Nova Documentation

OpenStack Foundation

May 29, 2023

CONTENTS

1 What is nova? 1

2 For End Users 3
- 2.1 Tools for using Nova . . . 3
- 2.2 Writing to the API . . . 3

3 For Operators 5
- 3.1 Architecture Overview . . . 5
- 3.2 Installation . . . 5
- 3.3 Deployment Considerations . . . 42
- 3.4 Maintenance . . . 42
- 3.5 Reference Material . . . 42

4 For Contributors 43
- 4.1 Contributor Documentation . . . 43
- 4.2 Technical Reference Deep Dives . . . 45
- 4.3 Compute . . . 46
- 4.4 Command-line Utilities . . . 261
- 4.5 Configuration Guide . . . 284
- 4.6 Development Quickstart . . . 543
- 4.7 Extending the API . . . 546
- 4.8 Adding a Method to the OpenStack API . . . 550
- 4.9 API reference guideline . . . 551
- 4.10 Blueprints, Specs and Priorities . . . 558
- 4.11 Code Review Guide for Nova . . . 559
- 4.12 Documentation Guidelines . . . 563
- 4.13 Evacuate vs Rebuild . . . 564
- 4.14 API Microversions . . . 565
- 4.15 Development policies . . . 571
- 4.16 Release Notes . . . 573
- 4.17 Test Strategy . . . 574
- 4.18 Testing NUMA related hardware setup with libvirt . . . 576
- 4.19 Testing Serial Console . . . 586
- 4.20 Testing Zero Downtime Upgrade Process . . . 587
- 4.21 Testing Down Cells . . . 591
- 4.22 Profiling With Eventlet . . . 596
- 4.23 How to get (more) involved with Nova . . . 600
- 4.24 Nova team process . . . 605
- 4.25 Scope of the Nova project . . . 619
- 4.26 Chronological PTL guide . . . 623

4.27 REST API Version History . . . 628
4.28 Conductor as a place for orchestrating tasks . . . 644
4.29 Guru Meditation Reports . . . 645
4.30 Internationalization . . . 646
4.31 Live Migration . . . 647
4.32 Notifications in Nova . . . 647
4.33 REST API Policy Enforcement . . . 656
4.34 AMQP and Nova . . . 658
4.35 Scheduling . . . 663
4.36 Scheduler Evolution . . . 666
4.37 Services, Managers and Drivers . . . 667
4.38 Nova Stable REST API . . . 670
4.39 Threading model . . . 673
4.40 ComputeDriver.update_provider_tree . . . 673
4.41 Upgrade checks . . . 677
4.42 Virtual Machine States and Transitions . . . 681
4.43 Scheduler hints versus flavor extra specs . . . 683
4.44 Filtering hosts by isolating aggregates . . . 685
4.45 User Documentation . . . 686
4.46 Host Aggregates . . . 721
4.47 Nova System Architecture . . . 725
4.48 Block Device Mapping in Nova . . . 726
4.49 Cells . . . 730
4.50 Cells Layout (v2) . . . 740
4.51 Feature Classification . . . 745
4.52 Filter Scheduler . . . 759
4.53 Flavors . . . 767
4.54 Manage IP addresses . . . 778
4.55 Quotas . . . 781
4.56 Feature Support Matrix . . . 782
4.57 Upgrades . . . 807
4.58 Using WSGI with Nova . . . 811

5 Search **813**

CHAPTER

ONE

WHAT IS NOVA?

Nova is the OpenStack project that provides a way to provision compute instances (aka virtual servers). Nova supports creating virtual machines, baremetal servers (through the use of ironic), and has limited support for system containers. Nova runs as a set of daemons on top of existing Linux servers to provide that service.

It requires the following additional OpenStack services for basic function:

- Keystone: This provides identity and authentication for all OpenStack services.
- Glance: This provides the compute image repository. All compute instances launch from glance images.
- Neutron: This is responsible for provisioning the virtual or physical networks that compute instances connect to on boot.
- Placement: This is responsible for tracking inventory of resources available in a cloud and assisting in choosing which provider of those resources will be used when creating a virtual machine.

It can also integrate with other services to include: persistent block storage, encrypted disks, and baremetal compute instances.

CHAPTER

TWO

FOR END USERS

As an end user of nova, youll use nova to create and manage servers with either tools or the API directly.

2.1 Tools for using Nova

- Horizon: The official web UI for the OpenStack Project.
- OpenStack Client: The official CLI for OpenStack Projects. You should use this as your CLI for most things, it includes not just nova commands but also commands for most of the projects in OpenStack.
- Nova Client: For some very advanced features (or administrative commands) of nova you may need to use nova client. It is still supported, but the `openstack` cli is recommended.

2.2 Writing to the API

All end user (and some administrative) features of nova are exposed via a REST API, which can be used to build more complicated logic or automation with nova. This can be consumed directly, or via various SDKs. The following resources will help you get started with consuming the API directly.

- Compute API Guide: The concept guide for the API. This helps lay out the concepts behind the API to make consuming the API reference easier.
- Compute API Reference: The complete reference for the compute API, including all methods and request / response parameters and their meaning.
- *Compute API Microversion History*: The compute API evolves over time through Microversions. This provides the history of all those changes. Consider it a whats new in the compute API.
- *Block Device Mapping*: One of the trickier parts to understand is the Block Device Mapping parameters used to connect specific block devices to computes. This deserves its own deep dive.
- *Metadata*: Provide information to the guest instance when it is created.

Nova can be configured to emit notifications over RPC.

- *Versioned Notifications*: This provides the list of existing versioned notifications with sample payloads.

CHAPTER

THREE

FOR OPERATORS

3.1 Architecture Overview

- *Nova architecture*: An overview of how all the parts in nova fit together.

3.2 Installation

The detailed install guide for nova. A functioning nova will also require having installed keystone, glance, neutron, and placement. Ensure that you follow their install guides first.

3.2.1 Compute service

Overview

The OpenStack project is an open source cloud computing platform that supports all types of cloud environments. The project aims for simple implementation, massive scalability, and a rich set of features. Cloud computing experts from around the world contribute to the project.

OpenStack provides an Infrastructure-as-a-Service (IaaS) solution through a variety of complementary services. Each service offers an Application Programming Interface (API) that facilitates this integration.

This guide covers step-by-step deployment of the major OpenStack services using a functional example architecture suitable for new users of OpenStack with sufficient Linux experience. This guide is not intended to be used for production system installations, but to create a minimum proof-of-concept for the purpose of learning about OpenStack.

After becoming familiar with basic installation, configuration, operation, and troubleshooting of these OpenStack services, you should consider the following steps toward deployment using a production architecture:

- Determine and implement the necessary core and optional services to meet performance and redundancy requirements.
- Increase security using methods such as firewalls, encryption, and service policies.
- Implement a deployment tool such as Ansible, Chef, Puppet, or Salt to automate deployment and management of the production environment.

Example architecture

The example architecture requires at least two nodes (hosts) to launch a basic virtual machine (VM) or instance. Optional services such as Block Storage and Object Storage require additional nodes.

Important: The example architecture used in this guide is a minimum configuration, and is not intended for production system installations. It is designed to provide a minimum proof-of-concept for the purpose of learning about OpenStack. For information on creating architectures for specific use cases, or how to determine which architecture is required, see the Architecture Design Guide.

This example architecture differs from a minimal production architecture as follows:

- Networking agents reside on the controller node instead of one or more dedicated network nodes.
- Overlay (tunnel) traffic for self-service networks traverses the management network instead of a dedicated network.

For more information on production architectures, see the Architecture Design Guide, OpenStack Operations Guide, and OpenStack Networking Guide.

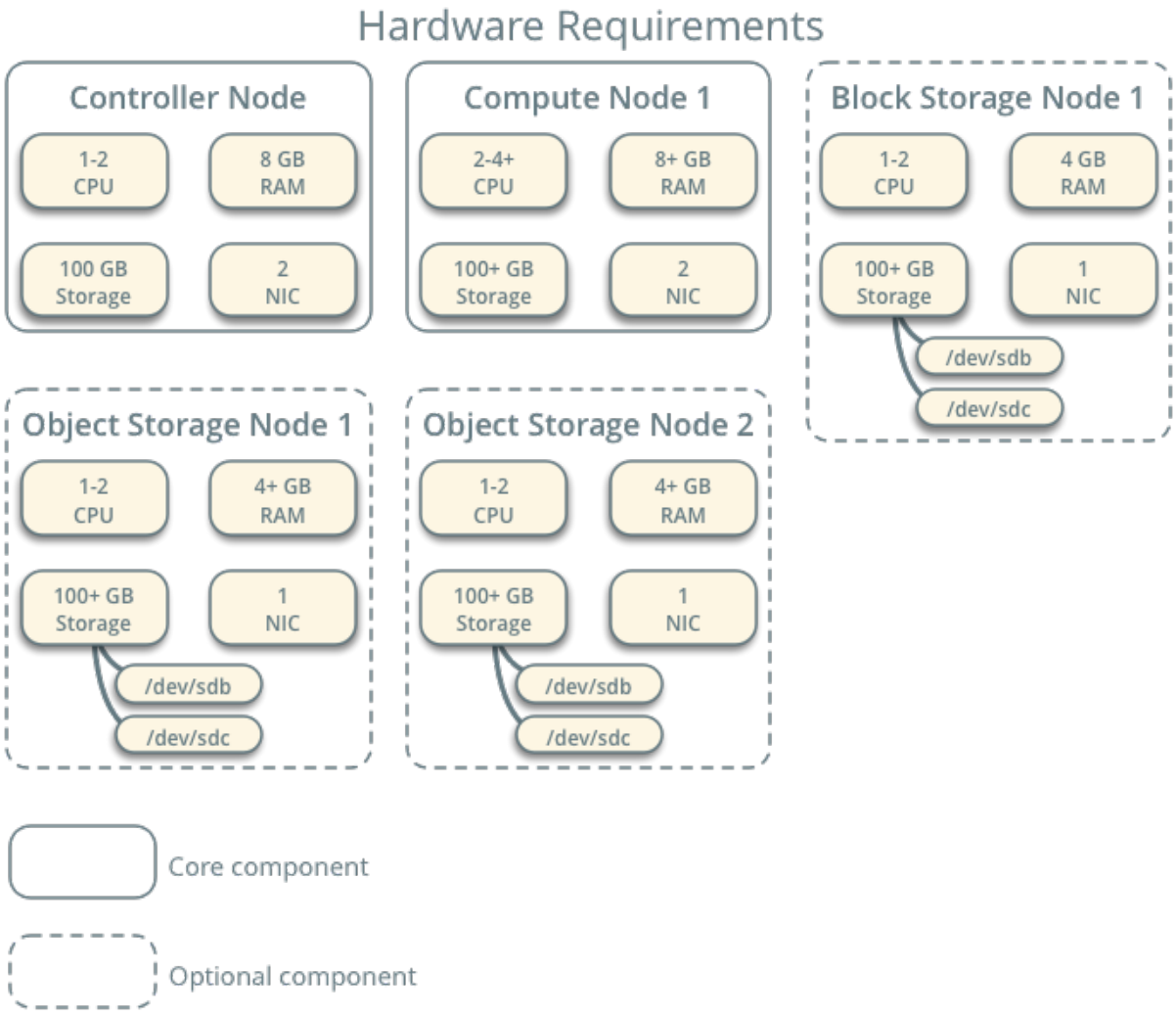


Fig. 1: **Hardware requirements**

Controller

The controller node runs the Identity service, Image service, management portions of Compute, management portion of Networking, various Networking agents, and the Dashboard. It also includes supporting services such as an SQL database, message queue, and Network Time Protocol (NTP).

Optionally, the controller node runs portions of the Block Storage, Object Storage, Orchestration, and Telemetry services.

The controller node requires a minimum of two network interfaces.

Compute

The compute node runs the hypervisor portion of Compute that operates instances. By default, Compute uses the kernel-based VM (KVM) hypervisor. The compute node also runs a Networking service agent that connects instances to virtual networks and provides firewalling services to instances via security groups.

You can deploy more than one compute node. Each node requires a minimum of two network interfaces.

Block Storage

The optional Block Storage node contains the disks that the Block Storage and Shared File System services provision for instances.

For simplicity, service traffic between compute nodes and this node uses the management network. Production environments should implement a separate storage network to increase performance and security.

You can deploy more than one block storage node. Each node requires a minimum of one network interface.

Object Storage

The optional Object Storage node contain the disks that the Object Storage service uses for storing accounts, containers, and objects.

For simplicity, service traffic between compute nodes and this node uses the management network. Production environments should implement a separate storage network to increase performance and security.

This service requires two nodes. Each node requires a minimum of one network interface. You can deploy more than two object storage nodes.

Networking

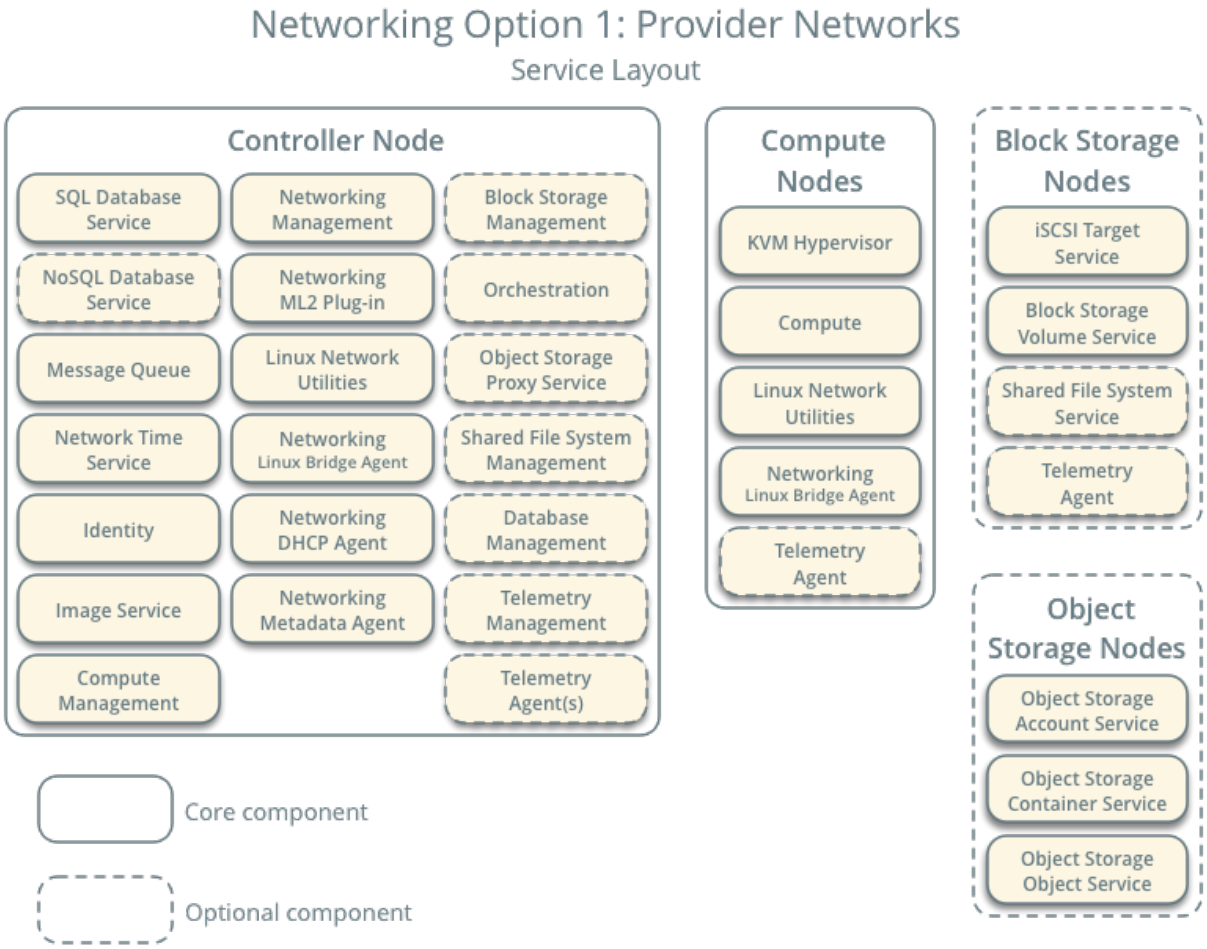
Choose one of the following virtual networking options.

Networking Option 1: Provider networks

The provider networks option deploys the OpenStack Networking service in the simplest way possible with primarily layer-2 (bridging/switching) services and VLAN segmentation of networks. Essentially, it bridges virtual networks to physical networks and relies on physical network infrastructure for layer-3 (routing) services. Additionally, a DHCP<Dynamic Host Configuration Protocol (DHCP) service provides IP address information to instances.

The OpenStack user requires more information about the underlying network infrastructure to create a virtual network to exactly match the infrastructure.

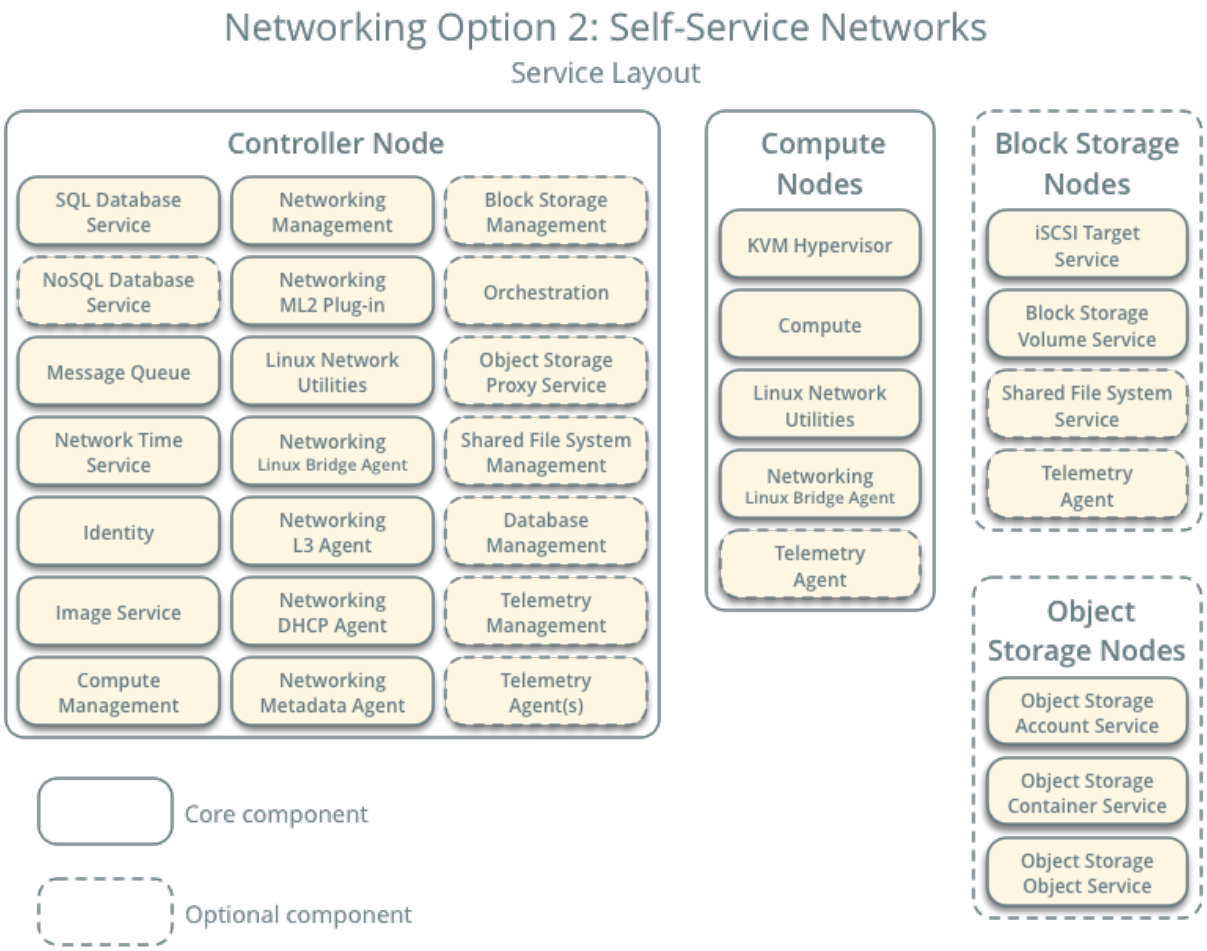
Warning: This option lacks support for self-service (private) networks, layer-3 (routing) services, and advanced services such as Load-Balancer-as-a-Service (LBaaS) and FireWall-as-a-Service (FWaaS). Consider the self-service networks option below if you desire these features.



Networking Option 2: Self-service networks

The self-service networks option augments the provider networks option with layer-3 (routing) services that enable self-service networks using overlay segmentation methods such as Virtual Extensible LAN (VXLAN). Essentially, it routes virtual networks to physical networks using Network Address Translation (NAT). Additionally, this option provides the foundation for advanced services such as LBaaS and FWaaS.

The OpenStack user can create virtual networks without the knowledge of underlying infrastructure on the data network. This can also include VLAN networks if the layer-2 plug-in is configured accordingly.


Networking Option 2: Self-Service Networks
Service Layout
Controller Node
SQL Database Service
Networking Management
Block Storage Management
NoSQL Database Service
Networking ML2 Plug-in
Orchestration
Message Queue
Linux Network Utilities
Object Storage Proxy Service
Network Time Service
Networking Linux Bridge Agent
Shared File System Management
Identity
Networking L3 Agent
Database Management
Image Service
Networking DHCP Agent
Telemetry Management
Compute Management
Networking Metadata Agent
Telemetry Agent(s)
Compute Nodes
KVM Hypervisor
Compute
Linux Network Utilities
Networking Linux Bridge Agent
Telemetry Agent
Block Storage Nodes
iSCSI Target Service
Block Storage Volume Service
Shared File System Service
Telemetry Agent
Object Storage Nodes
Object Storage Account Service
Object Storage Container Service
Object Storage Object Service
Core component
Optional component

Compute service overview

Todo: Update a lot of the links in here.

