



Red Hat OpenStack Platform 17.0

Creating and Managing Instances

Creating and managing instances using the CLI

Red Hat OpenStack Platform 17.0 Creating and Managing Instances

Creating and managing instances using the CLI

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Copyright © 2023 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This guide provides procedures for creating and managing instances.

Table of Contents

PREFACE	4
MAKING OPEN SOURCE MORE INCLUSIVE	5
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	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. CREATING AN INSTANCE WITH NUMA AFFINITY ON THE PORT	16
5.6. ADDITIONAL RESOURCES	17
CHAPTER 6. CREATING AN INSTANCE WITH A GUARANTEED MINIMUM BANDWIDTH QOS	18
6.1. REMOVING A GUARANTEED MINIMUM BANDWIDTH QOS FROM AN INSTANCE	19
CHAPTER 7. CREATING AN INSTANCE WITH A VDPA INTERFACE	20
CHAPTER 8. UPDATING AN INSTANCE	21
8.1. ATTACHING A NETWORK TO AN INSTANCE	21
8.2. DETACHING A NETWORK FROM AN INSTANCE	21
8.3. ATTACHING A PORT TO AN INSTANCE	22
8.4. DETACHING A PORT FROM AN INSTANCE	23
8.5. ATTACHING A VOLUME TO AN INSTANCE	23
8.6. VIEWING THE VOLUMES ATTACHED TO AN INSTANCE	25
8.7. DETACHING A VOLUME FROM AN INSTANCE	25
CHAPTER 9. PROVIDING PUBLIC ACCESS TO AN INSTANCE	27
9.1. PREREQUISITES	27
9.2. SECURING INSTANCE ACCESS WITH SECURITY GROUPS AND KEY PAIRS	27
9.2.1. Creating a security group	28
9.2.2. Updating security group rules	29
9.2.3. Deleting security group rules	29
9.2.4. Adding a security group to a port	30
9.2.5. Removing a security group from a port	30
9.2.6. Deleting a security group	31
9.2.7. Generating a new SSH key pair	31
9.2.8. Importing an existing SSH key pair	32
9.2.9. Additional resources	32
9.3. ASSIGNING A FLOATING IP ADDRESS TO AN INSTANCE	32

9.4. DISASSOCIATING A FLOATING IP ADDRESS FROM AN INSTANCE	33
9.5. CREATING AN INSTANCE WITH SSH ACCESS	34
9.6. ADDITIONAL RESOURCES	36
CHAPTER 10. CONNECTING TO AN INSTANCE	37
10.1. ACCESSING AN INSTANCE CONSOLE	37
10.2. LOGGING IN TO AN INSTANCE	37
CHAPTER 11. MANAGING AN INSTANCE	39
11.1. RESIZING AN INSTANCE	39
11.2. CREATING AN INSTANCE SNAPSHOT	40
11.3. RESCUING AN INSTANCE	40
11.4. SHELIVING AN INSTANCE	41
11.5. INSTANCE MANAGEMENT OPERATIONS	41
CHAPTER 12. CREATING A CUSTOMIZED INSTANCE	46
12.1. CUSTOMIZING AN INSTANCE BY USING USER DATA	47
12.2. CUSTOMIZING AN INSTANCE BY USING METADATA	48
12.3. CUSTOMIZING AN INSTANCE BY USING A CONFIG DRIVE	48

PREFACE



NOTE

You cannot apply a role-based access control (RBAC)-shared security group directly to an instance during instance creation. To apply an RBAC-shared security group to an instance you must first create the port, apply the shared security group to that port, and then assign that port to the instance. See [Adding a security group to a port](#) .

MAKING OPEN SOURCE MORE INCLUSIVE

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

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

Using the Direct Documentation Feedback (DDF) function

Use the **Add Feedback** DDF function for direct comments on specific sentences, paragraphs, or code blocks.

1. View the documentation in the *Multi-page HTML* format.
2. Ensure that you see the **Feedback** button in the upper right corner of the document.
3. Highlight the part of text that you want to comment on.
4. Click **Add Feedback**.
5. Complete the **Add Feedback** field with your comments.
6. Optional: Add your email address so that the documentation team can contact you for clarification on your issue.
7. Click **Submit**.

