE-NAME \n--import-method web-download \n--uri URI \n--stores STORE1,STORE2

- Replace IMAGE-NAME with the name of the image you want to import.
- Replace URI with the URI of the image.
- Replace STORE1, STORE2, and STORE3 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 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 Checking the progress of 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.

13.2.4. Importing image data to a single store
You can import image data to a single store.

Procedure

1. Import image data to a single store:

   $ glance image-create-via-import \n   --container-format bare \n   --name IMAGE-NAME \n   --import-method web-download \n   --uri URI \n   --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.
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.
   ```

### 13.2.5. 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 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 | status | central,dcn0,dcn1 | 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 |

### 13.3. COPY 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 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 STORE1,STORE2`. 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 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](#).

**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.

#### 13.3.1. Copying an image to all stores

Use the following procedure to copy image data to all available stores.
13.3.2. 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 STORE1,STORE2 \
       --import-method copy-image
   ```

   Replace `IMAGE-ID` with the name of the image you want to copy.
   
   Replace `STORE1` and `STORE2` 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 [Checking the progress of the image import operation](#).

13.4. DELETING AN IMAGE FROM A SPECIFIC STORE

This feature enables you to delete an existing image copy on a specific store using Red Hat OpenStack Image service (glance).

**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.
13.5. UNDERSTANDING THE LOCATIONS OF IMAGES

Although an image can be present on multiple sites, there is only a single 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 plus the two edge sites.

Procedure

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

   ```bash
   $ 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:

   ```bash
   $ glance image-list --include-stores
   | ID                                   | Name    | Stores
   | 2bd882e7-1da0-4078-97fe-f1bb81f61b00 | cirros | default_backend,dcn1,dcn2
   ``