Use OpenStack Compute to host and manage cloud computing systems. OpenStack Compute is a major part of an Infrastructure-as-a-Service (IaaS) system. The main modules are implemented in Python.

OpenStack Compute interacts with OpenStack Identity for authentication, OpenStack Placement for resource inventory tracking and selection, OpenStack Image service for disk and server images, and OpenStack Dashboard for the user and administrative interface. Image access is limited by projects, and by users; quotas are limited per project (the number of instances, for example). OpenStack Compute can scale horizontally on standard hardware, and download images to launch instances.

OpenStack Compute consists of the following areas and their components:

`nova-api` service Accepts and responds to end user compute API calls. The service supports the OpenStack Compute API. It enforces some policies and initiates most orchestration activities, such as running an instance.

`nova-api-metadata` service Accepts metadata requests from instances. The `nova-api-metadata` service is generally used when you run in multi-host mode with `nova-network` installations. For details, see *Metadata service* in the Compute Administrator Guide.

`nova-compute` service A worker daemon that creates and terminates virtual machine instances through hypervisor APIs. For example:

- XenAPI for XenServer/XCP
- libvirt for KVM or QEMU
- VMwareAPI for VMware

Processing is fairly complex. Basically, the daemon accepts actions from the queue and performs a series of system commands such as launching a KVM instance and updating its state in the database.

`nova-scheduler` service Takes a virtual machine instance request from the queue and determines on which compute server host it runs.

`nova-conductor` module Mediates interactions between the `nova-compute` service and the database. It eliminates direct accesses to the cloud database made by the `nova-compute` service. The `nova-conductor` module scales horizontally. However, do not deploy it on nodes where the `nova-compute` service runs. For more information, see the `conductor` section in the *Configuration Options*.

`nova-novncproxy` daemon Provides a proxy for accessing running instances through a VNC connection. Supports browser-based novnc clients.

`nova-spicehtml5proxy` daemon Provides a proxy for accessing running instances through a SPICE connection. Supports browser-based HTML5 client.

`nova-xvpvncproxy` daemon Provides a proxy for accessing running instances through a VNC connection. Supports an OpenStack-specific Java client.

Deprecated since version 19.0.0: **`nova-xvpvnxproxy`** is deprecated since 19.0.0 (Stein) and will be removed in an upcoming release.

The queue A central hub for passing messages between daemons. Usually implemented with RabbitMQ but other options are available.

SQL database Stores most build-time and run-time states for a cloud infrastructure, including:

- Available instance types
- Instances in use

- Available networks
- Projects

Theoretically, OpenStack Compute can support any database that SQLAlchemy supports. Common databases are SQLite3 for test and development work, MySQL, MariaDB, and PostgreSQL.

Install and configure controller node

This section describes how to install and configure the Compute service on the controller node for Ubuntu, openSUSE and SUSE Linux Enterprise, and Red Hat Enterprise Linux and CentOS.

Install and configure controller node for Ubuntu

This section describes how to install and configure the Compute service, code-named nova, on the controller node.

Prerequisites

Before you install and configure the Compute service, you must create databases, service credentials, and API endpoints.

1. To create the databases, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:

     ```
     # mysql
     ```

   - Create the `nova_api`, `nova`, and `nova_cell0` databases:

     ```
     MariaDB [(none)]> CREATE DATABASE nova_api;
     MariaDB [(none)]> CREATE DATABASE nova;
     MariaDB [(none)]> CREATE DATABASE nova_cell0;
     ```

   - Grant proper access to the databases:

     ```
     MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
       IDENTIFIED BY 'NOVA_DBPASS';
     MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
       IDENTIFIED BY 'NOVA_DBPASS';

     MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
       IDENTIFIED BY 'NOVA_DBPASS';
     MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
       IDENTIFIED BY 'NOVA_DBPASS';

     MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' \
       IDENTIFIED BY 'NOVA_DBPASS';
     MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%' \
       IDENTIFIED BY 'NOVA_DBPASS';
     ```

     Replace `NOVA_DBPASS` with a suitable password.

   - Exit the database access client.
2. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

3. Create the Compute service credentials:

   - Create the `nova` user:

```
$ openstack user create --domain default --password-prompt nova

User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 8a7dbf5279404537b1c7b86c033620fe |
| name                | nova                             |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

   - Add the `admin` role to the `nova` user:

```
$ openstack role add --project service --user nova admin
```

   Note: This command provides no output.

   - Create the `nova` service entity:

```
$ openstack service create --name nova \
  --description "OpenStack Compute" compute

+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Compute                |
| enabled     | True                             |
| id          | 060d59eac51b4594815603d75a00aba2 |
| name        | nova                             |
| type        | compute                          |
+-------------+----------------------------------+
```

4. Create the Compute API service endpoints:

```
$ openstack endpoint create --region RegionOne \
  compute public http://controller:8774/v2.1

+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 3c1caa473bfe4390a11e7177894bcc7b         |
| interface    | public                                   |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 060d59eac51b4594815603d75a00aba2         |
```

(continues on next page)

(continued from previous page)

```
| service_name | nova                                     |
| service_type | compute                                  |
| url          | http://controller:8774/v2.1              |
+--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute internal http://controller:8774/v2.1

+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | e3c918de680746a586eac1f2d9bc10ab         |
| interface    | internal                                 |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 060d59eac51b4594815603d75a00aba2         |
| service_name | nova                                     |
| service_type | compute                                  |
| url          | http://controller:8774/v2.1              |
+--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute admin http://controller:8774/v2.1

+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 38f7af91666a47cfb97b4dc790b94424         |
| interface    | admin                                    |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 060d59eac51b4594815603d75a00aba2         |
| service_name | nova                                     |
| service_type | compute                                  |
| url          | http://controller:8774/v2.1              |
+--------------+------------------------------------------+
```

Install and configure components

Note: Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# apt install nova-api nova-conductor nova-novncproxy nova-scheduler
```

2. Edit the `/etc/nova/nova.conf` file and complete the following actions:

   - In the `[api_database]` and `[database]` sections, configure database access:

```
[api_database]
# ...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api

[database]
# ...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova
```

Replace `NOVA_DBPASS` with the password you chose for the Compute databases.

- In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672/
```

Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

- In the `[api]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[api]
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS
```

Replace `NOVA_PASS` with the password you chose for the `nova` user in the Identity service.

Note: Comment out or remove any other options in the `[keystone_authtoken]` section.

- In the `[DEFAULT]` section, configure the `my_ip` option to use the management interface IP address of the controller node:

```
[DEFAULT]
# ...
my_ip = 10.0.0.11
```

- In the `[DEFAULT]` section, enable support for the Networking service:

```
[DEFAULT]
# ...
use_neutron = true
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

Note: By default, Compute uses an internal firewall driver. Since the Networking service includes a

firewall driver, you must disable the Compute firewall driver by using the `nova.virt.firewall.NoopFirewallDriver` firewall driver.

- Configure the `[neutron]` section of **/etc/nova/nova.conf**. Refer to the Networking service install guide for more information.
- In the `[vnc]` section, configure the VNC proxy to use the management interface IP address of the controller node:

```
[vnc]
enabled = true
# ...
server_listen = $my_ip
server_proxyclient_address = $my_ip
```

- In the `[glance]` section, configure the location of the Image service API:

```
[glance]
# ...
api_servers = http://controller:9292
```

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
# ...
lock_path = /var/lib/nova/tmp
```

- Due to a packaging bug, remove the `log_dir` option from the `[DEFAULT]` section.
- In the `[placement]` section, configure access to the Placement service:

```
[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS
```

Replace `PLACEMENT_PASS` with the password you choose for the `placement` service user created when installing Placement. Comment out or remove any other options in the `[placement]` section.

3. Populate the `nova-api` database:

```
# su -s /bin/sh -c "nova-manage api_db sync" nova
```

Note: Ignore any deprecation messages in this output.

4. Register the `cell0` database:

```
# su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova
```

5. Create the `cell1` cell:

```
# su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose" nova
```

6. Populate the nova database:

```
# su -s /bin/sh -c "nova-manage db sync" nova
```

7. Verify nova cell0 and cell1 are registered correctly:

```
# su -s /bin/sh -c "nova-manage cell_v2 list_cells" nova
+-------+--------------------------------------+------------------------------------
↪------------------+-------------------------------------------------------------
↪-+----------+
|  Name |                 UUID                 |                   Transport URL ␣
↪                  |                      Database Connection                      ␣
↪ | Disabled |
+-------+--------------------------------------+------------------------------------
↪------------------+-------------------------------------------------------------
↪-+----------+
| cell0 | 00000000-0000-0000-0000-000000000000 |                       none:/     ␣
↪                  | mysql+pymysql://nova:****@controller/nova_cell0?
↪charset=utf8 |  False   |
| cell1 | f690f4fd-2bc5-4f15-8145-db561a7b9d3d | rabbit://
↪openstack:****@controller:5672/nova_cell1 | mysql+pymysql://
↪nova:****@controller/nova_cell1?charset=utf8 |  False   |
+-------+--------------------------------------+------------------------------------
↪------------------+-------------------------------------------------------------
↪-+----------+
```

Finalize installation

- Restart the Compute services:

```
# service nova-api restart
# service nova-scheduler restart
# service nova-conductor restart
# service nova-novncproxy restart
```

Install and configure controller node for openSUSE and SUSE Linux Enterprise

This section describes how to install and configure the Compute service, code-named nova, on the controller node.

Prerequisites

Before you install and configure the Compute service, you must create databases, service credentials, and API endpoints.

1. To create the databases, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:

     ```
     $ mysql -u root -p
     ```

   - Create the `nova_api`, `nova`, and `nova_cell0` databases:

```
MariaDB [(none)]> CREATE DATABASE nova_api;
MariaDB [(none)]> CREATE DATABASE nova;
MariaDB [(none)]> CREATE DATABASE nova_cell0;
```

- Grant proper access to the databases:

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
```

  Replace `NOVA_DBPASS` with a suitable password.

- Exit the database access client.

2. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

3. Create the Compute service credentials:

   - Create the `nova` user:

```
$ openstack user create --domain default --password-prompt nova

User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 8a7dbf5279404537b1c7b86c033620fe |
| name                | nova                             |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

   - Add the `admin` role to the `nova` user:

```
$ openstack role add --project service --user nova admin
```

Note: This command provides no output.

   - Create the `nova` service entity:

```
$ openstack service create --name nova \
  --description "OpenStack Compute" compute

+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Compute                |
| enabled     | True                             |
| id          | 060d59eac51b4594815603d75a00aba2 |
| name        | nova                             |
| type        | compute                          |
+-------------+----------------------------------+
```

4. Create the Compute API service endpoints:

```
$ openstack endpoint create --region RegionOne \
  compute public http://controller:8774/v2.1

+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 3c1caa473bfe4390a11e7177894bcc7b         |
| interface    | public                                   |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 060d59eac51b4594815603d75a00aba2         |
| service_name | nova                                     |
| service_type | compute                                  |
| url          | http://controller:8774/v2.1              |
+--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute internal http://controller:8774/v2.1

+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | e3c918de680746a586eac1f2d9bc10ab         |
| interface    | internal                                 |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 060d59eac51b4594815603d75a00aba2         |
| service_name | nova                                     |
| service_type | compute                                  |
| url          | http://controller:8774/v2.1              |
+--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute admin http://controller:8774/v2.1

+--------------+------------------------------------------+
| Field        | Value                                    |
+--------------+------------------------------------------+
| enabled      | True                                     |
| id           | 38f7af91666a47cfb97b4dc790b94424         |
```

(continues on next page)

(continued from previous page)

```
| interface    | admin                                    |
| region       | RegionOne                                |
| region_id    | RegionOne                                |
| service_id   | 060d59eac51b4594815603d75a00aba2         |
| service_name | nova                                     |
| service_type | compute                                  |
| url          | http://controller:8774/v2.1              |
+--------------+------------------------------------------+
```

Install and configure components

Note: Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

Note: As of the Newton release, SUSE OpenStack packages are shipped with the upstream default configuration files. For example, `/etc/nova/nova.conf` has customizations in `/etc/nova/nova.conf.d/010-nova.conf`. While the following instructions modify the default configuration file, adding a new file in `/etc/nova/nova.conf.d` achieves the same result.

1. Install the packages:

```
# zypper install \
    openstack-nova-api \
    openstack-nova-scheduler \
    openstack-nova-conductor \
    openstack-nova-novncproxy \
    iptables
```

2. Edit the `/etc/nova/nova.conf` file and complete the following actions:

   - In the `[DEFAULT]` section, enable only the compute and metadata APIs:

```
[DEFAULT]
# ...
enabled_apis = osapi_compute,metadata
```

   - In the `[api_database]` and `[database]` sections, configure database access:

```
[api_database]
# ...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api

[database]
# ...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova
```

     Replace `NOVA_DBPASS` with the password you chose for the Compute databases.

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672/
```

Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

- In the `[api]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[api]
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS
```

Replace `NOVA_PASS` with the password you chose for the `nova` user in the Identity service.

Note: Comment out or remove any other options in the `[keystone_authtoken]` section.

- In the `[DEFAULT]` section, configure the `my_ip` option to use the management interface IP address of the controller node:

```
[DEFAULT]
# ...
my_ip = 10.0.0.11
```

- In the `[DEFAULT]` section, enable support for the Networking service:

```
[DEFAULT]
# ...
use_neutron = true
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

Note: By default, Compute uses an internal firewall driver. Since the Networking service includes a firewall driver, you must disable the Compute firewall driver by using the `nova.virt.firewall.NoopFirewallDriver` firewall driver.

- Configure the `[neutron]` section of **/etc/nova/nova.conf**. Refer to the Networking service install guide for more details.
- In the `[vnc]` section, configure the VNC proxy to use the management interface IP address of the controller node:

```
[vnc]
enabled = true
```

(continues on next page)

(continued from previous page)

```
# ...
server_listen = $my_ip
server_proxyclient_address = $my_ip
```

- In the `[glance]` section, configure the location of the Image service API:

```
[glance]
# ...
api_servers = http://controller:9292
```

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
# ...
lock_path = /var/run/nova
```

- In the `[placement]` section, configure access to the Placement service:

```
[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS
```

Replace `PLACEMENT_PASS` with the password you choose for the `placement` service user created when installing Placement. Comment out or remove any other options in the `[placement]` section.

3. Populate the `nova-api` database:

```
# su -s /bin/sh -c "nova-manage api_db sync" nova
```

Note: Ignore any deprecation messages in this output.

4. Register the `cell0` database:

```
# su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova
```

5. Create the `cell1` cell:

```
# su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose" nova
```

6. Populate the nova database:

```
# su -s /bin/sh -c "nova-manage db sync" nova
```

7. Verify nova cell0 and cell1 are registered correctly:

```
# su -s /bin/sh -c "nova-manage cell_v2 list_cells" nova
+-------+--------------------------------------+----------------------------------
 ↪------------------+-------------------------------------------------------------
 ↪-+-----------+
```

(continues on next page)

(continued from previous page)

```
|  Name |                 UUID                 |                   Transport URL ␣
↪                  |                     Database Connection                    ␣
↪ | Disabled |
+-------+--------------------------------------+----------------------------------
↪------------------+-------------------------------------------------------------
↪-+----------+
| cell0 | 00000000-0000-0000-0000-000000000000 |                       none:/    ␣
↪                  | mysql+pymysql://nova:****@controller/nova_cell0?
↪charset=utf8 |  False   |
| cell1 | f690f4fd-2bc5-4f15-8145-db561a7b9d3d | rabbit://
↪openstack:****@controller:5672/nova_cell1 | mysql+pymysql://
↪nova:****@controller/nova_cell1?charset=utf8 |  False   |
+-------+--------------------------------------+----------------------------------
↪------------------+-------------------------------------------------------------
↪-+----------+
```

Finalize installation

- Start the Compute services and configure them to start when the system boots:

```
# systemctl enable \
    openstack-nova-api.service \
    openstack-nova-scheduler.service \
    openstack-nova-conductor.service \
    openstack-nova-novncproxy.service
# systemctl start \
    openstack-nova-api.service \
    openstack-nova-scheduler.service \
    openstack-nova-conductor.service \
    openstack-nova-novncproxy.service
```

Install and configure controller node for Red Hat Enterprise Linux and CentOS

This section describes how to install and configure the Compute service, code-named nova, on the controller node.

Prerequisites

Before you install and configure the Compute service, you must create databases, service credentials, and API endpoints.

1. To create the databases, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:

     ```
     $ mysql -u root -p
     ```

   - Create the `nova_api`, `nova`, and `nova_cell0` databases:

     ```
     MariaDB [(none)]> CREATE DATABASE nova_api;
     MariaDB [(none)]> CREATE DATABASE nova;
     MariaDB [(none)]> CREATE DATABASE nova_cell0;
     ```

   - Grant proper access to the databases:

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
```

Replace `NOVA_DBPASS` with a suitable password.

- Exit the database access client.

2. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

3. Create the Compute service credentials:

- Create the `nova` user:

```
$ openstack user create --domain default --password-prompt nova

User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 8a7dbf5279404537b1c7b86c033620fe |
| name                | nova                             |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

- Add the `admin` role to the `nova` user:

```
$ openstack role add --project service --user nova admin
```

Note: This command provides no output.

- Create the `nova` service entity:

```
$ openstack service create --name nova \
  --description "OpenStack Compute" compute

+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
```

(continues on next page)

(continued from previous page)

```
| description | OpenStack Compute                |
| enabled     | True                             |
| id          | 060d59eac51b4594815603d75a00aba2 |
| name        | nova                             |
| type        | compute                          |
+-------------+----------------------------------+
```

4. Create the Compute API service endpoints:

```
$ openstack endpoint create --region RegionOne \
  compute public http://controller:8774/v2.1

+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 3c1caa473bfe4390a11e7177894bcc7b        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 060d59eac51b4594815603d75a00aba2        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1             |
+--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute internal http://controller:8774/v2.1

+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | e3c918de680746a586eac1f2d9bc10ab        |
| interface    | internal                                |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 060d59eac51b4594815603d75a00aba2        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1             |
+--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  compute admin http://controller:8774/v2.1

+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 38f7af91666a47cfb97b4dc790b94424        |
| interface    | admin                                   |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | 060d59eac51b4594815603d75a00aba2        |
| service_name | nova                                    |
| service_type | compute                                 |
```

(continues on next page)

(continued from previous page)

```
| url          | http://controller:8774/v2.1                |
+--------------+--------------------------------------------+
```

Install and configure components

Note: Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# yum install openstack-nova-api openstack-nova-conductor \
  openstack-nova-novncproxy openstack-nova-scheduler
```

2. Edit the `/etc/nova/nova.conf` file and complete the following actions:

   - In the `[DEFAULT]` section, enable only the compute and metadata APIs:

```
[DEFAULT]
# ...
enabled_apis = osapi_compute,metadata
```

   - In the `[api_database]` and `[database]` sections, configure database access:

```
[api_database]
# ...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api

[database]
# ...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova
```

     Replace `NOVA_DBPASS` with the password you chose for the Compute databases.

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672/
```

     Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

   - In the `[api]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[api]
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
```

(continues on next page)

(continued from previous page)

```
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS
```

Replace `NOVA_PASS` with the password you chose for the `nova` user in the Identity service.

Note: Comment out or remove any other options in the `[keystone_authtoken]` section.

- In the `[DEFAULT]` section, configure the `my_ip` option to use the management interface IP address of the controller node:

```
[DEFAULT]
# ...
my_ip = 10.0.0.11
```

- In the `[DEFAULT]` section, enable support for the Networking service:

```
[DEFAULT]
# ...
use_neutron = true
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

Note: By default, Compute uses an internal firewall driver. Since the Networking service includes a firewall driver, you must disable the Compute firewall driver by using the `nova.virt.firewall.NoopFirewallDriver` firewall driver.