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.

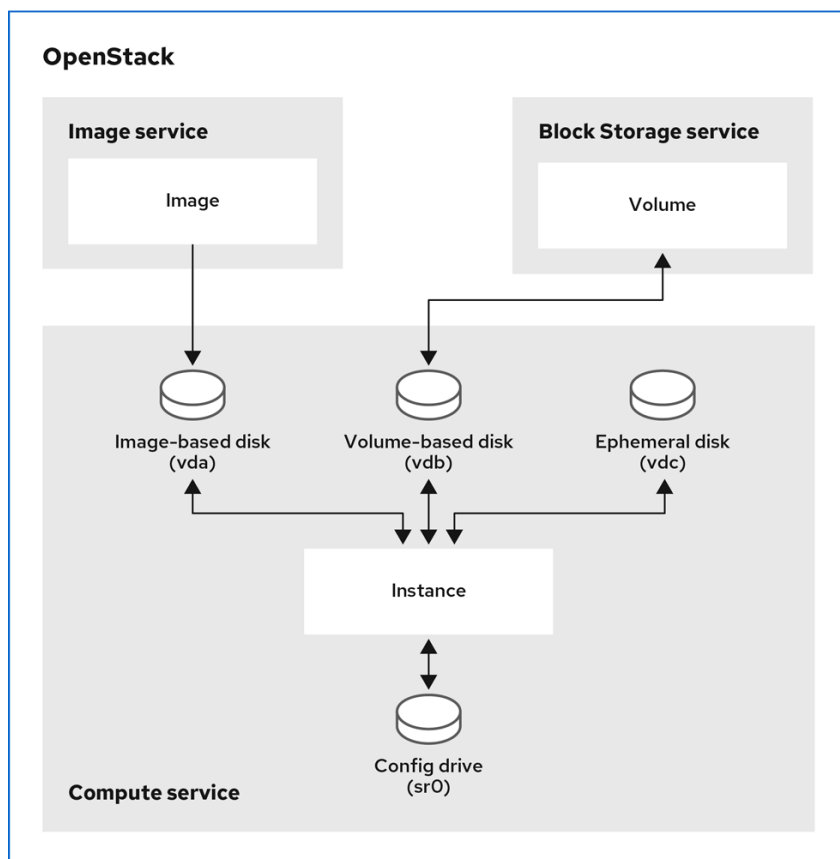
CHAPTER 2. INSTANCE BOOT SOURCE

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.



145_OpenStack_0321

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

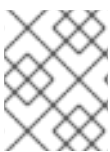
3.1. INSTANCE DISK

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.

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 images](#) in *Creating and Managing Images*.
 - 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:

```
$ 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 [Creating images](#).

**NOTE**

If you need to attach more than 26 volumes to your instance, then the image you use to create your instance must have the following properties:

- **hw_scsi_model=virtio-scsi**
- **hw_disk_bus=scsi**

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 [Creating images](#).

**NOTE**

If you need to attach more than 26 volumes to your instance, then the image you use to create your instance must have the following properties:

- **hw_scsi_model=virtio-scsi**
- **hw_disk_bus=scsi**

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.

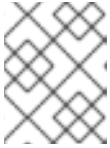
5.4. CREATING AN INSTANCE WITH A SR-IOV NETWORK INTERFACE

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](#) in the *Configuring the Compute Service for Instance Creation* guide. 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 [Creating images](#).

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](#) in the *Configuring the Compute Service for Instance Creation* guide. 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. CREATING AN INSTANCE WITH NUMA AFFINITY ON THE PORT

To create an instance with NUMA affinity on the port you create the port with the required NUMA affinity policy, then specify the port when creating the instance.



NOTE

Port NUMA affinity policies have a higher precedence than flavors, images, and PCI NUMA affinity policies. Cloud operators can set a default NUMA affinity policy for each PCI passthrough device. You can use the instance flavor, image, or port to override the default NUMA affinity policy applied to an instance.

Prerequisites

- The **port-numa-affinity-policy** extension must be enabled in the cloud platform.
- The service plugin must be configured in the Networking service (neutron).

Procedure

1. Create a port with the NUMA affinity policy that you require:

```
$ openstack port create --network <network> \
  [--numa-policy-required | --numa-policy-preferred | --numa-policy-legacy] \
  myNUMAAffinityPort
```

- Replace **<network>** with the name or ID of the tenant network that you want to connect your instance to.
 - Use one of the following options to specify the NUMA affinity policy to apply to the port:
 - **--numa-policy-required** - NUMA affinity policy required to schedule this port.
 - **--numa-policy-preferred** - NUMA affinity policy preferred to schedule this port.
 - **--numa-policy-legacy** - NUMA affinity policy using legacy mode to schedule this port.
2. Create your instance:

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

- Replace **<flavor>** with the name or ID of the flavor that has the hardware profile that you require for your instance.
- Replace **<image>** with the name or ID of the image that has the software profile that you require for your instance.
- Replace **<port>** with the name or ID of the port that you created in step 1.

5.6. ADDITIONAL RESOURCES

- [Creating a customized instance](#)

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 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

- 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-a346-520d7cfe0e2b | policy0 | True   | False   | ba4de51bf7694228a350dd22b7a3dc24 |
+-----+
```

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

```
(overcloud)$ openstack network qos policy show policy0
```

```

-----+
| Field      | Value                                                                 |
-----+
| description |
|
| id          | b80acc64-4fc2-41f2-a346-520d7cfe0e2b                                |
|
| is_default  | False                                                                |
|
| location    | cloud=', project.domain_id=', project.domain_name='Default,
project.id=ba4de51bf7694228a350dd22b7a3dc24, project.name=admin,
region_name=regionOne, zone=
|
| name        | policy0                                                              |
|
| project_id  | ba4de51bf7694228a350dd22b7a3dc24                                    |
|
| rules       | [{min_kbps: 100000, direction: egress, id: d46218fe-9218-4e96-952b-
9f45a5cb3b3c, qos_policy_id: b80acc64-4fc2-41f2-a346-520d7cfe0e2b, type:
minimum_bandwidth}, {min_kbps: 100000, direction: ingress, id: 1202c4e3-a03a-464c-80d5-
0bf90bb74c9d, qos_policy_id: b80acc64-4fc2-41f2-a346-520d7cfe0e2b, type:
minimum_bandwidth}] |
| shared      | True                                                                  |
|
| tags        | []                                                                    |
|
-----+

```

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. CREATING AN INSTANCE WITH A VDPA INTERFACE



IMPORTANT

This feature is available in this release as a *Technology Preview*, and therefore is not fully supported by Red Hat. It should only be used for testing, and should not be deployed in a production environment. For more information about Technology Preview features, see [Scope of Coverage Details](#).

You can create an instance with a VDPA interface by requesting a port for your instance that has a vNIC type of VDPA.

Limitations

- You cannot suspend, live migrate, shelve, or evacuate an instance that has a VDPA interface.
- You cannot detach the VDPA interface from an instance and then reattach it to the instance.

Procedure

1. Create a network that is mapped to the physical network:

```
$ openstack network create vdpa_network \  
--provider-physical-network tenant \  
--provider-network-type vlan \  
--provider-segment 1337
```

2. Create a subnet for the network:

```
$ openstack subnet create vdpa_subnet \  
--network vdpa_net1 \  
--subnet-range 192.0.2.0/24 \  
--dhcp
```

3. Create a port from a VDPA-enabled NIC:

```
$ openstack port create vdpa_direct_port \  
--network vdpa_network \  
--vnic-type vdpa
```

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

```
$ openstack server create vdpa_instance \  
--flavor cirros256 --image cirros-0.3.5-x86_64-disk \  
--nic port-id=vdpa_direct_port --wait
```

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

CHAPTER 8. 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.

8.1. 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*.