3. List the image locations properties to show the details of each location:

   ```bash
   $ 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://63df2767-8d8c-4e06-818e-8c155334f487/images/2bd882e7-1da0-4078-97fe-f1bb81f61b00/snap', 'metadata': {'store': 'dcn1'}}, {'url': 'rbd://1b324138-2ef9-4e09-4e09-aa7e6d6ead78/images/2bd882e7-1da0-4078-97fe-f1bb81f61b00/snap', 'metadata': {'store': 'dcn2'}}]'
   | (--- cut ---) |
   ``

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

---

**WARNING**

Using `glance image-delete` will permanently delete the image across all the sites. All image copies will be deleted, as well as the image instance and metadata.
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 CONFIGURATION PARAMETERS

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

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

Table A.1. Property keys

<table>
<thead>
<tr>
<th>Specific to</th>
<th>Key</th>
<th>Description</th>
<th>Supported values</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>architecture</td>
<td>The CPU architecture that must be supported by the hypervisor. For example, x86_64, arm, or ppc64. Run <code>uname -m</code> to get the architecture of a machine.</td>
<td>• alpha - DEC 64-bit RISC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• armv7l - ARM Cortex-A7 MPCore</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• cris - Ethernet, Token Ring, AXis-Code Reduced Instruction Set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• i686 - Intel sixth-generation x86 (P6 micro architecture)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• ia64 - Itanium</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• lm32 - Lattice Micro32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• m68k - Motorola 680000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• microblaze - Xilinx 32-bit FPGA (Big Endian)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• microblazeel - Xilinx 32-bit FPGA (Little Endian)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• mips - MIPS 32-bit RISC (Big Endian)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• mipsel - MIPS 32-bit RISC (Little Endian)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• mips64 - MIPS 64-bit RISC (Big Endian)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• mips64el - MIPS 64-bit RISC (Little Endian)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• openrisc - OpenCores RISC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• parisc - HP Precision Architecture RISC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• parisc64 - HP Precision Architecture 64-bit RISC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• ppc - PowerPC 32-bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• ppc64 - PowerPC 64-bit</td>
</tr>
<tr>
<td>Specific to</td>
<td>Key</td>
<td>Description</td>
<td>Supported values</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>All</td>
<td>hypervisor_type</td>
<td>The hypervisor type.</td>
<td>kvm, vmware</td>
</tr>
<tr>
<td>All</td>
<td>instance_uuid</td>
<td>For snapshot images, this is the UUID of the server used to create this image.</td>
<td>Valid server UUID</td>
</tr>
<tr>
<td>All</td>
<td>kernel_id</td>
<td>The ID of an image stored in the Image Service that should be used as the</td>
<td>Valid image ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kernel when booting an AMI-style image.</td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>os_distro</td>
<td>The common name of the operating system distribution in lowercase.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>arch - Arch Linux. Do not use archlinux or org.archlinux.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>centos - Community Enterprise Operating System. Do not use org.centos or CentOS.</td>
</tr>
</tbody>
</table>
Specific to | Key | Description | Supported values
---|---|---|---
debian | Debian | Do not use Debian or org.debian.
fedora | Fedora | Do not use Fedora, org.fedora, or org.fedoraproject.
freebsd | FreeBSD | Do not use org.freebsd, freeBSD, or FreeBSD.
gentoo | Gentoo Linux | Do not use Gentoo or org.gentoo.
mandrake | Mandrake Linux (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.
etbsd | NetBSD | Do not use NetBSD or org.netbsd.
netware | Novell NetWare | Do not use novell or NetWare.
opensuse | openSUSE | Do not use suse, SuSE, or org.opensuse.
opensolaris | OpenSolaris | Do not use OpenSolaris or org.opensolaris.
rhel | Red Hat Enterprise Linux | Do not use redhat, RedHat, or com.redhat.
sled | SUSE Linux Enterprise Desktop | Do not use com.suse.
<table>
<thead>
<tr>
<th>Specific to</th>
<th>Key</th>
<th>Description</th>
<th>Supported values</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>os_version</td>
<td>The operating system version as specified by the distributor.</td>
<td>Version number (for example, &quot;11.10&quot;)</td>
</tr>
<tr>
<td>All</td>
<td>ramdisk_id</td>
<td>The ID of image stored in the Image Service that should be used as the ramdisk when booting an AMI-style image.</td>
<td>Valid image ID</td>
</tr>
<tr>
<td>All</td>
<td>vm_mode</td>
<td>The virtual machine mode. This represents the host/guest ABI (application binary interface) used for the virtual machine.</td>
<td><strong>hvm</strong>-Fully virtualized. This is the mode used by QEMU and KVM.</td>
</tr>
<tr>
<td>libvirt API driver</td>
<td>hw_disk_bus</td>
<td>Specifies the type of disk controller to attach disk devices to.</td>
<td><strong>scsi, virtio, ide, or usb</strong>. Note that if using <strong>iscsi</strong>, the <strong>hw_scsi_model</strong> needs to be set to <strong>virtio-scsi</strong>.</td>
</tr>
<tr>
<td>libvirt API driver</td>
<td>hw_cdrom_bus</td>
<td>Specifies the type of disk controller to attach CD-ROM devices to.</td>
<td><strong>scsi, virtio, ide, or usb</strong>. If you specify <strong>iscsi</strong>, you must set the <strong>hw_scsi_model</strong> parameter to <strong>virtio-scsi</strong>.</td>
</tr>
<tr>
<td>libvirt API driver</td>