- Configure the `[neutron]` section of **/etc/nova/nova.conf**. Refer to the Networking service install guide for more details.
- In the `[vnc]` section, configure the VNC proxy to use the management interface IP address of the controller node:

```
[vnc]
enabled = true
# ...
server_listen = $my_ip
server_proxyclient_address = $my_ip
```

- In the `[glance]` section, configure the location of the Image service API:

```
[glance]
# ...
api_servers = http://controller:9292
```

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
# ...
lock_path = /var/lib/nova/tmp
```

- In the `[placement]` section, configure access to the Placement service:

```
[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS
```

Replace `PLACEMENT_PASS` with the password you choose for the `placement` service user created when installing Placement. Comment out or remove any other options in the `[placement]` section.

3. Populate the `nova-api` database:

```
# su -s /bin/sh -c "nova-manage api_db sync" nova
```

Note: Ignore any deprecation messages in this output.

4. Register the `cell0` database:

```
# su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova
```

5. Create the `cell1` cell:

```
# su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose" nova
```

6. Populate the nova database:

```
# su -s /bin/sh -c "nova-manage db sync" nova
```

7. Verify nova cell0 and cell1 are registered correctly:

```
# su -s /bin/sh -c "nova-manage cell_v2 list_cells" nova
+-------+--------------------------------------+----------------------------------
↪------------------+--------------------------------------------------------------
↪-+----------+
|  Name |                 UUID                 |                   Transport URL
↪                  |                      Database Connection                     
↪ | Disabled |
+-------+--------------------------------------+----------------------------------
↪------------------+--------------------------------------------------------------
↪-+----------+
| cell0 | 00000000-0000-0000-0000-000000000000 |                       none:/
↪                  | mysql+pymysql://nova:****@controller/nova_cell0?
↪charset=utf8 |  False   |
| cell1 | f690f4fd-2bc5-4f15-8145-db561a7b9d3d | rabbit://
↪openstack:****@controller:5672/nova_cell1 | mysql+pymysql://
↪nova:****@controller/nova_cell1?charset=utf8 |  False   |
+-------+--------------------------------------+----------------------------------
↪------------------+--------------------------------------------------------------
↪-+----------+
```

Finalize installation

- Start the Compute services and configure them to start when the system boots:

```
# systemctl enable \
    openstack-nova-api.service \
    openstack-nova-scheduler.service \
    openstack-nova-conductor.service \
    openstack-nova-novncproxy.service
# systemctl start \
    openstack-nova-api.service \
    openstack-nova-scheduler.service \
    openstack-nova-conductor.service \
    openstack-nova-novncproxy.service
```

Install and configure a compute node

This section describes how to install and configure the Compute service on a compute node for Ubuntu, openSUSE and SUSE Linux Enterprise, and Red Hat Enterprise Linux and CentOS.

The service supports several hypervisors to deploy instances or virtual machines (VMs). For simplicity, this configuration uses the Quick EMUlator (QEMU) hypervisor with the kernel-based VM (KVM) extension on compute nodes that support hardware acceleration for virtual machines. On legacy hardware, this configuration uses the generic QEMU hypervisor. You can follow these instructions with minor modifications to horizontally scale your environment with additional compute nodes.

Note: This section assumes that you are following the instructions in this guide step-by-step to configure the first compute node. If you want to configure additional compute nodes, prepare them in a similar fashion to the first compute node in the *example architectures* section. Each additional compute node requires a unique IP address.

Install and configure a compute node for Ubuntu

This section describes how to install and configure the Compute service on a compute node. The service supports several hypervisors to deploy instances or virtual machines (VMs). For simplicity, this configuration uses the Quick EMUlator (QEMU) hypervisor with the kernel-based VM (KVM) extension on compute nodes that support hardware acceleration for virtual machines. On legacy hardware, this configuration uses the generic QEMU hypervisor. You can follow these instructions with minor modifications to horizontally scale your environment with additional compute nodes.

Note: This section assumes that you are following the instructions in this guide step-by-step to configure the first compute node. If you want to configure additional compute nodes, prepare them in a similar fashion to the first compute node in the *example architectures* section. Each additional compute node requires a unique IP address.

Install and configure components

Note: Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential

default configuration options that you should retain.

1. Install the packages:

```
# apt install nova-compute
```

2. Edit the `/etc/nova/nova.conf` file and complete the following actions:

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

     Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

   - In the `[api]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[api]
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS
```

     Replace `NOVA_PASS` with the password you chose for the `nova` user in the Identity service.

     Note: Comment out or remove any other options in the `[keystone_authtoken]` section.

   - In the `[DEFAULT]` section, configure the `my_ip` option:

```
[DEFAULT]
# ...
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

     Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network interface on your compute node, typically 10.0.0.31 for the first node in the *example architecture*.

   - In the `[DEFAULT]` section, enable support for the Networking service:

```
[DEFAULT]
# ...
use_neutron = true
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

     Note: By default, Compute uses an internal firewall service. Since Networking includes a firewall service, you must disable the Compute firewall service by using the `nova.virt.firewall.`

`NoopFirewallDriver` firewall driver.

- Configure the `[neutron]` section of **/etc/nova/nova.conf**. Refer to the Networking service install guide for more details.
- In the `[vnc]` section, enable and configure remote console access:

```
[vnc]
# ...
enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html
```

The server component listens on all IP addresses and the proxy component only listens on the management interface IP address of the compute node. The base URL indicates the location where you can use a web browser to access remote consoles of instances on this compute node.

Note: If the web browser to access remote consoles resides on a host that cannot resolve the `controller` hostname, you must replace `controller` with the management interface IP address of the controller node.

- In the `[glance]` section, configure the location of the Image service API:

```
[glance]
# ...
api_servers = http://controller:9292
```

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
# ...
lock_path = /var/lib/nova/tmp
```

- In the `[placement]` section, configure the Placement API:

```
[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS
```

Replace `PLACEMENT_PASS` with the password you choose for the `placement` user in the Identity service. Comment out any other options in the `[placement]` section.

Finalize installation

1. Determine whether your compute node supports hardware acceleration for virtual machines:

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

If this command returns a value of `one or greater`, your compute node supports hardware acceleration which typically requires no additional configuration.

If this command returns a value of `zero`, your compute node does not support hardware acceleration and you must configure `libvirt` to use QEMU instead of KVM.

- Edit the `[libvirt]` section in the `/etc/nova/nova-compute.conf` file as follows:

```
[libvirt]
# ...
virt_type = qemu
```

2. Restart the Compute service:

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

Note: If the `nova-compute` service fails to start, check `/var/log/nova/nova-compute.log`. The error message `AMQP server on controller:5672 is unreachable` likely indicates that the firewall on the controller node is preventing access to port 5672. Configure the firewall to open port 5672 on the controller node and restart `nova-compute` service on the compute node.

Add the compute node to the cell database

Important: Run the following commands on the **controller** node.

1. Source the admin credentials to enable admin-only CLI commands, then confirm there are compute hosts in the database:

```
$ . admin-openrc

$ openstack compute service list --service nova-compute
+----+-------+--------------+------+-------+---------+----------------------------
↪+
| ID | Host  | Binary       | Zone | State | Status  | Updated At                 ␣
↪|
+----+-------+--------------+------+-------+---------+----------------------------
↪+
| 1  | node1 | nova-compute | nova | up    | enabled | 2017-04-14T15:30:44.000000␣
↪|
+----+-------+--------------+------+-------+---------+----------------------------
↪+
```

2. Discover compute hosts:

```
# su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova

Found 2 cell mappings.
Skipping cell0 since it does not contain hosts.
Getting compute nodes from cell 'cell1': ad5a5985-a719-4567-98d8-8d148aaae4bc
Found 1 computes in cell: ad5a5985-a719-4567-98d8-8d148aaae4bc
```

(continues on next page)

(continued from previous page)

```
Checking host mapping for compute host 'compute': fe58ddc1-1d65-4f87-9456-
↪bc040dc106b3
Creating host mapping for compute host 'compute': fe58ddc1-1d65-4f87-9456-
↪bc040dc106b3
```

Note: When you add new compute nodes, you must run `nova-manage cell_v2 discover_hosts` on the controller node to register those new compute nodes. Alternatively, you can set an appropriate interval in `/etc/nova/nova.conf`:

```
[scheduler]
discover_hosts_in_cells_interval = 300
```

Install and configure a compute node for Red Hat Enterprise Linux and CentOS

This section describes how to install and configure the Compute service on a compute node. The service supports several hypervisors to deploy instances or virtual machines (VMs). For simplicity, this configuration uses the Quick EMUlator (QEMU) hypervisor with the kernel-based VM (KVM) extension on compute nodes that support hardware acceleration for virtual machines. On legacy hardware, this configuration uses the generic QEMU hypervisor. You can follow these instructions with minor modifications to horizontally scale your environment with additional compute nodes.

Note: This section assumes that you are following the instructions in this guide step-by-step to configure the first compute node. If you want to configure additional compute nodes, prepare them in a similar fashion to the first compute node in the *example architectures* section. Each additional compute node requires a unique IP address.

Install and configure components

Note: Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# yum install openstack-nova-compute
```

2. Edit the `/etc/nova/nova.conf` file and complete the following actions:

   - In the `[DEFAULT]` section, enable only the compute and metadata APIs:

```
[DEFAULT]
# ...
enabled_apis = osapi_compute,metadata
```

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

- In the `[api]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[api]
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS
```

Replace `NOVA_PASS` with the password you chose for the `nova` user in the Identity service.

Note: Comment out or remove any other options in the `[keystone_authtoken]` section.

- In the `[DEFAULT]` section, configure the `my_ip` option:

```
[DEFAULT]
# ...
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network interface on your compute node, typically 10.0.0.31 for the first node in the *example architecture*.

- In the `[DEFAULT]` section, enable support for the Networking service:

```
[DEFAULT]
# ...
use_neutron = true
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

Note: By default, Compute uses an internal firewall service. Since Networking includes a firewall service, you must disable the Compute firewall service by using the `nova.virt.firewall.NoopFirewallDriver` firewall driver.

- Configure the `[neutron]` section of **/etc/nova/nova.conf**. Refer to the Networking service install guide for more details.
- In the `[vnc]` section, enable and configure remote console access:

```
[vnc]
# ...
```

(continues on next page)

(continued from previous page)

```
enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html
```

The server component listens on all IP addresses and the proxy component only listens on the management interface IP address of the compute node. The base URL indicates the location where you can use a web browser to access remote consoles of instances on this compute node.

Note: If the web browser to access remote consoles resides on a host that cannot resolve the `controller` hostname, you must replace `controller` with the management interface IP address of the controller node.

- In the `[glance]` section, configure the location of the Image service API:

```
[glance]
# ...
api_servers = http://controller:9292
```

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
# ...
lock_path = /var/lib/nova/tmp
```

- In the `[placement]` section, configure the Placement API:

```
[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS
```

Replace `PLACEMENT_PASS` with the password you choose for the `placement` user in the Identity service. Comment out any other options in the `[placement]` section.

Finalize installation

1. Determine whether your compute node supports hardware acceleration for virtual machines:

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

If this command returns a value of `one or greater`, your compute node supports hardware acceleration which typically requires no additional configuration.

If this command returns a value of `zero`, your compute node does not support hardware acceleration and you must configure `libvirt` to use QEMU instead of KVM.

- Edit the `[libvirt]` section in the `/etc/nova/nova.conf` file as follows:

```
[libvirt]
# ...
virt_type = qemu
```

2. Start the Compute service including its dependencies and configure them to start automatically when the system boots:

```
# systemctl enable libvirtd.service openstack-nova-compute.service
# systemctl start libvirtd.service openstack-nova-compute.service
```

Note: If the `nova-compute` service fails to start, check `/var/log/nova/nova-compute.log`. The error message `AMQP server on controller:5672 is unreachable` likely indicates that the firewall on the controller node is preventing access to port 5672. Configure the firewall to open port 5672 on the controller node and restart `nova-compute` service on the compute node.

Add the compute node to the cell database

Important: Run the following commands on the **controller** node.

1. Source the admin credentials to enable admin-only CLI commands, then confirm there are compute hosts in the database:

```
$ . admin-openrc

$ openstack compute service list --service nova-compute
+----+-------+--------------+------+-------+---------+----------------------------
↪+
| ID | Host  | Binary       | Zone | State | Status  | Updated At                 ␣
↪|
+----+-------+--------------+------+-------+---------+----------------------------
↪+
| 1  | node1 | nova-compute | nova | up    | enabled | 2017-04-14T15:30:44.000000␣
↪|
+----+-------+--------------+------+-------+---------+----------------------------
↪+
```

2. Discover compute hosts:

```
# su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova

Found 2 cell mappings.
Skipping cell0 since it does not contain hosts.
Getting compute nodes from cell 'cell1': ad5a5985-a719-4567-98d8-8d148aaae4bc
Found 1 computes in cell: ad5a5985-a719-4567-98d8-8d148aaae4bc
Checking host mapping for compute host 'compute': fe58ddc1-1d65-4f87-9456-
↪bc040dc106b3
Creating host mapping for compute host 'compute': fe58ddc1-1d65-4f87-9456-
↪bc040dc106b3
```

Note: When you add new compute nodes, you must run `nova-manage cell_v2 discover_hosts` on the controller node to register those new compute nodes. Alternatively, you can set an appropriate interval in

/etc/nova/nova.conf:

```
[scheduler]
discover_hosts_in_cells_interval = 300
```

Install and configure a compute node for openSUSE and SUSE Linux Enterprise

This section describes how to install and configure the Compute service on a compute node. The service supports several hypervisors to deploy instances or virtual machines (VMs). For simplicity, this configuration uses the Quick EMUlator (QEMU) hypervisor with the kernel-based VM (KVM) extension on compute nodes that support hardware acceleration for virtual machines. On legacy hardware, this configuration uses the generic QEMU hypervisor. You can follow these instructions with minor modifications to horizontally scale your environment with additional compute nodes.

Note: This section assumes that you are following the instructions in this guide step-by-step to configure the first compute node. If you want to configure additional compute nodes, prepare them in a similar fashion to the first compute node in the *example architectures* section. Each additional compute node requires a unique IP address.

Install and configure components

Note: Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (...) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# zypper install openstack-nova-compute genisoimage qemu-kvm libvirt
```

2. Edit the /etc/nova/nova.conf file and complete the following actions:

   - In the [DEFAULT] section, enable only the compute and metadata APIs:

```
[DEFAULT]
# ...
enabled_apis = osapi_compute,metadata
```

   - In the [DEFAULT] section, set the compute_driver:

```
[DEFAULT]
# ...
compute_driver = libvirt.LibvirtDriver
```

   - In the [DEFAULT] section, configure RabbitMQ message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

     Replace RABBIT_PASS with the password you chose for the openstack account in RabbitMQ.

   - In the [api] and [keystone_authtoken] sections, configure Identity service access:

```
[api]
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS
```

Replace `NOVA_PASS` with the password you chose for the `nova` user in the Identity service.

Note: Comment out or remove any other options in the `[keystone_authtoken]` section.

- In the `[DEFAULT]` section, configure the `my_ip` option:

```
[DEFAULT]
# ...
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network interface on your compute node, typically `10.0.0.31` for the first node in the *example architecture*.

- In the `[DEFAULT]` section, enable support for the Networking service:

```
[DEFAULT]
# ...
use_neutron = true
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

Note: By default, Compute uses an internal firewall service. Since Networking includes a firewall service, you must disable the Compute firewall service by using the `nova.virt.firewall.NoopFirewallDriver` firewall driver.

- Configure the `[neutron]` section of **/etc/nova/nova.conf**. Refer to the Networking service install guide for more details.
- In the `[vnc]` section, enable and configure remote console access:

```
[vnc]
# ...
enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html
```

The server component listens on all IP addresses and the proxy component only listens on the management interface IP address of the compute node. The base URL indicates the location where you can use a web browser to access remote consoles of instances on this compute node.

Note: If the web browser to access remote consoles resides on a host that cannot resolve the `controller` hostname, you must replace `controller` with the management interface IP address of the controller node.

- In the `[glance]` section, configure the location of the Image service API:

```
[glance]
# ...
api_servers = http://controller:9292
```

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
# ...
lock_path = /var/run/nova
```

- In the `[placement]` section, configure the Placement API:

```
[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS
```

Replace `PLACEMENT_PASS` with the password you choose for the `placement` user in the Identity service. Comment out any other options in the `[placement]` section.

3. Ensure the kernel module `nbd` is loaded.

```
# modprobe nbd
```

4. Ensure the module loads on every boot by adding `nbd` to the `/etc/modules-load.d/nbd.conf` file.

Finalize installation

1. Determine whether your compute node supports hardware acceleration for virtual machines:

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

If this command returns a value of `one or greater`, your compute node supports hardware acceleration which typically requires no additional configuration.

If this command returns a value of `zero`, your compute node does not support hardware acceleration and you must configure `libvirt` to use QEMU instead of KVM.

- Edit the `[libvirt]` section in the `/etc/nova/nova.conf` file as follows:

```
[libvirt]
# ...
virt_type = qemu
```

2. Start the Compute service including its dependencies and configure them to start automatically when the system boots:

```
# systemctl enable libvirtd.service openstack-nova-compute.service
# systemctl start libvirtd.service openstack-nova-compute.service
```

Note: If the `nova-compute` service fails to start, check `/var/log/nova/nova-compute.log`. The error message `AMQP server on controller:5672 is unreachable` likely indicates that the firewall on the controller node is preventing access to port 5672. Configure the firewall to open port 5672 on the controller node and restart `nova-compute` service on the compute node.

Add the compute node to the cell database

Important: Run the following commands on the **controller** node.

1. Source the admin credentials to enable admin-only CLI commands, then confirm there are compute hosts in the database:

```
$ . admin-openrc

$ openstack compute service list --service nova-compute
+----+-------+--------------+------+-------+---------+----------------------------
↪+
| ID | Host  | Binary       | Zone | State | Status  | Updated At                 ␣
↪|
+----+-------+--------------+------+-------+---------+----------------------------
↪+
| 1  | node1 | nova-compute | nova | up    | enabled | 2017-04-14T15:30:44.000000␣
↪|
+----+-------+--------------+------+-------+---------+----------------------------
↪+
```

2. Discover compute hosts:

```
# su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova

Found 2 cell mappings.
Skipping cell0 since it does not contain hosts.
Getting compute nodes from cell 'cell1': ad5a5985-a719-4567-98d8-8d148aaae4bc
Found 1 computes in cell: ad5a5985-a719-4567-98d8-8d148aaae4bc
Checking host mapping for compute host 'compute': fe58ddc1-1d65-4f87-9456-
↪bc040dc106b3
Creating host mapping for compute host 'compute': fe58ddc1-1d65-4f87-9456-
↪bc040dc106b3
```

Note: When you add new compute nodes, you must run `nova-manage cell_v2 discover_hosts` on the controller node to register those new compute nodes. Alternatively, you can set an appropriate interval in `/etc/nova/nova.conf`:

```
[scheduler]
discover_hosts_in_cells_interval = 300
```

Verify operation

Verify operation of the Compute service.

Note: Perform these commands on the controller node.

1. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

2. List service components to verify successful launch and registration of each process:

```
$ openstack compute service list

+----+--------------------+------------+----------+---------+-------+----------
↪---------------+
| Id | Binary             | Host       | Zone     | Status  | State | Updated At ␣
↪               |
+----+--------------------+------------+----------+---------+-------+----------
↪---------------+
|  1 | nova-scheduler     | controller | internal | enabled | up    | 2016-02-
↪09T23:11:15.000000 |
|  2 | nova-conductor     | controller | internal | enabled | up    | 2016-02-
↪09T23:11:16.000000 |
|  3 | nova-compute       | compute1   | nova     | enabled | up    | 2016-02-
↪09T23:11:20.000000 |
+----+--------------------+------------+----------+---------+-------+----------
↪---------------+
```

Note: This output should indicate two service components enabled on the controller node and one service component enabled on the compute node.

3. List API endpoints in the Identity service to verify connectivity with the Identity service:

Note: Below endpoints list may differ depending on the installation of OpenStack components.

```
$ openstack catalog list

+-----------+-----------+-----------------------------------------+
| Name      | Type      | Endpoints                               |
+-----------+-----------+-----------------------------------------+
| keystone  | identity  | RegionOne                               |
|           |           |   public: http://controller:5000/v3/    |
|           |           | RegionOne                               |
|           |           |   internal: http://controller:5000/v3/  |
|           |           | RegionOne                               |
|           |           |   admin: http://controller:5000/v3/     |
|           |           |                                         |
| glance    | image     | RegionOne                               |
|           |           |   admin: http://controller:9292         |
```

(continues on next page)

(continued from previous page)

```
|            |           | RegionOne                                    |
|            |           |   public: http://controller:9292            |
|            |           | RegionOne                                    |
|            |           |   internal: http://controller:9292           |
|            |           |                                              |
| nova       | compute   | RegionOne                                    |
|            |           |   admin: http://controller:8774/v2.1         |
|            |           | RegionOne                                    |
|            |           |   internal: http://controller:8774/v2.1      |
|            |           | RegionOne                                    |
|            |           |   public: http://controller:8774/v2.1        |
|            |           |                                              |
| placement  | placement | RegionOne                                    |
|            |           |   public: http://controller:8778             |
|            |           | RegionOne                                    |
|            |           |   admin: http://controller:8778              |
|            |           | RegionOne                                    |
|            |           |   internal: http://controller:8778           |
|            |           |                                              |
+------------+-----------+----------------------------------------------+
```

Note: Ignore any warnings in this output.

4. List images in the Image service to verify connectivity with the Image service:

```
$ openstack image list

+--------------------------------------+-------------+-------------+
| ID                                   | Name        | Status      |
+--------------------------------------+-------------+-------------+
| 9a76d9f9-9620-4f2e-8c69-6c5691fae163 | cirros      | active      |
+--------------------------------------+-------------+-------------+
```

5. Check the cells and placement API are working successfully and that other necessary prerequisites are in place:

```
# nova-status upgrade check

+---------------------------------------------------------------------+
| Upgrade Check Results                                               |
+---------------------------------------------------------------------+
| Check: Cells v2                                                     |
| Result: Success                                                     |
| Details: None                                                       |
+---------------------------------------------------------------------+
| Check: Placement API                                                |
| Result: Success                                                     |
| Details: None                                                       |
+---------------------------------------------------------------------+
| Check: Ironic Flavor Migration                                      |
| Result: Success                                                     |
| Details: None                                                       |
+---------------------------------------------------------------------+
| Check: Cinder API                                                   |
| Result: Success                                                     |
| Details: None                                                       |
```

(continues on next page)

(continued from previous page)

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

3.3 Deployment Considerations

There is information you might want to consider before doing your deployment, especially if it is going to be a larger deployment. For smaller deployments the defaults from the *install guide* will be sufficient.