8.2. 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:

```
(overcloud)$ openstack server show <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.

8.3. 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

Red Hat OpenStack Platform (RHOSP) provides up to 24 interfaces for each instance. By default, you can add up to 16 PCIe devices to an instance before you must reboot the instance to add more. The RHOSP administrator can use the **NovaLibvirtNumPciePorts** parameter to configure the number of PCIe devices that can be added to an instance, before a reboot of the instance is required to add more devices.

Prerequisites

- If attaching a port with an SR-IOV vNIC to an instance, there must be a free SR-IOV device on the host on the appropriate physical network, and the instance must have a free PCIe slot.

Procedure

1. Create the port that you want to attach to your instance:

```
$ openstack port create --network <network> [--vnic-type <vnic-type>] <port>
```

- Replace **<network>** with the name or ID of the network to create the port on.
- Optional: To create an SR-IOV port, replace **<vnic-type>** with 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 SR-IOV port that is attached to the instance through a MacVTap device.
- 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.

3. Verify that the port is attached to your instance:

```
$ openstack port list --device-id <instance_UUID>
```

Replace **<instance_UUID>** with the UUID of the instance that you attached the port to.

Additional resources

- [openstack port create](#) command in the *Command Line Interface Reference*.

8.4. 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.

8.5. 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 multi-attach volume. For more information about creating multi-attach volumes, see [Volumes that can be attached to multiple instances](#).

Prerequisites

- To attach a multi-attach volume, the environment variable **OS_COMPUTE_API_VERSION** is set to 2.60 or later.
- The instance is fully operational, or fully stopped. You cannot attach a volume to an instance when the instance is in the process of booting up or shutting down.

- To attach more than 26 volumes to your instance, the image you used to create the instance must have the following properties:
 - **hw_scsi_model=virtio-scsi**
 - **hw_disk_bus=scsi**

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

If the command returns the following error, the volume you chose to attach to the instance is a multi-attach volume, therefore you must use Compute API version 2.60 or later:

```
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 include the **--os-compute-api-version** argument when adding the volume to the instance:

```
$ openstack --os-compute-api-version 2.72 server add volume <instance>
<volume>
```

TIP

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

3. Confirm that the volume is attached to the instance or instances:

```
$ openstack volume show <volume>
```

Replace **<volume>** with the name or ID of the volume to display.

Example output:

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

8.6. 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-a85c-854197cd537b	1f9dcb02-9a20-4a4b-9f25-c7846a1ce9e8

8.7. 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](#) procedure to detach the port.

Prerequisites

- The instance is fully operational, or fully stopped. You cannot detach a volume from an instance when the instance is in the process of booting up or shutting down.

Procedure

- Identify the volume that is attached to the instance:

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

- 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**.

CHAPTER 9. 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.

9.1. 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.

9.2. 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.



NOTE

You cannot apply a role-based access control (RBAC)-shared security group directly to an instance during instance creation. To apply an RBAC-shared security group to an instance you must first create the port, apply the shared security group to that port, and then assign that port to the instance. See [Adding a security group to a port](#) .

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.

9.2.1. 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
```

9.2.2. 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:

```
$ openstack security group list
```

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
```

9.2.3. 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.

9.2.4. 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.

9.2.5. 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.

9.2.6. Deleting a security group

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

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> [<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.

9.2.7. 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
```

9.2.8. 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
```

9.2.9. Additional resources

- [Security groups](#) in the *Networking Guide*.
- [Project security management](#) in the *Users and Identity Management Guide*.

9.3. 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*.

9.4. 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
```

9.5. 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.

You can apply a security group directly to an instance during instance creation, or to a port on the running instance.



NOTE

You cannot apply a role-based access control (RBAC)-shared security group directly to an instance during instance creation. To apply an RBAC-shared security group to an instance you must first create the port, apply the shared security group to that port, and then assign that port to the instance. See [Adding a security group to a port](#) .

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 [Creating images](#).

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 ~]$
```

9.6. ADDITIONAL RESOURCES

- [Creating a network](#) in the *Networking Guide*.
- [Adding a router](#) in the *Networking Guide*.

CHAPTER 10. CONNECTING TO AN INSTANCE

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.