<td>hw_numa_nodes</td>
<td>Number of NUMA nodes to expose to the instance (does not override flavor definition).</td>
<td>Integer.</td>
</tr>
<tr>
<td>libvirt API driver</td>
<td>hw_numa_cpus.0</td>
<td>Mapping of vCPUs N-M to NUMA node 0 (does not override flavor definition).</td>
<td>Comma-separated list of integers.</td>
</tr>
<tr>
<td>libvirt API driver</td>
<td>hw_numa_cpus.1</td>
<td>Mapping of vCPUs N-M to NUMA node 1 (does not override flavor definition).</td>
<td>Comma-separated list of integers.</td>
</tr>
<tr>
<td>libvirt API driver</td>
<td>hw_numa_mem.0</td>
<td>Mapping N MB of RAM to NUMA node 0 (does not override flavor definition).</td>
<td>Integer</td>
</tr>
<tr>
<td>libvirt API driver</td>
<td>hw_numa_mem.1</td>
<td>Mapping N MB of RAM to NUMA node 1 (does not override flavor definition).</td>
<td>Integer</td>
</tr>
<tr>
<td>Specific to</td>
<td>Key</td>
<td>Description</td>
<td>Supported values</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>libvirt API driver</td>
<td>hw_qemu_guest_agent</td>
<td>Guest agent support. If set to yes, and if qemu-ga is also installed, file systems can be quiesced (frozen) and snapshots created automatically.</td>
<td>yes / no</td>
</tr>
<tr>
<td>libvirt API driver</td>
<td>hw_rng_model</td>
<td>Adds a random number generator (RNG) device to instances launched with this image.</td>
<td>virtio, or other supported device.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>libvirt API driver</td>
<td>hw_scsi_model</td>
<td>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.</td>
<td>virtio-scsi</td>
</tr>
</tbody>
</table>
## libvirt API driver

<table>
<thead>
<tr>
<th>Specific to</th>
<th>Key</th>
<th>Description</th>
<th>Supported values</th>
</tr>
</thead>
<tbody>
<tr>
<td>libvirt API driver</td>
<td>hw_video_model</td>
<td>The video device driver to use in virtual machine instances.</td>
<td>List of supported drivers, in order of precedence:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>virtio</strong>. (Recommended)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Virtual GPU with the Gallium GPU specification that uses the VIRGL renderer to render OpenGL. This GPU model is supported in all architectures, and can leverage hardware acceleration if the host has a dedicated GPU. For more information, see <a href="https://virgil3d.github.io/">https://virgil3d.github.io/</a>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>qxl</strong>. High-performance driver for Spice or noVNC environments.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>cirrus</strong>. Legacy driver, use if the QXL driver is not available.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>vga</strong>. Use this driver for IBM Power environments.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>gop</strong>. Not supported for QEMU/KVM environments.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>xen</strong>. Not supported for KVM environments.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>vmvga</strong>. Legacy driver, do not use.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>none</strong>. Use this value to disable emulated graphics or video in virtual GPU (vGPU) instances where the driver is configured separately.</td>
</tr>
<tr>
<td>libvirt API driver</td>
<td>hw_video_ram</td>
<td>Maximum RAM for the video image. Used only if a hw_video:ram_max_m b value has been set in the flavor’s extra_specs and that value is higher than the value set in hw_video_ram.</td>
<td>Integer in MB (for example, 64)</td>
</tr>
<tr>
<td>Specific to</td>
<td>Key</td>
<td>Description</td>
<td>Supported values</td>
</tr>
<tr>
<td>------------</td>
<td>-----</td>
<td>-------------</td>
<td>------------------</td>
</tr>
</tbody>
</table>
| libvirt API driver | hw_watchdog_action | Enables a virtual hardware watchdog device that carries out the specified action if the server hangs. The watchdog uses the i6300esb device (emulating a PCI Intel 6300ESB). If **hw_watchdog_action** is not specified, the watchdog is disabled. | • disabled—The device is not attached. Allows the user to disable the watchdog for the image, even if it has been enabled using the image’s flavor. The default value for this parameter is disabled.  
• reset—Forcefully reset the guest.  
• poweroff—Forcefully power off the guest.  
• pause—Pause the guest.  
• none—Only enable the watchdog; do nothing if the server hangs. |
| libvirt API driver | os_command_line | The kernel command line to be used by the libvirt driver, instead of the default. For Linux Containers (LXC), the value is used as arguments for initialization. This key is valid only for Amazon kernel, ramdisk, or machine images (aki, ari, or ami). | |
| libvirt API driver and VMware API driver | hw_vif_model | Specifies the model of virtual network interface device to use. | The valid options depend on the configured hypervisor.  
• KVM and QEMU: e1000, ne2k_pci, pcnet, rtl8139, and virtio.  
• VMware: e1000, e1000e, VirtualE1000, VirtualE1000e, VirtualPCNet32, VirtualSriovEthernetCard, and VirtualVmxnet.  
• Xen: e1000, netfront, ne2k_pci, pcnet, and rtl8139. |
<p>| VMware API driver | vmware_adaptertype | The virtual SCSI or IDE controller used by the hypervisor. | <strong>IsiLogic, busLogic, or ide</strong> |</p>
<table>
<thead>
<tr>
<th>Specific to</th>
<th>Key</th>
<th>Description</th>
<th>Supported values</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMware API driver</td>
<td>vmware_ostype</td>
<td>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.</td>
<td>For more information, see Images with VMware vSphere.</td>
</tr>
<tr>
<td>VMware API driver</td>
<td>vmware_image_version</td>
<td>Currently unused.</td>
<td>1</td>
</tr>
<tr>
<td>XenAPI driver</td>
<td>auto_disk_config</td>
<td>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.</td>
<td>true / false</td>
</tr>
<tr>
<td>libvirt API driver and XenAPI driver</td>
<td>os_type</td>
<td>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.</td>
<td>linux or windows</td>
</tr>
</tbody>
</table>