- **Compute Driver Features Supported**: While the majority of nova deployments use libvirt/kvm, you can use nova with other compute drivers. Nova attempts to provide a unified feature set across these, however, not all features are implemented on all backends, and not all features are equally well tested.
  - *Feature Support by Use Case*: A view of what features each driver supports based on whats important to some large use cases (General Purpose Cloud, NFV Cloud, HPC Cloud).
  - *Feature Support full list*: A detailed dive through features in each compute driver backend.
- *Cells v2 Planning*: For large deployments, Cells v2 allows sharding of your compute environment. Upfront planning is key to a successful Cells v2 layout.
- *Running nova-api on wsgi*: Considerations for using a real WSGI container instead of the baked-in eventlet web server.

3.4 Maintenance

Once you are running nova, the following information is extremely useful.

- *Admin Guide*: A collection of guides for administrating nova.
- *Flavors*: What flavors are and why they are used.
- *Upgrades*: How nova is designed to be upgraded for minimal service impact, and the order you should do them in.
- *Quotas*: Managing project quotas in nova.
- *Aggregates*: Aggregates are a useful way of grouping hosts together for scheduling purposes.
- *Filter Scheduler*: How the filter scheduler is configured, and how that will impact where compute instances land in your environment. If you are seeing unexpected distribution of compute instances in your hosts, youll want to dive into this configuration.
- *Exposing custom metadata to compute instances*: How and when you might want to extend the basic metadata exposed to compute instances (either via metadata server or config drive) for your specific purposes.

3.5 Reference Material

- *Nova CLI Command References*: the complete command reference for all the daemons and admin tools that come with nova.
- *Configuration Guide*: Information on configuring the system, including role-based access control policy rules.

CHAPTER

FOUR

FOR CONTRIBUTORS

If you are new to Nova, this should help you start to understand what Nova actually does, and why.

4.1 Contributor Documentation

Contributing to nova gives you the power to help add features, fix bugs, enhance documentation, and increase testing. Contributions of any type are valuable, and part of what keeps the project going. Here are a list of resources to get your started.

4.1.1 Getting Started

- *How to get (more) involved with Nova*: Overview of engaging in the project
- *Development Quickstart*: Get your computer setup to contribute

4.1.2 Nova Process

The nova community is a large community. We have lots of users, and they all have a lot of expectations around upgrade and backwards compatibility. For example, having a good stable API, with discoverable versions and capabilities is important for maintaining the strong ecosystem around nova.

Our process is always evolving, just as nova and the community around nova evolves over time. If there are things that seem strange, or you have ideas on how to improve things, please bring them forward on IRC or the openstack-discuss mailing list, so we continue to improve how the nova community operates.

This section looks at the processes and why. The main aim behind all the process is to aid communication between all members of the nova community, while keeping users happy and keeping developers productive.

- *Scope of the Nova project*: The focus is on features and bug fixes that make nova work better within this scope
- *Development policies*: General guidelines about whats supported
- *Nova team process*: The processes we follow around feature and bug submission, including how the release calendar works, and the freezes we go under
- *Blueprints, Specs and Priorities*: An overview of our tracking artifacts.
- *Chronological PTL guide*: A chronological PTL reference guide

4.1.3 Reviewing

- *Release Notes*: When we need a release note for a contribution.
- *Code Review Guide for Nova*: Important cheat sheet for whats important when doing code review in Nova, especially some things that are hard to test for, but need human eyes.
- *Internationalization*: What we require for i18n in patches
- *Documentation Guidelines*: Guidelines for handling documentation contributions

4.1.4 Testing

Because Python is a dynamic language, code that is not tested might not even be Python code. All new code needs to be validated somehow.

- *Test Strategy*: An overview of our test taxonomy and the kinds of testing we do and expect.
- **Testing Guides**: There are also specific testing guides for features that are hard to test in our gate.
  - *Testing NUMA related hardware setup with libvirt*
  - *Testing Serial Console*
  - *Testing Zero Downtime Upgrade Process*
  - *Testing Down Cells*
- **Profiling Guides**: These are guides to profiling nova.
  - *Profiling With Eventlet*

4.1.5 The Nova API

Because we have many consumers of our API, were extremely careful about changes done to the API, as the impact can be very wide.

- *Extending the API*: How the code is structured inside the API layer
- *Adding a Method to the OpenStack API*: (needs update)
- *API Microversions*: How the API is (micro)versioned and what you need to do when adding an API exposed feature that needs a new microversion.
- *API reference guideline*: The guideline to write the API reference.

4.1.6 Nova Major Subsystems

Major subsystems in nova have different needs. If you are contributing to one of these please read the *reference guide* before diving in.

Move operations

- *Evacuate vs Rebuild*: Describes the differences between the often-confused evacuate and rebuild operations.

There are also a number of technical references on both current and future looking parts of our architecture. These are collected below.

4.2 Technical Reference Deep Dives

The nova project is large, and there are lots of complicated parts in it where it helps to have an overview to understand how the internals of a particular part work.

4.2.1 Internals

The following is a dive into some of the internals in nova.

- *AMQP and Nova*: How nova uses AMQP as an RPC transport
- *Scheduling*: The workflow through the scheduling process
- *Scheduler hints versus flavor extra specs*: The similarities and differences between flavor extra specs and scheduler hints.
- *Live Migration*: The live migration flow
- *Services, Managers and Drivers*: Module descriptions for some of the key modules used in starting / running services
- *Virtual Machine States and Transitions*: Cheat sheet for understanding the life cycle of compute instances
- *Threading model*: The concurrency model used in nova, which is based on eventlet, and may not be familiar to everyone.
- *Notifications in Nova*: How the notifications subsystem works in nova, and considerations when adding notifications.
- *ComputeDriver.update_provider_tree*: A detailed explanation of the `ComputeDriver.update_provider_tree` method.
- *Upgrade checks*: A guide to writing automated upgrade checks.
- *Conductor as a place for orchestrating tasks*

Todo: Need something about versioned objects and how they fit in with conductor as an object backporter during upgrades.

- *Filtering hosts by isolating aggregates*: Describes how the placement filter works in nova to isolate groups of hosts.

4.2.2 Debugging

- *Guru Meditation Reports*: Inspired by Amiga, a way to trigger a very comprehensive dump of a running service for deep debugging.

4.2.3 Forward Looking Plans

The following section includes documents that describe the overall plan behind groups of nova-specs. Most of these cover items relating to the evolution of various parts of novas architecture. Once the work is complete, these documents will move into the Internals section.

If you want to get involved in shaping the future of novas architecture, these are a great place to start reading up on the current plans.

- *Cells*: How cells v2 is evolving
- *REST API Policy Enforcement*: How we want policy checks on API actions to work in the future
- *Nova Stable REST API*: What stable api means to nova
- *Scheduler Evolution*: Motivation behind the scheduler / placement evolution

4.3 Compute

The OpenStack Compute service allows you to control an Infrastructure-as-a-Service (IaaS) cloud computing platform. It gives you control over instances and networks, and allows you to manage access to the cloud through users and projects.

Compute does not include virtualization software. Instead, it defines drivers that interact with underlying virtualization mechanisms that run on your host operating system, and exposes functionality over a web-based API.

4.3.1 Injecting the administrator password

Compute can generate a random administrator (root) password and inject that password into an instance. If this feature is enabled, users can run **ssh** to an instance without an **ssh** keypair. The random password appears in the output of the **openstack server create** command. You can also view and set the admin password from the dashboard.

Password injection using the dashboard

By default, the dashboard will display the `admin` password and allow the user to modify it.

If you do not want to support password injection, disable the password fields by editing the dashboards `local_settings.py` file.

```
OPENSTACK_HYPERVISOR_FEATURES = {
...
    'can_set_password': False,
}
```

Password injection on libvirt-based hypervisors

For hypervisors that use the libvirt back end (such as KVM, QEMU, and LXC), admin password injection is disabled by default. To enable it, set this option in `/etc/nova/nova.conf`:

```
[libvirt]
inject_password=true
```

When enabled, Compute will modify the password of the admin account by editing the `/etc/shadow` file inside the virtual machine instance.

Note: Linux distribution guest only.

Note: Users can only use **ssh** to access the instance by using the admin password if the virtual machine image is a Linux distribution, and it has been configured to allow users to use **ssh** as the root user with password authorization.

This is not the case for Ubuntu cloud images which, by default, does not allow users to use **ssh** to access the root account, or CentOS cloud images which, by default, does not allow **ssh** access to the instance with password.

Password injection and XenAPI (XenServer/XCP)

When using the XenAPI hypervisor back end, Compute uses the XenAPI agent to inject passwords into guests. The virtual machine image must be configured with the agent for password injection to work.

Password injection and Windows images (all hypervisors)

For Windows virtual machines, configure the Windows image to retrieve the admin password on boot by installing an agent such as cloudbase-init.

4.3.2 Advanced configuration

OpenStack clouds run on platforms that differ greatly in the capabilities that they provide. By default, the Compute service seeks to abstract the underlying hardware that it runs on, rather than exposing specifics about the underlying host platforms. This abstraction manifests itself in many ways. For example, rather than exposing the types and topologies of CPUs running on hosts, the service exposes a number of generic CPUs (virtual CPUs, or vCPUs) and allows for overcommitting of these. In a similar manner, rather than exposing the individual types of network devices available on hosts, generic software-powered network ports are provided. These features are designed to allow high resource utilization and allows the service to provide a generic cost-effective and highly scalable cloud upon which to build applications.

This abstraction is beneficial for most workloads. However, there are some workloads where determinism and per-instance performance are important, if not vital. In these cases, instances can be expected to deliver near-native performance. The Compute service provides features to improve individual instance for these kind of workloads.

Important: In deployments older than Train, or in mixed Stein/Train deployments with a rolling upgrade in progress, unless *specifically enabled*, live migration is not possible for instances with a NUMA topology when using the libvirt driver. A NUMA topology may be specified explicitly or can be added implicitly due to the use of CPU pinning or huge pages. Refer to bug #1289064 for more information. As of Train, live migration of instances with a NUMA topology when using the libvirt driver is fully supported.

Attaching physical PCI devices to guests

The PCI passthrough feature in OpenStack allows full access and direct control of a physical PCI device in guests. This mechanism is generic for any kind of PCI device, and runs with a Network Interface Card (NIC), Graphics Processing Unit (GPU), or any other devices that can be attached to a PCI bus. Correct driver installation is the only requirement for the guest to properly use the devices.

Some PCI devices provide Single Root I/O Virtualization and Sharing (SR-IOV) capabilities. When SR-IOV is used, a physical device is virtualized and appears as multiple PCI devices. Virtual PCI devices are assigned to the same or different guests. In the case of PCI passthrough, the full physical device is assigned to only one guest and cannot be shared.

PCI devices are requested through flavor extra specs, specifically via the `pci_passthrough:alias=<alias>` flavor extra spec. This guide demonstrates how to enable PCI passthrough for a type of PCI device with a vendor ID of `8086` and a product ID of `154d` - an Intel X520 Network Adapter - by mapping them to the alias `a1`. You should adjust the instructions for other devices with potentially different capabilities.

Note: For information on creating servers with SR-IOV network interfaces, refer to the Networking Guide.

Limitations

- Attaching SR-IOV ports to existing servers is not currently supported. This is now rejected by the API but previously it fail later in the process. See bug 1708433 for details.
- Cold migration (resize) of servers with SR-IOV devices attached was not supported until the 14.0.0 Newton release, see bug 1512800 for details.

Configure host (Compute)

To enable PCI passthrough on an x86, Linux-based compute node, the following are required:

- VT-d enabled in the BIOS
- IOMMU enabled on the host OS, e.g. by adding the `intel_iommu=on` or `amd_iommu=on` parameter to the kernel parameters
- Assignable PCIe devices

To enable PCI passthrough on a Hyper-V compute node, the following are required:

- Windows 10 or Windows / Hyper-V Server 2016 or newer
- VT-d enabled on the host
- Assignable PCI devices

In order to check the requirements above and if there are any assignable PCI devices, run the following Powershell commands:

```
Start-BitsTransfer https://raw.githubusercontent.com/Microsoft/Virtualization-
↪Documentation/master/hyperv-samples/benarm-powershell/DDA/survey-dda.ps1
 .\survey-dda.ps1
```

If the compute node passes all the requirements, the desired assignable PCI devices to be disabled and unmounted from the host, in order to be assignable by Hyper-V. The following can be read for more details: Hyper-V PCI passthrough.

Configure `nova-compute` (Compute)

Once PCI passthrough has been configured for the host, **`nova-compute`** must be configured to allow the PCI device to pass through to VMs. This is done using the *`pci.passthrough_whitelist`* option. For example, assuming our sample PCI device has a PCI address of `41:00.0` on each host:

```
[pci]
passthrough_whitelist = { "address": "0000:41:00.0" }
```

Refer to *`pci.passthrough_whitelist`* for syntax information.

Alternatively, to enable passthrough of all devices with the same product and vendor ID:

```
[pci]
passthrough_whitelist = { "vendor_id": "8086", "product_id": "154d" }
```

If using vendor and product IDs, all PCI devices matching the `vendor_id` and `product_id` are added to the pool of PCI devices available for passthrough to VMs.

In addition, it is necessary to configure the *`pci.alias`* option, which is a JSON-style configuration option that allows you to map a given device type, identified by the standard PCI `vendor_id` and (optional) `product_id` fields, to an arbitrary name or *alias*. This alias can then be used to request a PCI device using the `pci_passthrough:alias=<alias>` flavor extra spec, as discussed previously. For our sample device with a vendor ID of `0x8086` and a product ID of `0x154d`, this would be:

```
[pci]
alias = { "vendor_id":"8086", "product_id":"154d", "device_type":"type-PF", "name":"a1
↪" }
```

Its important to note the addition of the `device_type` field. This is necessary because this PCI device supports SR-IOV. The `nova-compute` service categorizes devices into one of three types, depending on the capabilities the devices report:

`type-PF` The device supports SR-IOV and is the parent or root device.

`type-VF` The device is a child device of a device that supports SR-IOV.

`type-PCI` The device does not support SR-IOV.

By default, it is only possible to attach `type-PCI` devices using PCI passthrough. If you wish to attach `type-PF` or `type-VF` devices, you must specify the `device_type` field in the config option. If the device was a device that did not support SR-IOV, the `device_type` field could be omitted.

Refer to *`pci.alias`* for syntax information.

Important: This option must also be configured on controller nodes. This is discussed later in this document.

Once configured, restart the **`nova-compute`** service.

Configure `nova-scheduler` (Controller)

The **`nova-scheduler`** service must be configured to enable the `PciPassthroughFilter`. To do this, add this filter to the list of filters specified in *`filter_scheduler.enabled_filters`* and set *`filter_scheduler.available_filters`* to the default of `nova.scheduler.filters.all_filters`. For example:

```
[filter_scheduler]
enabled_filters = ...,PciPassthroughFilter
available_filters = nova.scheduler.filters.all_filters
```

Once done, restart the **`nova-scheduler`** service.

Configure `nova-api` (Controller)

It is necessary to also configure the *`pci.alias`* config option on the controller. This configuration should match the configuration found on the compute nodes. For example:

```
[pci]
alias = { "vendor_id":"8086", "product_id":"154d", "device_type":"type-PF", "name":"a1
↪" }
```

Refer to *pci.alias* for syntax information.

Once configured, restart the **nova-api** service.

Configure a flavor (API)

Once the alias has been configured, it can be used for an flavor extra spec. For example, to request two of the PCI devices referenced by alias `a1`, run:

```
$ openstack flavor set m1.large --property "pci_passthrough:alias"="a1:2"
```

For more information about the syntax for `pci_passthrough:alias`, refer to *Flavors*.

Create instances with PCI passthrough devices

The **nova-scheduler** service selects a destination host that has PCI devices available that match the `alias` specified in the flavor.

```
# openstack server create --flavor m1.large --image cirros-0.3.5-x86_64-uec --wait␣
↪test-pci
```

CPU topologies

The NUMA topology and CPU pinning features in OpenStack provide high-level control over how instances run on hypervisor CPUs and the topology of virtual CPUs available to instances. These features help minimize latency and maximize performance.

Important: In deployments older than Train, or in mixed Stein/Train deployments with a rolling upgrade in progress, unless *specifically enabled*, live migration is not possible for instances with a NUMA topology when using the libvirt driver. A NUMA topology may be specified explicitly or can be added implicitly due to the use of CPU pinning or huge pages. Refer to bug #1289064 for more information. As of Train, live migration of instances with a NUMA topology when using the libvirt driver is fully supported.

SMP, NUMA, and SMT

Symmetric multiprocessing (SMP) SMP is a design found in many modern multi-core systems. In an SMP system, there are two or more CPUs and these CPUs are connected by some interconnect. This provides CPUs with equal access to system resources like memory and input/output ports.

Non-uniform memory access (NUMA) NUMA is a derivative of the SMP design that is found in many multi-socket systems. In a NUMA system, system memory is divided into cells or nodes that are associated with particular CPUs. Requests for memory on other nodes are possible through an interconnect bus. However, bandwidth across this shared bus is limited. As a result, competition for this resource can incur performance penalties.

Simultaneous Multi-Threading (SMT) SMT is a design complementary to SMP. Whereas CPUs in SMP systems share a bus and some memory, CPUs in SMT systems share many more components. CPUs that share components are known as thread siblings. All CPUs appear as usable CPUs on the system and can execute workloads in parallel. However, as with NUMA, threads compete for shared resources.

Non-Uniform I/O Access (NUMA I/O) In a NUMA system, I/O to a device mapped to a local memory region is more efficient than I/O to a remote device. A device connected to the same socket providing the CPU and memory offers lower latencies for I/O operations due to its physical proximity. This generally manifests itself in devices connected to the PCIe bus, such as NICs or vGPUs, but applies to any device support memory-mapped I/O.

In OpenStack, SMP CPUs are known as *cores*, NUMA cells or nodes are known as *sockets*, and SMT CPUs are known as *threads*. For example, a quad-socket, eight core system with Hyper-Threading would have four sockets, eight cores per socket and two threads per core, for a total of 64 CPUs.

PCPU and VCPU

PCPU Resource class representing an amount of dedicated CPUs for a single guest.

VCPU Resource class representing a unit of CPU resources for a single guest approximating the processing power of a single physical processor.

Customizing instance NUMA placement policies

Important: The functionality described below is currently only supported by the libvirt/KVM and Hyper-V driver. The Hyper-V driver may require *some host configuration* for this to work.

When running workloads on NUMA hosts, it is important that the vCPUs executing processes are on the same NUMA node as the memory used by these processes. This ensures all memory accesses are local to the node and thus do not consume the limited cross-node memory bandwidth, adding latency to memory accesses. Similarly, large pages are assigned from memory and benefit from the same performance improvements as memory allocated using standard pages. Thus, they also should be local. Finally, PCI devices are directly associated with specific NUMA nodes for the purposes of DMA. Instances that use PCI or SR-IOV devices should be placed on the NUMA node associated with these devices.

NUMA topology can exist on both the physical hardware of the host and the virtual hardware of the instance. In OpenStack, when booting a process, the hypervisor driver looks at the NUMA topology field of both the instance and the host it is being booted on, and uses that information to generate an appropriate configuration.

By default, an instance floats across all NUMA nodes on a host. NUMA awareness can be enabled implicitly through the use of huge pages or pinned CPUs or explicitly through the use of flavor extra specs or image metadata. If the instance has requested a specific NUMA topology, compute will try to pin the vCPUs of different NUMA cells on the instance to the corresponding NUMA cells on the host. It will also expose the NUMA topology of the instance to the guest OS.

In all cases where NUMA awareness is used, the `NUMATopologyFilter` filter must be enabled. Details on this filter are provided in *Compute schedulers*.

Caution: The NUMA node(s) used are normally chosen at random. However, if a PCI passthrough or SR-IOV device is attached to the instance, then the NUMA node that the device is associated with will be used. This can provide important performance improvements. However, booting a large number of similar instances can result in unbalanced NUMA node usage. Care should be taken to mitigate this issue. See this discussion for more details.

Caution: Inadequate per-node resources will result in scheduling failures. Resources that are specific to a node include not only CPUs and memory, but also PCI and SR-IOV resources. It is not possible to use multiple resources

from different nodes without requesting a multi-node layout. As such, it may be necessary to ensure PCI or SR-IOV resources are associated with the same NUMA node or force a multi-node layout.

When used, NUMA awareness allows the operating system of the instance to intelligently schedule the workloads that it runs and minimize cross-node memory bandwidth. To restrict an instances vCPUs to a single host NUMA node, run:

```
$ openstack flavor set [FLAVOR_ID] --property hw:numa_nodes=1
```

Some workloads have very demanding requirements for memory access latency or bandwidth that exceed the memory bandwidth available from a single NUMA node. For such workloads, it is beneficial to spread the instance across multiple host NUMA nodes, even if the instances RAM/vCPUs could theoretically fit on a single NUMA node. To force an instances vCPUs to spread across two host NUMA nodes, run:

```
$ openstack flavor set [FLAVOR_ID] --property hw:numa_nodes=2
```

The allocation of instances vCPUs and memory from different host NUMA nodes can be configured. This allows for asymmetric allocation of vCPUs and memory, which can be important for some workloads. To spread the 6 vCPUs and 6 GB of memory of an instance across two NUMA nodes and create an asymmetric 1:2 vCPU and memory mapping between the two nodes, run:

```
$ openstack flavor set [FLAVOR_ID] --property hw:numa_nodes=2
# configure guest node 0
$ openstack flavor set [FLAVOR_ID] \
  --property hw:numa_cpus.0=0,1 \
  --property hw:numa_mem.0=2048
# configure guest node 1
$ openstack flavor set [FLAVOR_ID] \
  --property hw:numa_cpus.1=2,3,4,5 \
  --property hw:numa_mem.1=4096
```

Note: Hyper-V does not support asymmetric NUMA topologies, and the Hyper-V driver will not spawn instances with such topologies.