10.1. 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:

```
$ openstack console url show <vm_name>
+-----+
| Field | Value |
+-----+
| type | novnc |
| url | http://172.25.250.50:6080/vnc_auto.html?token= |
| | 962dfd71-f047-43d3-89a5-13cb88261eb9 |
+-----+
```

2. To connect directly to the VNC console, enter the displayed URL in a browser.

10.2. 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:

```
$ 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 11. 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](#).

11.1. 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.

11.2. 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.

11.3. 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.

Procedure

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>
```

11.4. SHELIVING 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. Shelf 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**.

11.5. INSTANCE MANAGEMENT OPERATIONS

After you create an instance, you can perform the following management operations.

Table 11.1. Management operations

Operation	Description	Command
Stop an instance	Stops the instance.	<code>openstack server stop</code>
Start an instance	Starts a stopped instance.	<code>openstack server start</code>
Pause a running instance	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.	<code>openstack server pause</code>
Resume running of a paused instance	Immediately resume a paused instance. You are not prompted to confirm the resume action.	<code>openstack server unpause</code>
Suspend a running instance	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.	<code>openstack server suspend</code>
Resume running of a suspended instance	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.	<code>openstack server resume</code>

Operation	Description	Command
Delete an instance	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.  NOTE Deleting an instance does not delete its attached volumes. You must delete attached volumes separately. For more information, see Deleting a Block Storage service volume in the <i>Storage Guide</i>.	openstack server delete
Edit the instance metadata	You can use instance metadata to specify the properties of an instance. For more information, see Creating a customized instance .	openstack server set --property <key=value> [--property <key=value>] <instance>
Add security groups	Adds the specified security group to the instance.	openstack server add security group
Remove security groups	Removes the specified security group from the instance.	openstack remove security group

Operation	Description	Command
Rescue 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 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.	openstack server rescue
Restore a rescued instance	Reboots the rescued instance.	openstack server unrescue
View instance logs	View the most recent section of the instance console log.	openstack console log show
Shelve an instance	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 an instance is in the SHELVED_OFFLOADED 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 SHELVED_OFFLOADED immediately. You are not prompted to confirm the shelve action.	openstack server shelve
Unshelve an instance	Restores the instance using the disk image of the shelved instance.	openstack server unshelve
Lock an instance	Lock an instance to prevent non-admin users from executing actions on the instance.	openstack server lock openstack server unlock

Operation	Description	Command
Soft reboot an instance	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.	openstack server reboot --soft <server>
Hard reboot an instance	Stop and restart the instance. A hard reboot shuts down the power to the instance and then turns it back on.	openstack server reboot --hard <server>
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.	openstack server rebuild

CHAPTER 12. 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**

- `/openstack/{version}/vendor_data2.json`

You can view these files using the metadata service or from the config drive on your instance. To access the files by using the metadata service, make a GET request to either **`http://169.254.169.254/openstack/{version}/vendor_data.json`** or **`http://169.254.169.254/openstack/{version}/vendor_data2.json`**.

12.1. CUSTOMIZING AN INSTANCE BY USING USER DATA

You can use user data to include instructions in the instance launch command. **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 **cloud-init**. For example, create a bash script that installs and enables a web server on the instance:

```
$ vim /home/scripts/install_httpd
#!/bin/bash

yum -y install httpd python-psycpg2
systemctl enable httpd --now
```

2. Launch an instance with the **--user-data** option to pass the bash script:

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

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

```
$ 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:

```
$ 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:

```
$ 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 `/var/log/cloud-init.log` file for relevant messages, such as whether or not the **cloud-init** executed:

```
$ sudo less /var/log/cloud-init.log
...output omitted...
...util.py[DEBUG]: Cloud-init v. 0.7.9 finished at Sat, 23 Jun 2018 02:26:02 +0000.
Datasource DataSourceOpenStack [net,ver=2]. Up 21.25 seconds
```

12.2. 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=webserver \
--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
```

12.3. 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" -o device
/dev/vdb
# mkdir -p /mnt/config
# mount /dev/vdb /mnt/config
```