For more information about the syntax for `hw:numa_nodes`, `hw:numa_cpus.N` and `hw:num_mem.N`, refer to the *NUMA topology* guide.

Customizing instance CPU pinning policies

Important: The functionality described below is currently only supported by the libvirt/KVM driver and requires *some host configuration* for this to work. Hyper-V does not support CPU pinning.

Note: There is no correlation required between the NUMA topology exposed in the instance and how the instance is actually pinned on the host. This is by design. See this invalid bug for more information.

By default, instance vCPU processes are not assigned to any particular host CPU, instead, they float across host CPUs like any other process. This allows for features like overcommitting of CPUs. In heavily contended systems, this provides optimal system performance at the expense of performance and latency for individual instances.

Some workloads require real-time or near real-time behavior, which is not possible with the latency introduced by the default CPU policy. For such workloads, it is beneficial to control which host CPUs are bound to an instances vCPUs. This process is known as pinning. No instance with pinned CPUs can use the CPUs of another pinned instance, thus preventing resource contention between instances.

CPU pinning policies can be used to determine whether an instance should be pinned or not. There are two policies: `dedicated` and `shared` (the default). The `dedicated` CPU policy is used to specify that an instance should use pinned CPUs. To configure a flavor to use the `dedicated` CPU policy, run:

```
$ openstack flavor set [FLAVOR_ID] --property hw:cpu_policy=dedicated
```

This works by ensuring `PCPU` allocations are used instead of `VCPU` allocations. As such, it is also possible to request this resource type explicitly. To configure this, run:

```
$ openstack flavor set [FLAVOR_ID] --property resources:PCPU=N
```

where `N` is the number of vCPUs defined in the flavor.

Note: It is not currently possible to request `PCPU` and `VCPU` resources in the same instance.

The `shared` CPU policy is used to specify that an instance **should not** use pinned CPUs. To configure a flavor to use the `shared` CPU policy, run:

```
$ openstack flavor set [FLAVOR_ID] --property hw:cpu_policy=shared
```

Note: For more information about the syntax for `hw:cpu_policy`, refer to the *Manage Flavors* guide.

It is also possible to configure the CPU policy via image metadata. This can be useful when packaging applications that require real-time or near real-time behavior by ensuring instances created with a given image are always pinned regardless of flavor. To configure an image to use the `dedicated` CPU policy, run:

```
$ openstack image set [IMAGE_ID] --property hw_cpu_policy=dedicated
```

Likewise, to configure an image to use the `shared` CPU policy, run:

```
$ openstack image set [IMAGE_ID] --property hw_cpu_policy=shared
```

Note: For more information about image metadata, refer to the Image metadata guide.

Important: Flavor-based policies take precedence over image-based policies. For example, if a flavor specifies a CPU policy of `dedicated` then that policy will be used. If the flavor specifies a CPU policy of `shared` and the image specifies no policy or a policy of `shared` then the `shared` policy will be used. However, the flavor specifies a CPU policy of `shared` and the image specifies a policy of `dedicated`, or vice versa, an exception will be raised. This is by design. Image metadata is often configurable by non-admin users, while flavors are only configurable by admins. By setting a `shared` policy through flavor extra-specs, administrators can prevent users configuring CPU policies in images and impacting resource utilization.

Customizing instance CPU thread pinning policies

Important: The functionality described below requires the use of pinned instances and is therefore currently only supported by the libvirt/KVM driver and requires *some host configuration* for this to work. Hyper-V does not support CPU pinning.

When running pinned instances on SMT hosts, it may also be necessary to consider the impact that thread siblings can have on the instance workload. The presence of an SMT implementation like Intel Hyper-Threading can boost performance by up to 30% for some workloads. However, thread siblings share a number of components and contention on these components can diminish performance for other workloads. For this reason, it is also possible to explicitly request hosts with or without SMT.

To configure whether an instance should be placed on a host with SMT or not, a CPU thread policy may be specified. For workloads where sharing benefits performance, you can request hosts **with** SMT. To configure this, run:

```
$ openstack flavor set [FLAVOR_ID] \
  --property hw:cpu_policy=dedicated \
  --property hw:cpu_thread_policy=require
```

This will ensure the instance gets scheduled to a host with SMT by requesting hosts that report the `HW_CPU_HYPERTHREADING` trait. It is also possible to request this trait explicitly. To configure this, run:

```
$ openstack flavor set [FLAVOR_ID] \
  --property resources:PCPU=N \
  --property trait:HW_CPU_HYPERTHREADING=required
```

For other workloads where performance is impacted by contention for resources, you can request hosts **without** SMT. To configure this, run:

```
$ openstack flavor set [FLAVOR_ID] \
  --property hw:cpu_policy=dedicated \
  --property hw:cpu_thread_policy=isolate
```

This will ensure the instance gets scheduled to a host with SMT by requesting hosts that **do not** report the `HW_CPU_HYPERTHREADING` trait. It is also possible to request this trait explicitly. To configure this, run:

```
$ openstack flavor set [FLAVOR_ID] \
  --property resources:PCPU=N \
  --property trait:HW_CPU_HYPERTHREADING=forbidden
```

Finally, for workloads where performance is minimally impacted, you may use thread siblings if available and fallback to not using them if necessary. This is the default, but it can be set explicitly:

```
$ openstack flavor set [FLAVOR_ID] \
  --property hw:cpu_policy=dedicated \
  --property hw:cpu_thread_policy=prefer
```

This does not utilize traits and, as such, there is no trait-based equivalent.

Note: For more information about the syntax for `hw:cpu_thread_policy`, refer to the *Manage Flavors* guide.

As with CPU policies, it also possible to configure the CPU thread policy via image metadata. This can be useful when packaging applications that require real-time or near real-time behavior by ensuring instances created with a given image are always pinned regardless of flavor. To configure an image to use the `require` CPU policy, run:

```
$ openstack image set [IMAGE_ID] \
  --property hw_cpu_policy=dedicated \
  --property hw_cpu_thread_policy=require
```

Likewise, to configure an image to use the `isolate` CPU thread policy, run:

```
$ openstack image set [IMAGE_ID] \
  --property hw_cpu_policy=dedicated \
  --property hw_cpu_thread_policy=isolate
```

Finally, to configure an image to use the `prefer` CPU thread policy, run:

```
$ openstack image set [IMAGE_ID] \
  --property hw_cpu_policy=dedicated \
  --property hw_cpu_thread_policy=prefer
```

If the flavor does not specify a CPU thread policy then the CPU thread policy specified by the image (if any) will be used. If both the flavor and image specify a CPU thread policy then they must specify the same policy, otherwise an exception will be raised.

Note: For more information about image metadata, refer to the Image metadata guide.

Customizing instance emulator thread pinning policies

Important: The functionality described below requires the use of pinned instances and is therefore currently only supported by the libvirt/KVM driver and requires *some host configuration* for this to work. Hyper-V does not support CPU pinning.

In addition to the work of the guest OS and applications running in an instance, there is a small amount of overhead associated with the underlying hypervisor. By default, these overhead tasks - known collectively as emulator threads - run on the same host CPUs as the instance itself and will result in a minor performance penalty for the instance. This is not usually an issue, however, for things like real-time instances, it may not be acceptable for emulator thread to steal time from instance CPUs.

Emulator thread policies can be used to ensure emulator threads are run on cores separate from those used by the instance. There are two policies: `isolate` and `share`. The default is to run the emulator threads on the same core. The `isolate` emulator thread policy is used to specify that emulator threads for a given instance should be run on their own unique core, chosen from one of the host cores listed in *`compute.cpu_dedicated_set`*. To configure a flavor to use the `isolate` emulator thread policy, run:

```
$ openstack flavor set [FLAVOR_ID] \
  --property hw:cpu_policy=dedicated \
  --property hw:emulator_threads_policy=isolate
```

The `share` policy is used to specify that emulator threads from a given instance should be run on the pool of host cores listed in *`compute.cpu_shared_set`*. To configure a flavor to use the `share` emulator thread policy, run:

```
$ openstack flavor set [FLAVOR_ID] \
  --property hw:cpu_policy=dedicated \
  --property hw:emulator_threads_policy=share
```

Note: For more information about the syntax for `hw:emulator_threads_policy`, refer to the *Manage Flavors* guide.

Customizing instance CPU topologies

Important: The functionality described below is currently only supported by the libvirt/KVM driver.

Note: Currently it also works with libvirt/QEMU driver but we dont recommend it in production use cases. This is because vCPUs are actually running in one thread on host in qemu TCG (Tiny Code Generator), which is the backend for libvirt/QEMU driver. Work to enable full multi-threading support for TCG (a.k.a. MTTCG) is on going in QEMU community. Please see this MTTCG project page for detail.

In addition to configuring how an instance is scheduled on host CPUs, it is possible to configure how CPUs are represented in the instance itself. By default, when instance NUMA placement is not specified, a topology of N sockets, each with one core and one thread, is used for an instance, where N corresponds to the number of instance vCPUs requested. When instance NUMA placement is specified, the number of sockets is fixed to the number of host NUMA nodes to use and the total number of instance CPUs is split over these sockets.

Some workloads benefit from a custom topology. For example, in some operating systems, a different license may be needed depending on the number of CPU sockets. To configure a flavor to use a maximum of two sockets, run:

```
$ openstack flavor set [FLAVOR_ID] --property hw:cpu_sockets=2
```

Similarly, to configure a flavor to use one core and one thread, run:

```
$ openstack flavor set [FLAVOR_ID] \
  --property hw:cpu_cores=1 \
  --property hw:cpu_threads=1
```

Caution: If specifying all values, the product of sockets multiplied by cores multiplied by threads must equal the number of instance vCPUs. If specifying any one of these values or the multiple of two values, the values must be a factor of the number of instance vCPUs to prevent an exception. For example, specifying `hw:cpu_sockets=2` on a host with an odd number of cores fails. Similarly, specifying `hw:cpu_cores=2` and `hw:cpu_threads=4` on a host with ten cores fails.

For more information about the syntax for `hw:cpu_sockets`, `hw:cpu_cores` and `hw:cpu_threads`, refer to the *Manage Flavors* guide.

It is also possible to set upper limits on the number of sockets, cores, and threads used. Unlike the hard values above, it is not necessary for this exact number to used because it only provides a limit. This can be used to provide some flexibility in scheduling, while ensuring certain limits are not exceeded. For example, to ensure no more than two sockets are defined in the instance topology, run:

```
$ openstack flavor set [FLAVOR_ID] --property hw:cpu_max_sockets=2
```

For more information about the syntax for `hw:cpu_max_sockets`, `hw:cpu_max_cores`, and `hw:cpu_max_threads`, refer to the *Manage Flavors* guide.

Applications are frequently packaged as images. For applications that prefer certain CPU topologies, configure image metadata to hint that created instances should have a given topology regardless of flavor. To configure an image to request a two-socket, four-core per socket topology, run:

```
$ openstack image set [IMAGE_ID] \
  --property hw_cpu_sockets=2 \
  --property hw_cpu_cores=4
```

To constrain instances to a given limit of sockets, cores or threads, use the `max_` variants. To configure an image to have a maximum of two sockets and a maximum of one thread, run:

```
$ openstack image set [IMAGE_ID] \
  --property hw_cpu_max_sockets=2 \
  --property hw_cpu_max_threads=1
```

The value specified in the flavor is treated as the absolute limit. The image limits are not permitted to exceed the flavor limits, they can only be equal to or lower than what the flavor defines. By setting a `max` value for sockets, cores, or threads, administrators can prevent users configuring topologies that might, for example, incur an additional licensing fees.

For more information about image metadata, refer to the Image metadata guide.

Configuring libvirt compute nodes for CPU pinning

Changed in version 20.0.0: Prior to 20.0.0 (Train), it was not necessary to explicitly configure hosts for pinned instances. However, it was not possible to place pinned instances on the same host as unpinned CPUs, which typically meant hosts had to be grouped into host aggregates. If this was not done, unpinned instances would continue floating across all enabled host CPUs, even those that some instance CPUs were pinned to. Starting in 20.0.0, it is necessary to explicitly identify the host cores that should be used for pinned instances.

Nova treats host CPUs used for unpinned instances differently from those used by pinned instances. The former are tracked in placement using the `VCPU` resource type and can be overallocated, while the latter are tracked using the `PCPU` resource type. By default, nova will report all host CPUs as `VCPU` inventory, however, this can be configured using the *`compute.cpu_shared_set`* config option, to specify which host CPUs should be used for `VCPU` inventory, and the *`compute.cpu_dedicated_set`* config option, to specify which host CPUs should be used for `PCPU` inventory.

Consider a compute node with a total of 24 host physical CPU cores with hyperthreading enabled. The operator wishes to reserve 1 physical CPU core and its thread sibling for host processing (not for guest instance use). Furthermore, the operator wishes to use 8 host physical CPU cores and their thread siblings for dedicated guest CPU resources. The remaining 15 host physical CPU cores and their thread siblings will be used for shared guest vCPU usage, with an 8:1 allocation ratio for those physical processors used for shared guest CPU resources.

The operator could configure `nova.conf` like so:

```
[DEFAULT]
cpu_allocation_ratio=8.0

[compute]
cpu_dedicated_set=2-17
cpu_shared_set=18-47
```

The virt driver will construct a provider tree containing a single resource provider representing the compute node and report inventory of `PCPU` and `VCPU` for this single provider accordingly:

```
COMPUTE NODE provider
    PCPU:
        total: 16
        reserved: 0
        min_unit: 1
        max_unit: 16
        step_size: 1
        allocation_ratio: 1.0
    VCPU:
        total: 30
        reserved: 0
        min_unit: 1
        max_unit: 30
        step_size: 1
        allocation_ratio: 8.0
```

Instances using the `dedicated` CPU policy or an explicit `PCPU` resource request, `PCPU` inventory will be consumed. Instances using the `shared` CPU policy, meanwhile, will consume `VCPU` inventory.

Note: `PCPU` and `VCPU` allocations are currently combined to calculate the value for the `cores` quota class.

Configuring Hyper-V compute nodes for instance NUMA policies

Hyper-V is configured by default to allow instances to span multiple NUMA nodes, regardless if the instances have been configured to only span N NUMA nodes. This behaviour allows Hyper-V instances to have up to 64 vCPUs and 1 TB of memory.

Checking NUMA spanning can easily be done by running this following PowerShell command:

```
(Get-VMHost).NumaSpanningEnabled
```

In order to disable this behaviour, the host will have to be configured to disable NUMA spanning. This can be done by executing these following PowerShell commands:

```
Set-VMHost -NumaSpanningEnabled $false
Restart-Service vmms
```

In order to restore this behaviour, execute these PowerShell commands:

```
Set-VMHost -NumaSpanningEnabled $true
Restart-Service vmms
```

The *Virtual Machine Management Service* (*vmms*) is responsible for managing the Hyper-V VMs. The VMs will still run while the service is down or restarting, but they will not be manageable by the `nova-compute` service. In order for the effects of the host NUMA spanning configuration to take effect, the VMs will have to be restarted.

Hyper-V does not allow instances with a NUMA topology to have dynamic memory allocation turned on. The Hyper-V driver will ignore the configured `dynamic_memory_ratio` from the given `nova.conf` file when spawning instances with a NUMA topology.

Huge pages

The huge page feature in OpenStack provides important performance improvements for applications that are highly memory IO-bound.

Note: Huge pages may also be referred to hugepages or large pages, depending on the source. These terms are synonyms.

Pages, the TLB and huge pages

Pages Physical memory is segmented into a series of contiguous regions called pages. Each page contains a number of bytes, referred to as the page size. The system retrieves memory by accessing entire pages, rather than byte by byte.

Translation Lookaside Buffer (TLB) A TLB is used to map the virtual addresses of pages to the physical addresses in actual memory. The TLB is a cache and is not limitless, storing only the most recent or frequently accessed pages. During normal operation, processes will sometimes attempt to retrieve pages that are not stored in the cache. This is known as a TLB miss and results in a delay as the processor iterates through the pages themselves to find the missing address mapping.

Huge Pages The standard page size in x86 systems is 4 kB. This is optimal for general purpose computing but larger page sizes - 2 MB and 1 GB - are also available. These larger page sizes are known as huge pages. Huge pages result in less efficient memory usage as a process will not generally use all memory available in each page. However, use of huge pages will result in fewer overall pages and a reduced risk of TLB misses. For processes that have significant memory requirements or are memory intensive, the benefits of huge pages frequently outweigh the drawbacks.

Persistent Huge Pages On Linux hosts, persistent huge pages are huge pages that are reserved upfront. The HugeTLB provides for the mechanism for this upfront configuration of huge pages. The HugeTLB allows for the allocation of varying quantities of different huge page sizes. Allocation can be made at boot time or run time. Refer to the Linux hugetlbfs guide for more information.

Transparent Huge Pages (THP) On Linux hosts, transparent huge pages are huge pages that are automatically provisioned based on process requests. Transparent huge pages are provisioned on a best effort basis, attempting to provision 2 MB huge pages if available but falling back to 4 kB small pages if not. However, no upfront configuration is necessary. Refer to the Linux THP guide for more information.

Enabling huge pages on the host

Important: Huge pages may not be used on a host configured for file-backed memory. See *File-backed memory* for details

Persistent huge pages are required owing to their guaranteed availability. However, persistent huge pages are not enabled by default in most environments. The steps for enabling huge pages differ from platform to platform and only the steps for Linux hosts are described here. On Linux hosts, the number of persistent huge pages on the host can be queried by checking `/proc/meminfo`:

```
$ grep Huge /proc/meminfo
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

In this instance, there are 0 persistent huge pages (`HugePages_Total`) and 0 transparent huge pages (`AnonHugePages`) allocated. Huge pages can be allocated at boot time or run time. Huge pages require a contiguous area of memory - memory that gets increasingly fragmented the long a host is running. Identifying contiguous areas of memory is an issue for all huge page sizes, but it is particularly problematic for larger huge page sizes such as 1 GB huge pages. Allocating huge pages at boot time will ensure the correct number of huge pages is always available, while allocating them at run time can fail if memory has become too fragmented.

To allocate huge pages at run time, the kernel boot parameters must be extended to include some huge page-specific parameters. This can be achieved by modifying `/etc/default/grub` and appending the `hugepagesz`, `hugepages`, and `transparent_hugepages=never` arguments to `GRUB_CMDLINE_LINUX`. To allocate, for example, 2048 persistent 2 MB huge pages at boot time, run:

```
# echo 'GRUB_CMDLINE_LINUX="$GRUB_CMDLINE_LINUX hugepagesz=2M hugepages=2048␣
↪transparent_hugepage=never"' > /etc/default/grub
$ grep GRUB_CMDLINE_LINUX /etc/default/grub
GRUB_CMDLINE_LINUX="..."
GRUB_CMDLINE_LINUX="$GRUB_CMDLINE_LINUX hugepagesz=2M hugepages=2048 transparent_
↪hugepage=never"
```

Important: Persistent huge pages are not usable by standard host OS processes. Ensure enough free, non-huge page memory is reserved for these processes.

Reboot the host, then validate that huge pages are now available:

```
$ grep "Huge" /proc/meminfo
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
HugePages_Total:    2048
HugePages_Free:     2048
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

There are now 2048 2 MB huge pages totalling 4 GB of huge pages. These huge pages must be mounted. On most platforms, this happens automatically. To verify that the huge pages are mounted, run:

```
# mount | grep huge
hugetlbfs on /dev/hugepages type hugetlbfs (rw)
```

In this instance, the huge pages are mounted at `/dev/hugepages`. This mount point varies from platform to platform. If the above command did not return anything, the hugepages must be mounted manually. To mount the huge pages at `/dev/hugepages`, run:

```
# mkdir -p /dev/hugepages
# mount -t hugetlbfs hugetlbfs /dev/hugepages
```

There are many more ways to configure huge pages, including allocating huge pages at run time, specifying varying allocations for different huge page sizes, or allocating huge pages from memory affinitized to different NUMA nodes. For more information on configuring huge pages on Linux hosts, refer to the Linux hugetlbfs guide.

Customizing instance huge pages allocations

Important: The functionality described below is currently only supported by the libvirt/KVM driver.

Important: For performance reasons, configuring huge pages for an instance will implicitly result in a NUMA topology being configured for the instance. Configuring a NUMA topology for an instance requires enablement of `NUMATopologyFilter`. Refer to *CPU topologies* for more information.

By default, an instance does not use huge pages for its underlying memory. However, huge pages can bring important or required performance improvements for some workloads. Huge pages must be requested explicitly through the use of flavor extra specs or image metadata. To request an instance use huge pages, run:

```
$ openstack flavor set m1.large --property hw:mem_page_size=large
```

Different platforms offer different huge page sizes. For example: x86-based platforms offer 2 MB and 1 GB huge page sizes. Specific huge page sizes can be also be requested, with or without a unit suffix. The unit suffix must be one of: Kb(it), Kib(it), Mb(it), Mib(it), Gb(it), Gib(it), Tb(it), Tib(it), KB, KiB, MB, MiB, GB, GiB, TB, TiB. Where a unit suffix is not provided, Kilobytes are assumed. To request an instance to use 2 MB huge pages, run one of:

```
$ openstack flavor set m1.large --property hw:mem_page_size=2MB
```

```
$ openstack flavor set m1.large --property hw:mem_page_size=2048
```

Enabling huge pages for an instance can have negative consequences for other instances by consuming limited huge pages resources. To explicitly request an instance use small pages, run:

```
$ openstack flavor set m1.large --property hw:mem_page_size=small
```

Note: Explicitly requesting any page size will still result in a NUMA topology being applied to the instance, as described earlier in this document.

Finally, to leave the decision of huge or small pages to the compute driver, run:

```
$ openstack flavor set m1.large --property hw:mem_page_size=any
```

For more information about the syntax for `hw:mem_page_size`, refer to *Manage Flavors*.

Applications are frequently packaged as images. For applications that require the IO performance improvements that huge pages provides, configure image metadata to ensure instances always request the specific page size regardless of flavor. To configure an image to use 1 GB huge pages, run:

```
$ openstack image set [IMAGE_ID]  --property hw_mem_page_size=1GB
```

If the flavor specifies a numerical page size or a page size of small the image is not allowed to specify a page size and if it does an exception will be raised. If the flavor specifies a page size of `any` or `large` then any page size specified in the image will be used. By setting a `small` page size in the flavor, administrators can prevent users requesting huge pages in flavors and impacting resource utilization. To configure this page size, run:

```
$ openstack flavor set m1.large --property hw:mem_page_size=small
```

Note: Explicitly requesting any page size will still result in a NUMA topology being applied to the instance, as described earlier in this document.

For more information about image metadata, refer to the Image metadata guide.

Attaching virtual GPU devices to guests

The virtual GPU feature in Nova allows a deployment to provide specific GPU types for instances using physical GPUs that can provide virtual devices.

For example, a single Intel GVT-g or a NVIDIA GRID vGPU physical Graphics Processing Unit (pGPU) can be virtualized as multiple virtual Graphics Processing Units (vGPUs) if the hypervisor supports the hardware driver and has the capability to create guests using those virtual devices.

This feature is highly dependent on the hypervisor, its version and the physical devices present on the host. In addition, the vendors vGPU driver software must be installed and configured on the host at the same time.

Hypervisor-specific caveats are mentioned in the *Caveats* section.

To enable virtual GPUs, follow the steps below:

1. *Enable GPU types (Compute)*
2. *Configure a flavor (Controller)*

Enable GPU types (Compute)

1. Specify which specific GPU type(s) the instances would get.

   Edit `devices.enabled_vgpu_types`:

   ```
   [devices]
   enabled_vgpu_types = nvidia-35
   ```

   Note: As of the Queens release, Nova only supports a single type. If more than one vGPU type is specified (as a comma-separated list), only the first one will be used.

   To know which specific type(s) to mention, please refer to *How to discover a GPU type*.

2. Restart the `nova-compute` service.

   Warning: Changing the type is possible but since existing physical GPUs cant address multiple guests having different types, that will make Nova return you a NoValidHost if existing instances with the original type still exist. Accordingly, its highly recommended to instead deploy the new type to new compute nodes that dont already have workloads and rebuild instances on the nodes that need to change types.

Configure a flavor (Controller)

Configure a flavor to request one virtual GPU:

```
$ openstack flavor set vgpu_1 --property "resources:VGPU=1"
```

Note: As of the Queens release, all hypervisors that support virtual GPUs only accept a single virtual GPU per instance.

The enabled vGPU types on the compute hosts are not exposed to API users. Flavors configured for vGPU support can be tied to host aggregates as a means to properly schedule those flavors onto the compute hosts that support them. See the *Host Aggregates* for more information.

Create instances with virtual GPU devices

The `nova-scheduler` selects a destination host that has vGPU devices available by calling the Placement API for a specific VGPU resource class provided by compute nodes.

```
$ openstack server create --flavor vgpu_1 --image cirros-0.3.5-x86_64-uec --wait test-
↪vgpu
```

Note: As of the Queens release, only the *FilterScheduler* scheduler driver uses the Placement API.

How to discover a GPU type

Depending on your hypervisor:

- For libvirt, virtual GPUs are seen as mediated devices. Physical PCI devices (the graphic card here) supporting virtual GPUs propose mediated device (mdev) types. Since mediated devices are supported by the Linux kernel through sysfs files after installing the vendors virtual GPUs driver software, you can see the required properties as follows:

```
$ ls /sys/class/mdev_bus/*/mdev_supported_types
/sys/class/mdev_bus/0000:84:00.0/mdev_supported_types:
nvidia-35  nvidia-36  nvidia-37  nvidia-38  nvidia-39  nvidia-40  nvidia-41 ␣
↪nvidia-42  nvidia-43  nvidia-44  nvidia-45

/sys/class/mdev_bus/0000:85:00.0/mdev_supported_types:
nvidia-35  nvidia-36  nvidia-37  nvidia-38  nvidia-39  nvidia-40  nvidia-41 ␣
↪nvidia-42  nvidia-43  nvidia-44  nvidia-45

/sys/class/mdev_bus/0000:86:00.0/mdev_supported_types:
nvidia-35  nvidia-36  nvidia-37  nvidia-38  nvidia-39  nvidia-40  nvidia-41 ␣
↪nvidia-42  nvidia-43  nvidia-44  nvidia-45

/sys/class/mdev_bus/0000:87:00.0/mdev_supported_types:
nvidia-35  nvidia-36  nvidia-37  nvidia-38  nvidia-39  nvidia-40  nvidia-41 ␣
↪nvidia-42  nvidia-43  nvidia-44  nvidia-45
```

- For XenServer, virtual GPU types are created by XenServer at startup depending on the available hardware and config files present in dom0. You can run the command of `xe vgpu-type-list` from dom0 to get the available vGPU types. The value for the field of `model-name ( RO)` : is the vGPU types name which can be used to set the nova config option `[devices]/enabled_vgpu_types`. See the following example:

```
[root@trailblazer-2 ~]# xe vgpu-type-list
uuid ( RO)               : 78d2d963-41d6-4130-8842-aedbc559709f
       vendor-name ( RO): NVIDIA Corporation
        model-name ( RO): GRID M60-8Q
         max-heads ( RO): 4
    max-resolution ( RO): 4096x2160
```

(continues on next page)

(continued from previous page)

```
uuid ( RO)              : a1bb1692-8ce3-4577-a611-6b4b8f35a5c9
       vendor-name ( RO): NVIDIA Corporation
        model-name ( RO): GRID M60-0Q
         max-heads ( RO): 2
    max-resolution ( RO): 2560x1600


uuid ( RO)              : 69d03200-49eb-4002-b661-824aec4fd26f
       vendor-name ( RO): NVIDIA Corporation
        model-name ( RO): GRID M60-2A
         max-heads ( RO): 1
    max-resolution ( RO): 1280x1024


uuid ( RO)              : c58b1007-8b47-4336-95aa-981a5634d03d
       vendor-name ( RO): NVIDIA Corporation
        model-name ( RO): GRID M60-4Q
         max-heads ( RO): 4
    max-resolution ( RO): 4096x2160


uuid ( RO)              : 292a2b20-887f-4a13-b310-98a75c53b61f
       vendor-name ( RO): NVIDIA Corporation
        model-name ( RO): GRID M60-2Q
         max-heads ( RO): 4
    max-resolution ( RO): 4096x2160


uuid ( RO)              : d377db6b-a068-4a98-92a8-f94bd8d6cc5d
       vendor-name ( RO): NVIDIA Corporation
        model-name ( RO): GRID M60-0B
         max-heads ( RO): 2
    max-resolution ( RO): 2560x1600

...
```

Checking allocations and inventories for virtual GPUs

Note: The information below is only valid from the 19.0.0 Stein release and only for the libvirt driver. Before this release or when using the Xen driver, inventories and allocations related to a `VGPU` resource class are still on the root resource provider related to the compute node. If upgrading from Rocky and using the libvirt driver, `VGPU` inventory and allocations are moved to child resource providers that represent actual physical GPUs.

The examples you will see are using the osc-placement plugin for OpenStackClient. For details on specific commands, see its documentation.

1. Get the list of resource providers

```
$ openstack resource provider list
+--------------------------------------+-----------------------------------------
↪--------------+------------+
| uuid                                 | name                                    ␣
↪               | generation |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+-----------------------------------------
↪---------------+------------+
| 5958a366-3cad-416a-a2c9-cfbb5a472287 | virtlab606.xxxxxxxxxxxxxxxxxxxxxxxxx  ␣
↪               |          7 |
| fc9b9287-ef5e-4408-aced-d5577560160c | virtlab606.xxxxxxxxxxxxxxxxxxxxxxxxx_
↪pci_0000_86_00_0 |          2 |
| e2f8607b-0683-4141-a8af-f5e20682e28c | virtlab606.xxxxxxxxxxxxxxxxxxxxxxxxx_
↪pci_0000_85_00_0 |          3 |
| 85dd4837-76f9-41f2-9f19-df386017d8a0 | virtlab606.xxxxxxxxxxxxxxxxxxxxxxxxx_
↪pci_0000_87_00_0 |          2 |
| 7033d860-8d8a-4963-8555-0aa902a08653 | virtlab606.xxxxxxxxxxxxxxxxxxxxxxxxx_
↪pci_0000_84_00_0 |          2 |
+--------------------------------------+-----------------------------------------
↪---------------+------------+
```

In this example, we see the root resource provider `5958a366-3cad-416a-a2c9-cfbb5a472287` with four other resource providers that are its children and where each of them corresponds to a single physical GPU.

2. Check the inventory of each resource provider to see resource classes

```
$ openstack resource provider inventory list 5958a366-3cad-416a-a2c9-cfbb5a472287
+----------------+------------------+----------+----------+-----------+----------
↪+-------+
| resource_class | allocation_ratio | max_unit | reserved | step_size | min_unit␣
↪| total |
+----------------+------------------+----------+----------+-----------+----------
↪+-------+
| VCPU           |             16.0 |       48 |        0 |         1 |        1␣
↪|    48 |
| MEMORY_MB      |              1.5 |    65442 |      512 |         1 |        1␣
↪| 65442 |
| DISK_GB        |              1.0 |       49 |        0 |         1 |        1␣
↪|    49 |
+----------------+------------------+----------+----------+-----------+----------
↪+-------+
$ openstack resource provider inventory list e2f8607b-0683-4141-a8af-f5e20682e28c
+----------------+------------------+----------+----------+-----------+----------
↪+-------+
| resource_class | allocation_ratio | max_unit | reserved | step_size | min_unit␣
↪| total |
+----------------+------------------+----------+----------+-----------+----------
↪+-------+
| VGPU           |              1.0 |       16 |        0 |         1 |        1␣
↪|    16 |
+----------------+------------------+----------+----------+-----------+----------
↪+-------+
```

Here you can see a `VGPU` inventory on the child resource provider while other resource class inventories are still located on the root resource provider.

3. Check allocations for each server that is using virtual GPUs

```
$ openstack server list
+--------------------------------------+-------+--------+-------------------------
↪-------------------------------+--------------------------+--------+
| ID                                   | Name  | Status | Networks                ␣
↪                              | Image                    | Flavor |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+-------+--------+-------------------------
↪-------------------------------+--------------------------+--------+
| 5294f726-33d5-472a-bef1-9e19bb41626d | vgpu2 | ACTIVE | private=10.0.0.14,␣
↪fd45:cdad:c431:0:f816:3eff:fe78:a748 | cirros-0.4.0-x86_64-disk | vgpu   |
| a6811fc2-cec8-4f1d-baea-e2c6339a9697 | vgpu1 | ACTIVE | private=10.0.0.34,␣
↪fd45:cdad:c431:0:f816:3eff:fe54:cc8f | cirros-0.4.0-x86_64-disk | vgpu   |
+--------------------------------------+-------+--------+-------------------------
↪-------------------------------+--------------------------+--------+

$ openstack resource provider allocation show 5294f726-33d5-472a-bef1-9e19bb41626d
+--------------------------------------+------------+------------------------------
↪--------------------+
| resource_provider                    | generation | resources                    ␣
↪                    |
+--------------------------------------+------------+------------------------------
↪--------------------+
| 5958a366-3cad-416a-a2c9-cfbb5a472287 |          8 | {u'VCPU': 1, u'MEMORY_MB':␣
↪512, u'DISK_GB': 1} |
| 7033d860-8d8a-4963-8555-0aa902a08653 |          3 | {u'VGPU': 1}                 ␣
↪                    |
+--------------------------------------+------------+------------------------------
↪--------------------+

$ openstack resource provider allocation show a6811fc2-cec8-4f1d-baea-e2c6339a9697
+--------------------------------------+------------+------------------------------
↪--------------------+
| resource_provider                    | generation | resources                    ␣
↪                    |
+--------------------------------------+------------+------------------------------
↪--------------------+
| e2f8607b-0683-4141-a8af-f5e20682e28c |          3 | {u'VGPU': 1}                 ␣
↪                    |
| 5958a366-3cad-416a-a2c9-cfbb5a472287 |          8 | {u'VCPU': 1, u'MEMORY_MB':␣
↪512, u'DISK_GB': 1} |
+--------------------------------------+------------+------------------------------
↪--------------------+
```

In this example, two servers were created using a flavor asking for 1 `VGPU`, so when looking at the allocations for each consumer UUID (which is the server UUID), you can see that VGPU allocation is against the child resource provider while other allocations are for the root resource provider. Here, that means that the virtual GPU used by `a6811fc2-cec8-4f1d-baea-e2c6339a9697` is actually provided by the physical GPU having the PCI ID `0000:85:00.0`.

Caveats

Note: This information is correct as of the 17.0.0 Queens release. Where improvements have been made or issues fixed, they are noted per item.

For libvirt:

- Suspending a guest that has vGPUs doesnt yet work because of a libvirt limitation (it cant hot-unplug mediated devices from a guest). Workarounds using other instance actions (like snapshotting the instance or shelving it) are recommended until libvirt gains mdev hot-unplug support. If a user attempts to suspend the instance, the

libvirt driver will raise an exception that will cause the instance to be set back to ACTIVE. The `suspend` action in the `os-instance-actions` API will have an *Error* state.

- Resizing an instance with a new flavor that has vGPU resources doesnt allocate those vGPUs to the instance (the instance is created without vGPU resources). The proposed workaround is to rebuild the instance after resizing it. The rebuild operation allocates vGPUS to the instance.
- Cold migrating an instance to another host will have the same problem as resize. If you want to migrate an instance, make sure to rebuild it after the migration.
- Rescue images do not use vGPUs. An instance being rescued does not keep its vGPUs during rescue. During that time, another instance can receive those vGPUs. This is a known issue. The recommended workaround is to rebuild an instance immediately after rescue. However, rebuilding the rescued instance only helps if there are other free vGPUs on the host.

Note: This has been resolved in the Rocky release[1].

For XenServer:

- Suspend and live migration with vGPUs attached depends on support from the underlying XenServer version. Please see XenServer release notes for up to date information on when a hypervisor supporting live migration and suspend/resume with vGPUs is available. If a suspend or live migrate operation is attempted with a XenServer version that does not support that operation, an internal exception will occur that will cause nova setting the instance to be in ERROR status. You can use the command of `openstack server set --state active <server>` to set it back to ACTIVE.
- Resizing an instance with a new flavor that has vGPU resources doesnt allocate those vGPUs to the instance (the instance is created without vGPU resources). The proposed workaround is to rebuild the instance after resizing it. The rebuild operation allocates vGPUS to the instance.
- Cold migrating an instance to another host will have the same problem as resize. If you want to migrate an instance, make sure to rebuild it after the migration.

File-backed memory

Important: As of the 18.0.0 Rocky release, the functionality described below is only supported by the libvirt/KVM driver.

The file-backed memory feature in Openstack allows a Nova node to serve guest memory from a file backing store. This mechanism uses the libvirt file memory source, causing guest instance memory to be allocated as files within the libvirt memory backing directory.

Since instance performance will be related to the speed of the backing store, this feature works best when used with very fast block devices or virtual file systems - such as flash or RAM devices.

When configured, `nova-compute` will report the capacity configured for file-backed memory to placement in place of the total system memory capacity. This allows the node to run more instances than would normally fit within system memory.

When available in libvirt and qemu, instance memory will be discarded by qemu at shutdown by calling `madvise(MADV_REMOVE)`, to avoid flushing any dirty memory to the backing store on exit.

To enable file-backed memory, follow the steps below:

1. *Configure the backing store*

[1] https://bugs.launchpad.net/nova/+bug/1762688

2. *Configure Nova Compute for file-backed memory*

Important: It is not possible to live migrate from a node running a version of OpenStack that does not support file-backed memory to a node with file backed memory enabled. It is recommended that all Nova compute nodes are upgraded to Rocky before enabling file-backed memory.

Prerequisites and Limitations

Libvirt File-backed memory requires libvirt version 4.0.0 or newer. Discard capability requires libvirt version 4.4.0 or newer.

Qemu File-backed memory requires qemu version 2.6.0 or newer.Discard capability requires qemu version 2.10.0 or newer.

Memory overcommit File-backed memory is not compatible with memory overcommit. `ram_allocation_ratio` must be set to `1.0` in `nova.conf`, and the host must not be added to a host aggregate with `ram_allocation_ratio` set to anything but `1.0`.

Huge pages File-backed memory is not compatible with huge pages. Instances with huge pages configured will not start on a host with file-backed memory enabled. It is recommended to use host aggregates to ensure instances configured for huge pages are not placed on hosts with file-backed memory configured.

Handling these limitations could be optimized with a scheduler filter in the future.

Configure the backing store

Note: `/dev/sdb` and the `ext4` filesystem are used here as an example. This will differ between environments.

Note: `/var/lib/libvirt/qemu/ram` is the default location. The value can be set via `memory_backing_dir` in `/etc/libvirt/qemu.conf`, and the mountpoint must match the value configured there.

By default, Libvirt with qemu/KVM allocates memory within `/var/lib/libvirt/qemu/ram/`. To utilize this, you need to have the backing store mounted at (or above) this location.

1. Create a filesystem on the backing device

```
# mkfs.ext4 /dev/sdb
```

2. Mount the backing device

   Add the backing device to `/etc/fstab` for automatic mounting to `/var/lib/libvirt/qemu/ram`

   Mount the device

```
# mount /dev/sdb /var/lib/libvirt/qemu/ram
```

Configure Nova Compute for file-backed memory

1. Enable File-backed memory in `nova-compute`

Configure Nova to utilize file-backed memory with the capacity of the backing store in MiB. 1048576 MiB (1 TiB) is used in this example.

Edit `/etc/nova/nova.conf`

```
[libvirt]
file_backed_memory=1048576
```

2. Restart the `nova-compute` service

Using ports with resource request

Starting from microversion 2.72 nova supports creating servers with neutron ports having resource request visible as a admin-only port attribute `resource_request`. For example a neutron port has resource request if it has a QoS minimum bandwidth rule attached.

The Quality of Service (QoS): Guaranteed Bandwidth document describes how to configure neutron to use this feature.

Resource allocation

Nova collects and combines the resource request from each port in a boot request and sends one allocation candidate request to placement during scheduling so placement will make sure that the resource request of the ports are fulfilled. At the end of the scheduling nova allocates one candidate in placement. Therefore the requested resources for each port from a single boot request will be allocated under the servers allocation in placement.

Resource Group policy

Nova represents the resource request of each neutron port as a separate Granular Resource Request group when querying placement for allocation candidates. When a server create request includes more than one port with resource requests then more than one group will be used in the allocation candidate query. In this case placement requires to define the `group_policy`. Today it is only possible via the `group_policy` key of the flavor extra_spec. The possible values are `isolate` and `none`.

When the policy is set to `isolate` then each request group and therefore the resource request of each neutron port will be fulfilled from separate resource providers. In case of neutron ports with `vnic_type=direct` or `vnic_type=macvtap` this means that each port will use a virtual function from different physical functions.

When the policy is set to `none` then the resource request of the neutron ports can be fulfilled from overlapping resource providers. In case of neutron ports with `vnic_type=direct` or `vnic_type=macvtap` this means the ports may use virtual functions from the same physical function.

For neutron ports with `vnic_type=normal` the group policy defines the collocation policy on OVS bridge level so `group_policy=none` is a reasonable default value in this case.

If the `group_policy` is missing from the flavor then the server create request will fail with No valid host was found and a warning describing the missing policy will be logged.

Virt driver support

Supporting neutron ports with `vnic_type=direct` or `vnic_type=macvtap` depends on the capability of the virt driver. For the supported virt drivers see the Support matrix

If the virt driver on the compute host does not support the needed capability then the PCI claim will fail on the host and re-schedule will be triggered. It is suggested not to configure bandwidth inventory in the neutron agents on these compute hosts to avoid unnecessary reschedule.

Attaching virtual persistent memory to guests

New in version 20.0.0: (Train)

Starting in the 20.0.0 (Train) release, the virtual persistent memory (vPMEM) feature in Nova allows a deployment using the libvirt compute driver to provide vPMEMs for instances using physical persistent memory (PMEM) that can provide virtual devices.

PMEM must be partitioned into PMEM namespaces for applications to use. This vPMEM feature only uses PMEM namespaces in `devdax` mode as QEMU vPMEM backends. If you want to dive into related notions, the document NVDIMM Linux kernel document is recommended.

To enable vPMEMs, follow the steps below.

Dependencies

The following are required to support the vPMEM feature:

- Persistent Memory Hardware

  One such product is Intelő Optane DC Persistent Memory. ipmctl is used to configure it.

- Linux Kernel version >= 4.18 with the following modules loaded:

  `dax_pmem`, `nd_pmem`, `device_dax`, `nd_btt`

Note: NVDIMM support is present in the Linux Kernel v4.0 or newer. It is recommended to use Kernel version 4.2 or later since NVDIMM support is enabled by default. We met some bugs in older versions, and we have done all verification works with OpenStack on 4.18 version, so 4.18 version and newer will probably guarantee its functionality.

- QEMU version >= 3.1.0
- Libvirt version >= 5.0.0
- ndctl version >= 62
- daxio version >= 1.6

The vPMEM feature has been verified under the software and hardware listed above.

Configure PMEM namespaces (Compute)

1. Create PMEM namespaces as vPMEM backends using the ndctl utility.

   For example, to create a 30GiB namespace named `ns3`:

```
$ sudo ndctl create-namespace -s 30G -m devdax -M mem -n ns3
{
  "dev":"namespace1.0",
  "mode":"devdax",
  "map":"mem",
  "size":"30.00 GiB (32.21 GB)",
```

(continues on next page)

(continued from previous page)

```
  "uuid":"937e9269-512b-4f65-9ac6-b74b61075c11",
  "raw_uuid":"17760832-a062-4aef-9d3b-95ea32038066",
  "daxregion":{
    "id":1,
    "size":"30.00 GiB (32.21 GB)",
    "align":2097152,
    "devices":[
    {
      "chardev":"dax1.0",
      "size":"30.00 GiB (32.21 GB)"
    }
    ]
  },
  "name":"ns3",
  "numa_node":1
}
```

Then list the available PMEM namespaces on the host:

```
$ ndctl list -X
[
  {
    ...
    "size":6440353792,
    ...
    "name":"ns0",
    ...
  },
  {
    ...
    "size":6440353792,
    ...
    "name":"ns1",
    ...
  },
  {
    ...
    "size":6440353792,
    ...
    "name":"ns2",
    ...
  },
  {
    ...
    "size":32210157568,
    ...
    "name":"ns3",
    ...
  }
]
```

2. Specify which PMEM namespaces should be available to instances.

   Edit *libvirt.pmem_namespaces*:

```
[libvirt]
# pmem_namespaces=$LABEL:$NSNAME[|$NSNAME][,$LABEL:$NSNAME[|$NSNAME]]
```

(continues on next page)

(continued from previous page)

```
pmem_namespaces = 6GB:ns0|ns1|ns2,LARGE:ns3
```

Configured PMEM namespaces must have already been created on the host as described above. The conf syntax allows the admin to associate one or more namespace `$NSNAME`s with an arbitrary `$LABEL` that can subsequently be used in a flavor to request one of those namespaces. It is recommended, but not required, for namespaces under a single `$LABEL` to be the same size.

3. Restart the `nova-compute` service.

   Nova will invoke ndctl to identify the configured PMEM namespaces, and report vPMEM resources to placement.

Configure a flavor

Specify a comma-separated list of the `$LABEL`s from *`libvirt.pmem_namespaces`* to the flavors `hw:pmem` property. Note that multiple instances of the same label are permitted:

```
$ openstack flavor set --property hw:pmem='6GB' my_flavor
$ openstack flavor set --property hw:pmem='6GB,LARGE' my_flavor_large
$ openstack flavor set --property hw:pmem='6GB,6GB' m1.medium
```

Note: If a NUMA topology is specified, all vPMEM devices will be put on guest NUMA node 0; otherwise nova will generate one NUMA node automatically for the guest.

Based on the above examples, an `openstack server create` request with `my_flavor_large` will spawn an instance with two vPMEMs. One, corresponding to the `LARGE` label, will be `ns3`; the other, corresponding to the `6G` label, will be arbitrarily chosen from `ns0`, `ns1`, or `ns2`.

Note: Using vPMEM inside a virtual machine requires the following:

- Guest kernel version 4.18 or higher;
- The `dax_pmem`, `nd_pmem`, `device_dax`, and `nd_btt` kernel modules;
- The ndctl utility.

Note: When resizing an instance with vPMEMs, the vPMEM data wont be migrated.

Verify inventories and allocations

This section describes how to check that:

- vPMEM inventories were created correctly in placement, validating the *configuration described above*.
- allocations were created correctly in placement for instances spawned from *flavors configured with vPMEMs*.

Note: Inventories and allocations related to vPMEM resource classes are on the root resource provider related to the compute node.

---

1. Get the list of resource providers

```
$ openstack resource provider list
+--------------------------------------+--------+------------+
| uuid                                 | name   | generation |
+--------------------------------------+--------+------------+
| 1bc545f9-891f-4930-ab2b-88a56078f4be | host-1 |         47 |
| 7d994aef-680d-43d4-9325-a67c807e648e | host-2 |         67 |
+--------------------------------------+--------+------------+
```

2. Check the inventory of each resource provider to see resource classes

   Each `$LABEL` configured in *`libvirt.pmem_namespaces`* is used to generate a resource class named `CUSTOM_PMEM_NAMESPACE_$LABEL`. Nova will report to Placement the number of vPMEM namespaces configured for each `$LABEL`. For example, assuming `host-1` was configured as described above:

```
$ openstack resource provider inventory list 1bc545f9-891f-4930-ab2b-88a56078f4be
+-----------------------------+------------------+----------+----------+----------
↪-+----------+--------+
| resource_class              | allocation_ratio | max_unit | reserved | step_
↪size | min_unit |  total |
+-----------------------------+------------------+----------+----------+----------
↪-+----------+--------+
| VCPU                        |             16.0 |       64 |        0 |        ␣
↪1 |        1 |     64 |
| MEMORY_MB                   |              1.5 |   190604 |      512 |        ␣
↪1 |        1 | 190604 |
| CUSTOM_PMEM_NAMESPACE_LARGE |              1.0 |        1 |        0 |        ␣
↪1 |        1 |      1 |
| CUSTOM_PMEM_NAMESPACE_6GB   |              1.0 |        3 |        0 |        ␣
↪1 |        1 |      3 |
| DISK_GB                     |              1.0 |      439 |        0 |        ␣
↪1 |        1 |    439 |
+-----------------------------+------------------+----------+----------+----------
↪-+----------+--------+
```

   Here you can see the vPMEM resource classes prefixed with `CUSTOM_PMEM_NAMESPACE_`. The `LARGE` label was configured with one namespace (`ns3`), so it has an inventory of `1`. Since the `6GB` label was configured with three namespaces (`ns0`, `ns1`, and `ns2`), the `CUSTOM_PMEM_NAMESPACE_6GB` inventory has a `total` and `max_unit` of `3`.

3. Check allocations for each server that is using vPMEMs

```
$ openstack server list
+--------------------------------------+----------------------+--------+----------
↪---------+---------------+-----------------+
| ID                                   | Name                 | Status | Networks␣
↪          | Image         | Flavor          |
+--------------------------------------+----------------------+--------+----------
↪---------+---------------+-----------------+
| 41d3e139-de5c-40fd-9d82-016b72f2ba1d | server-with-2-vpmems | ACTIVE |␣
↪private=10.0.0.24 | ubuntu-bionic | my_flavor_large |
| a616a7f6-b285-4adf-a885-dd8426dd9e6a | server-with-1-vpmem  | ACTIVE |␣
↪private=10.0.0.13 | ubuntu-bionic | my_flavor       |
+--------------------------------------+----------------------+--------+----------
↪---------+---------------+-----------------+

$ openstack resource provider allocation show 41d3e139-de5c-40fd-9d82-016b72f2ba1d
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+------------+------------------------------
↪------------------------------------------------------------------------------
↪-----------+
| resource_provider                    | generation | resources                    ␣
↪                                                                              ␣
↪           |
+--------------------------------------+------------+------------------------------
↪------------------------------------------------------------------------------
↪-----------+
| 1bc545f9-891f-4930-ab2b-88a56078f4be |         49 | {u'MEMORY_MB': 32768, u'VCPU
↪': 16, u'DISK_GB': 20, u'CUSTOM_PMEM_NAMESPACE_6GB': 1, u'CUSTOM_PMEM_NAMESPACE_
↪LARGE': 1} |
+--------------------------------------+------------+------------------------------
↪------------------------------------------------------------------------------
↪-----------+

$ openstack resource provider allocation show a616a7f6-b285-4adf-a885-dd8426dd9e6a
+--------------------------------------+------------+------------------------------
↪-----------------------------------------------------+
| resource_provider                    | generation | resources                    ␣
↪                                                     |
+--------------------------------------+------------+------------------------------
↪-----------------------------------------------------+
| 1bc545f9-891f-4930-ab2b-88a56078f4be |         49 | {u'MEMORY_MB': 8192, u'VCPU
↪': 8, u'DISK_GB': 20, u'CUSTOM_PMEM_NAMESPACE_6GB': 1} |
+--------------------------------------+------------+------------------------------
↪-----------------------------------------------------+
```

In this example, two servers were created. `server-with-2-vpmems` used `my_flavor_large` asking for one `6GB` vPMEM and one `LARGE` vPMEM. `server-with-1-vpmem` used `my_flavor` asking for a single `6GB` vPMEM.

4.3.3 System architecture

OpenStack Compute contains several main components.

- The cloud controller represents the global state and interacts with the other components. The `API server` acts as the web services front end for the cloud controller. The `compute controller` provides compute server resources and usually also contains the Compute service.
- The `object store` is an optional component that provides storage services; you can also use OpenStack Object Storage instead.
- An `auth manager` provides authentication and authorization services when used with the Compute system; you can also use OpenStack Identity as a separate authentication service instead.
- A `volume controller` provides fast and permanent block-level storage for the compute servers.
- The `network controller` provides virtual networks to enable compute servers to interact with each other and with the public network. You can also use OpenStack Networking instead.
- The `scheduler` is used to select the most suitable compute controller to host an instance.

Compute uses a messaging-based, `shared nothing` architecture. All major components exist on multiple servers, including the compute, volume, and network controllers, and the Object Storage or Image service. The state of the entire system is stored in a database. The cloud controller communicates with the internal object store using HTTP, but it communicates with the scheduler, network controller, and volume controller using Advanced Message Queuing

Protocol (AMQP). To avoid blocking a component while waiting for a response, Compute uses asynchronous calls, with a callback that is triggered when a response is received.

Hypervisors

Compute controls hypervisors through an API server. Selecting the best hypervisor to use can be difficult, and you must take budget, resource constraints, supported features, and required technical specifications into account. However, the majority of OpenStack development is done on systems using KVM and Xen-based hypervisors. For a detailed list of features and support across different hypervisors, see *Feature Support Matrix*.

You can also orchestrate clouds using multiple hypervisors in different availability zones. Compute supports the following hypervisors:

- Baremetal
- Hyper-V
- Kernel-based Virtual Machine (KVM)
- Linux Containers (LXC)
- PowerVM
- Quick Emulator (QEMU)
- User Mode Linux (UML)
- Virtuozzo
- VMware vSphere
- Xen (using libvirt)
- XenServer
- zVM

For more information about hypervisors, see *Hypervisors* section in the Nova Configuration Reference.

Projects, users, and roles

The Compute system is designed to be used by different consumers in the form of projects on a shared system, and role-based access assignments. Roles control the actions that a user is allowed to perform.

Projects are isolated resource containers that form the principal organizational structure within the Compute service. They consist of an individual VLAN, and volumes, instances, images, keys, and users. A user can specify the project by appending `project_id` to their access key. If no project is specified in the API request, Compute attempts to use a project with the same ID as the user.

For projects, you can use quota controls to limit the:

- Number of volumes that can be launched.
- Number of processor cores and the amount of RAM that can be allocated.
- Floating IP addresses assigned to any instance when it launches. This allows instances to have the same publicly accessible IP addresses.
- Fixed IP addresses assigned to the same instance when it launches. This allows instances to have the same publicly or privately accessible IP addresses.

Roles control the actions a user is allowed to perform. By default, most actions do not require a particular role, but you can configure them by editing the `policy.json` file for user roles. For example, a rule can be defined so that a user must have the `admin` role in order to be able to allocate a public IP address.

A project limits users access to particular images. Each user is assigned a user name and password. Keypairs granting access to an instance are enabled for each user, but quotas are set, so that each project can control resource consumption across available hardware resources.

Note: Earlier versions of OpenStack used the term `tenant` instead of `project`. Because of this legacy terminology, some command-line tools use `--tenant_id` where you would normally expect to enter a project ID.

Block storage

OpenStack provides two classes of block storage: ephemeral storage and persistent volume.

Ephemeral storage

Ephemeral storage includes a root ephemeral volume and an additional ephemeral volume.

The root disk is associated with an instance, and exists only for the life of this very instance. Generally, it is used to store an instances root file system, persists across the guest operating system reboots, and is removed on an instance deletion. The amount of the root ephemeral volume is defined by the flavor of an instance.

In addition to the ephemeral root volume, all default types of flavors, except `m1.tiny`, which is the smallest one, provide an additional ephemeral block device sized between 20 and 160 GB (a configurable value to suit an environment). It is represented as a raw block device with no partition table or file system. A cloud-aware operating system can discover, format, and mount such a storage device. OpenStack Compute defines the default file system for different operating systems as Ext4 for Linux distributions, VFAT for non-Linux and non-Windows operating systems, and NTFS for Windows. However, it is possible to specify any other filesystem type by using `virt_mkfs` or `default_ephemeral_format` configuration options.

Note: For example, the `cloud-init` package included into an Ubuntus stock cloud image, by default, formats this space as an Ext4 file system and mounts it on `/mnt`. This is a cloud-init feature, and is not an OpenStack mechanism. OpenStack only provisions the raw storage.

Persistent volume

A persistent volume is represented by a persistent virtualized block device independent of any particular instance, and provided by OpenStack Block Storage.

Only a single configured instance can access a persistent volume. Multiple instances cannot access a persistent volume. This type of configuration requires a traditional network file system to allow multiple instances accessing the persistent volume. It also requires a traditional network file system like NFS, CIFS, or a cluster file system such as GlusterFS. These systems can be built within an OpenStack cluster, or provisioned outside of it, but OpenStack software does not provide these features.

You can configure a persistent volume as bootable and use it to provide a persistent virtual instance similar to the traditional non-cloud-based virtualization system. It is still possible for the resulting instance to keep ephemeral storage, depending on the flavor selected. In this case, the root file system can be on the persistent volume, and its state is maintained, even if the instance is shut down. For more information about this type of configuration, see Introduction to the Block Storage service.

Note: A persistent volume does not provide concurrent access from multiple instances. That type of configuration requires a traditional network file system like NFS, or CIFS, or a cluster file system such as GlusterFS. These systems can be built within an OpenStack cluster, or provisioned outside of it, but OpenStack software does not provide these features.

Building blocks

In OpenStack the base operating system is usually copied from an image stored in the OpenStack Image service. This is the most common case and results in an ephemeral instance that starts from a known template state and loses all accumulated states on virtual machine deletion. It is also possible to put an operating system on a persistent volume in the OpenStack Block Storage volume system. This gives a more traditional persistent system that accumulates states which are preserved on the OpenStack Block Storage volume across the deletion and re-creation of the virtual machine. To get a list of available images on your system, run:

```
$ openstack image list
+--------------------------------------+-----------------------------+--------+
| ID                                   | Name                        | Status |
+--------------------------------------+-----------------------------+--------+
| aee1d242-730f-431f-88c1-87630c0f07ba | Ubuntu 14.04 cloudimg amd64 | active |
| 0b27baa1-0ca6-49a7-b3f4-48388e440245 | Ubuntu 14.10 cloudimg amd64 | active |
| df8d56fc-9cea-4dfd-a8d3-28764de3cb08 | jenkins                     | active |
+--------------------------------------+-----------------------------+--------+
```

The displayed image attributes are:

ID Automatically generated UUID of the image

Name Free form, human-readable name for image

Status The status of the image. Images marked `ACTIVE` are available for use.

Server For images that are created as snapshots of running instances, this is the UUID of the instance the snapshot derives from. For uploaded images, this field is blank.

Virtual hardware templates are called `flavors`. By default, these are configurable by admin users, however that behavior can be changed by redefining the access controls for `compute_extension:flavormanage` in `/etc/nova/policy.json` on the `compute-api` server. For more information, refer to *Nova Policies*.

For a list of flavors that are available on your system:

```
$ openstack flavor list
+-----+-----------+-------+------+-----------+-------+-----------+
| ID  | Name      |   RAM | Disk | Ephemeral | VCPUs | Is_Public |
+-----+-----------+-------+------+-----------+-------+-----------+
| 1   | m1.tiny   |   512 |    1 |         0 |     1 | True      |
| 2   | m1.small  |  2048 |   20 |         0 |     1 | True      |
| 3   | m1.medium |  4096 |   40 |         0 |     2 | True      |
| 4   | m1.large  |  8192 |   80 |         0 |     4 | True      |
| 5   | m1.xlarge | 16384 |  160 |         0 |     8 | True      |
+-----+-----------+-------+------+-----------+-------+-----------+
```

Compute service architecture

These basic categories describe the service architecture and information about the cloud controller.

API server

At the heart of the cloud framework is an API server, which makes command and control of the hypervisor, storage, and networking programmatically available to users.

The API endpoints are basic HTTP web services which handle authentication, authorization, and basic command and control functions using various API interfaces under the Amazon, Rackspace, and related models. This enables API compatibility with multiple existing tool sets created for interaction with offerings from other vendors. This broad compatibility prevents vendor lock-in.

Message queue

A messaging queue brokers the interaction between compute nodes (processing), the networking controllers (software which controls network infrastructure), API endpoints, the scheduler (determines which physical hardware to allocate to a virtual resource), and similar components. Communication to and from the cloud controller is handled by HTTP requests through multiple API endpoints.

A typical message passing event begins with the API server receiving a request from a user. The API server authenticates the user and ensures that they are permitted to issue the subject command. The availability of objects implicated in the request is evaluated and, if available, the request is routed to the queuing engine for the relevant workers. Workers continually listen to the queue based on their role, and occasionally their type host name. When an applicable work request arrives on the queue, the worker takes assignment of the task and begins executing it. Upon completion, a response is dispatched to the queue which is received by the API server and relayed to the originating user. Database entries are queried, added, or removed as necessary during the process.

Compute worker

Compute workers manage computing instances on host machines. The API dispatches commands to compute workers to complete these tasks:

- Run instances
- Delete instances (Terminate instances)
- Reboot instances
- Attach volumes
- Detach volumes
- Get console output

Network Controller

The Network Controller manages the networking resources on host machines. The API server dispatches commands through the message queue, which are subsequently processed by Network Controllers. Specific operations include:

- Allocating fixed IP addresses
- Configuring VLANs for projects
- Configuring networks for compute nodes

4.3.4 Select hosts where instances are launched

With the appropriate permissions, you can select which host instances are launched on and which roles can boot instances on this host.

Starting with the 2.74 microversion, there are two ways to specify a host and/or node when creating a server.

Using Explicit Host and/or Node

We can create servers by using explicit host and/or node. When we use this way to request where instances are launched, we will still execute scheduler filters on the requested destination.

- To select the host where instances are launched, use the `--host HOST` and/or `--hypervisor-hostname HYPERVISOR` options on the **`openstack server create`** command.

  For example:

```
$ openstack --os-compute-api-version 2.74 server create --image IMAGE \
--flavor m1.tiny --key-name KEY --host HOST \
--hypervisor-hostname HYPERVISOR --nic net-id=UUID SERVER
```

- To specify which roles can launch an instance on a specified host, enable the `compute:servers:create:requested_destination` rule in the `policy.json` file. By default, this rule is enabled for only the admin role. If you see `Forbidden (HTTP 403)` in the response, then you are not using the required credentials.
- To view the list of valid compute hosts and nodes, you can follow *Finding Host and Node Names*.

[Legacy] Using Host and/or Node with Availability Zone

We can create servers by using host and/or node with availability zone. When we use this way to select hosts where instances are launched, we will not run the scheduler filters.

- To select the host where instances are launched, use the `--availability-zone ZONE:HOST:NODE` parameter on the **`openstack server create`** command.

  For example:

```
$ openstack server create --image IMAGE --flavor m1.tiny --key-name KEY \
--availability-zone ZONE:HOST:NODE --nic net-id=UUID SERVER
```

Note: HOST and NODE are optional parameters. In such cases, use the `--availability-zone ZONE::NODE`, `--availability-zone ZONE:HOST` or `--availability-zone ZONE`.

- To specify which roles can launch an instance on a specified host, enable the `os_compute_api:servers:create:forced_host` rule in the `policy.json` file. By default, this rule is enabled for only the admin role. If you see `Forbidden (HTTP 403)` in return, then you are not using the required credentials.
- To view the list of valid zones, use the **`openstack availability zone list --compute`** command.

```
$ openstack availability zone list --compute
+-----------+-------------+
| Zone Name | Zone Status |
```

(continues on next page)

(continued from previous page)

```
+-----------+-------------+
| zone1     | available   |
| zone2     | available   |
+-----------+-------------+
```

- To view the list of valid compute hosts and nodes, you can follow *Finding Host and Node Names*.

Finding Host and Node Names

- To view the list of valid compute hosts, use the **`openstack compute service list --service nova-compute`** command.

```
$ openstack compute service list --service nova-compute
+----+--------------+---------------+------+---------+-------+--------------------
↪--------+
| ID | Binary       | Host          | Zone | Status  | State | Updated At
↪        |
+----+--------------+---------------+------+---------+-------+--------------------
↪--------+
| 10 | nova-compute | compute01     | nova | enabled | up    | 2019-07-
↪09T03:59:19.000000 |
| 11 | nova-compute | compute02     | nova | enabled | up    | 2019-07-
↪09T03:59:19.000000 |
| 12 | nova-compute | compute03     | nova | enabled | up    | 2019-07-
↪09T03:59:19.000000 |
+----+--------------+---------------+------+---------+-------+--------------------
↪--------+
```

- To view the list of valid compute nodes, use the **`openstack hypervisor list`** command.

```
$ openstack hypervisor list
+----+---------------------+-----------------+---------------+-------+
| ID | Hypervisor Hostname | Hypervisor Type | Host IP       | State |
+----+---------------------+-----------------+---------------+-------+
|  6 | compute01           | QEMU            | 172.16.50.100 | up    |
|  7 | compute02           | QEMU            | 172.16.50.101 | up    |
|  8 | compute03           | QEMU            | 172.16.50.102 | up    |
+----+---------------------+-----------------+---------------+-------+
```

4.3.5 CellsV2 Management

This section describes the various recommended practices/tips for runnning and maintaining CellsV2 for admins and operators. For more details regarding the basic concept of CellsV2 and its layout please see the main *Cells Layout (v2)* page.

Handling cell failures

For an explanation on how `nova-api` handles cell failures please see the Handling Down Cells section of the Compute API guide. Below, you can find some recommended practices and considerations for effectively tolerating cell failure situations.

Configuration considerations

Since a cell being reachable or not is determined through timeouts, it is suggested to provide suitable values for the following settings based on your requirements.

1. *database.max_retries* is 10 by default meaning every time a cell becomes unreachable, it would retry 10 times before nova can declare the cell as a down cell.
2. *database.retry_interval* is 10 seconds and *oslo_messaging_rabbit.rabbit_retry_interval* is 1 second by default meaning every time a cell becomes unreachable it would retry every 10 seconds or 1 second depending on if its a database or a message queue problem.
3. Nova also has a timeout value called `CELL_TIMEOUT` which is hardcoded to 60 seconds and that is the total time the nova-api would wait before returning partial results for the down cells.

The values of the above settings will affect the time required for nova to decide if a cell is unreachable and then take the necessary actions like returning partial results.

The operator can also control the results of certain actions like listing servers and services depending on the value of the *api.list_records_by_skipping_down_cells* config option. If this is true, the results from the unreachable cells will be skipped and if it is false, the request will just fail with an API error in situations where partial constructs cannot be computed.

Disabling down cells

While the temporary outage in the infrastructure is being fixed, the affected cells can be disabled so that they are removed from being scheduling candidates. To enable or disable a cell, use **`nova-manage cell_v2 update_cell --cell_uuid <cell_uuid> --disable`**. See the *Nova Cells v2* man page for details on command usage.

Known issues

1. **Services and Performance:** In case a cell is down during the startup of nova services, there is the chance that the services hang because of not being able to connect to all the cell databases that might be required for certain calculations and initializations. An example scenario of this situation is if *upgrade_levels.compute* is set to `auto` then the `nova-api` service hangs on startup if there is at least one unreachable cell. This is because it needs to connect to all the cells to gather information on each of the compute services version to determine the compute version cap to use. The current workaround is to pin the *upgrade_levels.compute* to a particular version like rocky and get the service up under such situations. See bug 1815697 for more details. Also note that in general during situations where cells are not reachable certain slowness may be experienced in operations requiring hitting all the cells because of the aforementioned configurable timeout/retry values.
2. **Counting Quotas:** Another known issue is in the current approach of counting quotas where we query each cell database to get the used resources and aggregate them which makes it sensitive to temporary cell outages. While the cell is unavailable, we cannot count resource usage residing in that cell database and things would behave as though more quota is available than should be. That is, if a tenant has used all of their quota and part of it is in cell A and cell A goes offline temporarily, that tenant will suddenly be able to allocate more resources than their limit (assuming cell A returns, the tenant will have more resources allocated than their allowed quota).

Note: Starting in the Train (20.0.0) release, it is possible to configure counting of quota usage from the placement service and API database to make quota usage calculations resilient to down or poor-performing cells in a multi-cell environment. See the *quotas documentation* for more details.

4.3.6 Config drives

Note: This section provides deployment information about the config drive feature. For end-user information about the config drive feature and instance metadata in general, refer to the *user guide*.

Config drives are special drives that are attached to an instance when it boots. The instance can mount this drive and read files from it to get information that is normally available through *the metadata service*.

There are many use cases for the config drive. One such use case is to pass a networking configuration when you do not use DHCP to assign IP addresses to instances. For example, you might pass the IP address configuration for the instance through the config drive, which the instance can mount and access before you configure the network settings for the instance. Another common reason to use config drives is load. If running something like the OpenStack puppet providers in your instances, they can hit the *metadata servers* every fifteen minutes, simultaneously for every instance you have. They are just checking in, and building facts, but its not insignificant load. With a config drive, that becomes a local (cached) disk read. Finally, using a config drive means youre not dependent on the metadata service being up, reachable, or performing well to do things like reboot your instance that runs cloud-init at the beginning.

Any modern guest operating system that is capable of mounting an ISO 9660 or VFAT file system can use the config drive.

Requirements and guidelines

To use the config drive, you must follow the following requirements for the compute host and image.

Compute host requirements

The following virt drivers support the config drive: libvirt, XenServer, Hyper-V, VMware, and (since 17.0.0 Queens) PowerVM. The Bare Metal service also supports the config drive.

- To use config drives with libvirt, XenServer, or VMware, you must first install the **genisoimage** package on each compute host. Use the *mkisofs_cmd* config option to set the path where you install the **genisoimage** program. If **genisoimage** is in the same path as the **nova-compute** service, you do not need to set this flag.
- To use config drives with Hyper-V, you must set the *mkisofs_cmd* config option to the full path to an **mkisofs.exe** installation. Additionally, you must set the *hyperv.qemu_img_cmd* config option to the full path to an **qemu-img** command installation.
- To use config drives with PowerVM or the Bare Metal service, you do not need to prepare anything.

Image requirements

An image built with a recent version of the cloud-init package can automatically access metadata passed through the config drive. The cloud-init package version 0.7.1 works with Ubuntu, Fedora based images (such as Red Hat Enterprise Linux) and openSUSE based images (such as SUSE Linux Enterprise Server). If an image does not have the cloud-init package installed, you must customize the image to run a script that mounts the config drive on boot, reads the data from the drive, and takes appropriate action such as adding the public key to an account. For more details about how data is organized on the config drive, refer to the *user guide*.

Configuration

The **nova-compute** service accepts the following config drive-related options:

- *api.config_drive_skip_versions*
- *force_config_drive*
- *config_drive_format*

If using the HyperV compute driver, the following additional options are supported:

- *hyperv.config_drive_cdrom*

For example, to ensure nova always provides a config drive to instances but versions `2018-08-27` (Rocky) and `2017-02-22` (Ocata) are skipped, add the following to `nova.conf`:

```
[DEFAULT]
force_config_drive = True

[api]
config_drive_skip_versions = 2018-08-27 2017-02-22
```

Note: The `img_config_drive` image metadata property can be used to force enable the config drive. In addition, users can explicitly request a config drive when booting instances. For more information, refer to the *user guide*.

Note: If using Xen with a config drive, you must use the *xenserver.disable_agent* config option to disable the agent.

4.3.7 Configuration

To configure your Compute installation, you must define configuration options in these files:

- `nova.conf` contains most of the Compute configuration options and resides in the `/etc/nova` directory.
- `api-paste.ini` defines Compute limits and resides in the `/etc/nova` directory.
- Configuration files for related services, such as the Image and Identity services.

A list of config options based on different topics can be found below:

Compute API configuration

The Compute API, is the component of OpenStack Compute that receives and responds to user requests, whether they be direct API calls, or via the CLI tools or dashboard.

Configure Compute API password handling

The OpenStack Compute API enables users to specify an administrative password when they create, rebuild, rescue or evacuate a server instance. If the user does not specify a password, a random password is generated and returned in the API response.

In practice, how the admin password is handled depends on the hypervisor in use and might require additional configuration of the instance. For example, you might have to install an agent to handle the password setting. If the hypervisor and instance configuration do not support setting a password at server create time, the password that is returned by the create API call is misleading because it was ignored.

To prevent this confusion, set the `enable_instance_password` configuration to `False` to disable the return of the admin password for installations that do not support setting instance passwords.

Configure resize

Resize (or Server resize) is the ability to change the flavor of a server, thus allowing it to upscale or downscale according to user needs. For this feature to work properly, you might need to configure some underlying virt layers.

Virt drivers

Todo: This section needs to be updated for other virt drivers, shared storage considerations, etc.

KVM

Resize on KVM is implemented currently by transferring the images between compute nodes over ssh. For KVM you need hostnames to resolve properly and passwordless ssh access between your compute hosts. Direct access from one compute host to another is needed to copy the VM file across.

Cloud end users can find out how to resize a server by reading *Resize an instance*.

XenServer

To get resize to work with XenServer (and XCP), you need to establish a root trust between all hypervisor nodes and provide an `/image` mount point to your hypervisors dom0.

Automatic confirm

There is a periodic task configured by configuration option *`resize_confirm_window`* (in seconds). If this value is not 0, the `nova-compute` service will check whether servers are in a resized state longer than the value of *`resize_confirm_window`* and if so will automatically confirm the resize of the servers.

Configuring Fibre Channel Support

Fibre Channel support in OpenStack Compute is remote block storage attached to compute nodes for VMs.

Todo: This below statement needs to be verified for current release

Fibre Channel supported only the KVM hypervisor.

Compute and Block Storage support Fibre Channel automatic zoning on Brocade and Cisco switches. On other hardware Fibre Channel arrays must be pre-zoned or directly attached to the KVM hosts.

KVM host requirements

You must install these packages on the KVM host:

sysfsutils Nova uses the `systool` application in this package.

sg3-utils or sg3_utils Nova uses the `sg_scan` and `sginfo` applications.

Installing the `multipath-tools` or `device-mapper-multipath` package is optional.

Configuring iSCSI interface and offload support

Compute supports open-iscsi iSCSI interfaces for offload cards. Offload hardware must be present and configured on every compute node where offload is desired. Once an open-iscsi interface is configured, the iface name (`iface.iscsi_ifacename`) should be passed to libvirt via the `iscsi_iface` parameter for use. All iSCSI sessions will be bound to this iSCSI interface.

Currently supported transports (`iface.transport_name`) are `be2iscsi`, `bnx2i`, `cxgb3i`, `cxgb4i`, `qla4xxx`, `ocs`. Configuration changes are required on the compute node only.

iSER is supported using the separate iSER LibvirtISERVolumeDriver and will be rejected if used via the `iscsi_iface` parameter.

iSCSI iface configuration

- Note the distinction between the transport name (`iface.transport_name`) and iface name (`iface.iscsi_ifacename`). The actual iface name must be specified via the iscsi_iface parameter to libvirt for offload to work.
- The default name for an iSCSI iface (open-iscsi parameter `iface.iscsi_ifacename`) is in the format transport_name.hwaddress when generated by `iscsiadm`.
- `iscsiadm` can be used to view and generate current iface configuration. Every network interface that supports an open-iscsi transport can have one or more iscsi ifaces associated with it. If no ifaces have been configured for a network interface supported by an open-iscsi transport, this command will create a default iface configuration for that network interface. For example :

```
# iscsiadm -m iface
default tcp,<empty>,<empty>,<empty>,<empty>
iser iser,<empty>,<empty>,<empty>,<empty>
bnx2i.00:05:b5:d2:a0:c2 bnx2i,00:05:b5:d2:a0:c2,5.10.10.20,<empty>,<empty>
```

  The output is in the format:

```
iface_name transport_name,hwaddress,ipaddress,net_ifacename,initiatorname
```

- Individual iface configuration can be viewed via

```
# iscsiadm -m iface -I IFACE_NAME
# BEGIN RECORD 2.0-873
iface.iscsi_ifacename = cxgb4i.00:07:43:28:b2:58
iface.net_ifacename = <empty>
iface.ipaddress = 102.50.50.80
iface.hwaddress = 00:07:43:28:b2:58
iface.transport_name = cxgb4i
iface.initiatorname = <empty>
# END RECORD
```

Configuration can be updated as desired via

```
# iscsiadm -m iface-I IFACE_NAME--op=update -n iface.SETTING -v VALUE
```

- All iface configurations need a minimum of `iface.iface_name`, `iface.transport_name` and `iface.hwaddress` to be correctly configured to work. Some transports may require `iface.ipaddress` and `iface.net_ifacename` as well to bind correctly.

  Detailed configuration instructions can be found at: https://github.com/open-iscsi/open-iscsi/blob/master/README

Hypervisors

Hypervisor Configuration Basics

The node where the `nova-compute` service is installed and operates on the same node that runs all of the virtual machines. This is referred to as the compute node in this guide.

By default, the selected hypervisor is KVM. To change to another hypervisor, change the `virt_type` option in the `[libvirt]` section of `nova.conf` and restart the `nova-compute` service.

Specific options for particular hypervisors can be found in the following sections.

KVM

Todo: Some of this is installation guide material and should probably be moved.

KVM is configured as the default hypervisor for Compute.

Note: This document contains several sections about hypervisor selection. If you are reading this document linearly, you do not want to load the KVM module before you install `nova-compute`. The `nova-compute` service depends on qemu-kvm, which installs `/lib/udev/rules.d/45-qemu-kvm.rules`, which sets the correct permissions on the `/dev/kvm` device node.

To enable KVM explicitly, add the following configuration options to the `/etc/nova/nova.conf` file:

```
compute_driver = libvirt.LibvirtDriver

[libvirt]
virt_type = kvm
```

The KVM hypervisor supports the following virtual machine image formats:

- Raw
- QEMU Copy-on-write (QCOW2)
- QED Qemu Enhanced Disk
- VMware virtual machine disk format (vmdk)

This section describes how to enable KVM on your system. For more information, see the following distribution-specific documentation:

- Fedora: Virtualization Getting Started Guide from the Fedora 22 documentation.

- Ubuntu: KVM/Installation from the Community Ubuntu documentation.
- Debian: Virtualization with KVM from the Debian handbook.
- Red Hat Enterprise Linux: Installing virtualization packages on an existing Red Hat Enterprise Linux system from the `Red Hat Enterprise Linux Virtualization Host Configuration and Guest Installation Guide`.
- openSUSE: Installing KVM from the openSUSE Virtualization with KVM manual.
- SLES: Installing KVM from the SUSE Linux Enterprise Server `Virtualization Guide`.

Enable KVM

The following sections outline how to enable KVM based hardware virtualization on different architectures and platforms. To perform these steps, you must be logged in as the `root` user.

For x86 based systems

1. To determine whether the `svm` or `vmx` CPU extensions are present, run this command:

```
# grep -E 'svm|vmx' /proc/cpuinfo
```

   This command generates output if the CPU is capable of hardware-virtualization. Even if output is shown, you might still need to enable virtualization in the system BIOS for full support.

   If no output appears, consult your system documentation to ensure that your CPU and motherboard support hardware virtualization. Verify that any relevant hardware virtualization options are enabled in the system BIOS.

   The BIOS for each manufacturer is different. If you must enable virtualization in the BIOS, look for an option containing the words `virtualization`, `VT`, `VMX`, or `SVM`.

2. To list the loaded kernel modules and verify that the `kvm` modules are loaded, run this command:

```
# lsmod | grep kvm
```

   If the output includes `kvm_intel` or `kvm_amd`, the `kvm` hardware virtualization modules are loaded and your kernel meets the module requirements for OpenStack Compute.

   If the output does not show that the `kvm` module is loaded, run this command to load it:

```
# modprobe -a kvm
```

   Run the command for your CPU. For Intel, run this command:

```
# modprobe -a kvm-intel
```

   For AMD, run this command:

```
# modprobe -a kvm-amd
```

   Because a KVM installation can change user group membership, you might need to log in again for changes to take effect.

   If the kernel modules do not load automatically, use the procedures listed in these subsections.

If the checks indicate that required hardware virtualization support or kernel modules are disabled or unavailable, you must either enable this support on the system or find a system with this support.

Note: Some systems require that you enable VT support in the system BIOS. If you believe your processor supports hardware acceleration but the previous command did not produce output, reboot your machine, enter the system BIOS, and enable the VT option.

If KVM acceleration is not supported, configure Compute to use a different hypervisor, such as `QEMU` or `Xen`. See *QEMU* or *XenServer (and other XAPI based Xen variants)* for details.

These procedures help you load the kernel modules for Intel-based and AMD-based processors if they do not load automatically during KVM installation.

Intel-based processors

If your compute host is Intel-based, run these commands as root to load the kernel modules:

```
# modprobe kvm
# modprobe kvm-intel
```

Add these lines to the `/etc/modules` file so that these modules load on reboot:

```
kvm
kvm-intel
```

AMD-based processors

If your compute host is AMD-based, run these commands as root to load the kernel modules:

```
# modprobe kvm
# modprobe kvm-amd
```

Add these lines to `/etc/modules` file so that these modules load on reboot:

```
kvm
kvm-amd
```

For POWER based systems

KVM as a hypervisor is supported on POWER systems PowerNV platform.

1. To determine if your POWER platform supports KVM based virtualization run the following command:

   ```
   # cat /proc/cpuinfo | grep PowerNV
   ```

   If the previous command generates the following output, then CPU supports KVM based virtualization.

   ```
   platform: PowerNV
   ```

   If no output is displayed, then your POWER platform does not support KVM based hardware virtualization.

2. To list the loaded kernel modules and verify that the `kvm` modules are loaded, run the following command:

```
# lsmod | grep kvm
```

If the output includes `kvm_hv`, the `kvm` hardware virtualization modules are loaded and your kernel meets the module requirements for OpenStack Compute.

If the output does not show that the `kvm` module is loaded, run the following command to load it:

```
# modprobe -a kvm
```

For PowerNV platform, run the following command:

```
# modprobe -a kvm-hv
```

Because a KVM installation can change user group membership, you might need to log in again for changes to take effect.

Configure Compute backing storage

Backing Storage is the storage used to provide the expanded operating system image, and any ephemeral storage. Inside the virtual machine, this is normally presented as two virtual hard disks (for example, `/dev/vda` and `/dev/vdb` respectively). However, inside OpenStack, this can be derived from one of these methods: `lvm`, `qcow`, `rbd` or `flat`, chosen using the *`libvirt.images_type`* option in `nova.conf` on the compute node.

Note: The option `raw` is acceptable but deprecated in favor of `flat`. The Flat back end uses either raw or QCOW2 storage. It never uses a backing store, so when using QCOW2 it copies an image rather than creating an overlay. By default, it creates raw files but will use QCOW2 when creating a disk from a QCOW2 if *`force_raw_images`* is not set in configuration.

QCOW is the default backing store. It uses a copy-on-write philosophy to delay allocation of storage until it is actually needed. This means that the space required for the backing of an image can be significantly less on the real disk than what seems available in the virtual machine operating system.

Flat creates files without any sort of file formatting, effectively creating files with the plain binary one would normally see on a real disk. This can increase performance, but means that the entire size of the virtual disk is reserved on the physical disk.

Local LVM volumes can also be used. Set the *`libvirt.images_volume_group`* configuration option to the name of the LVM group you have created.

Specify the CPU model of KVM guests

The Compute service enables you to control the guest CPU model that is exposed to KVM virtual machines. Use cases include:

- To maximize performance of virtual machines by exposing new host CPU features to the guest
- To ensure a consistent default CPU across all machines, removing reliance of variable QEMU defaults

In libvirt, the CPU is specified by providing a base CPU model name (which is a shorthand for a set of feature flags), a set of additional feature flags, and the topology (sockets/cores/threads). The libvirt KVM driver provides a number of standard CPU model names. These models are defined in the `/usr/share/libvirt/cpu_map.xml` file for libvirt prior to version 4.7.0 or `/usr/share/libvirt/cpu_map/*.xml` files thereafter. Make a check to determine which models are supported by your local installation.

Two Compute configuration options in the *libvirt* group of `nova.conf` define which type of CPU model is exposed to the hypervisor when using KVM: *libvirt.cpu_mode* and *libvirt.cpu_models*.

The *libvirt.cpu_mode* option can take one of the following values: `none`, `host-passthrough`, `host-model`, and `custom`.

See Effective Virtual CPU configuration in Nova for a recorded presentation about this topic.

Host model (default for KVM & QEMU)

If your `nova.conf` file contains `cpu_mode=host-model`, libvirt identifies the CPU model in `/usr/share/libvirt/cpu_map.xml` for version prior to 4.7.0 or `/usr/share/libvirt/cpu_map/*.xml` for version 4.7.0 and higher that most closely matches the host, and requests additional CPU flags to complete the match. This configuration provides the maximum functionality and performance and maintains good reliability.

With regard to enabling and facilitating live migration between compute nodes, you should assess whether `host-model` is suitable for your compute architecture. In general, using `host-model` is a safe choice if your compute node CPUs are largely identical. However, if your compute nodes span multiple processor generations, you may be better advised to select a `custom` CPU model.

Host pass through

If your `nova.conf` file contains `cpu_mode=host-passthrough`, libvirt tells KVM to pass through the host CPU with no modifications. The difference to host-model, instead of just matching feature flags, every last detail of the host CPU is matched. This gives the best performance, and can be important to some apps which check low level CPU details, but it comes at a cost with respect to migration.

In `host-passthrough` mode, the guest can only be live-migrated to a target host that matches the source host extremely closely. This definitely includes the physical CPU model and running microcode, and may even include the running kernel. Use this mode only if

- your compute nodes have a very large degree of homogeneity (i.e. substantially all of your compute nodes use the exact same CPU generation and model), and you make sure to only live-migrate between hosts with exactly matching kernel versions, *or*
- you decide, for some reason and against established best practices, that your compute infrastructure should not support any live migration at all.

Custom

If `nova.conf` contains *libvirt.cpu_mode*=custom, you can explicitly specify an ordered list of supported named models using the *libvirt.cpu_models* configuration option. It is expected that the list is ordered so that the more common and less advanced cpu models are listed earlier.

An end user can specify required CPU features through traits. When specified, the libvirt driver will select the first cpu model in the *libvirt.cpu_models* list that can provide the requested feature traits. If no CPU feature traits are specified then the instance will be configured with the first cpu model in the list.

For example, if specifying CPU features `avx` and `avx2` as follows:

```
$ openstack flavor set FLAVOR_ID --property trait:HW_CPU_X86_AVX=required \
                                 --property trait:HW_CPU_X86_AVX2=required
```

and *libvirt.cpu_models* is configured like this:

```
[libvirt]
cpu_mode = custom
cpu_models = Penryn,IvyBridge,Haswell,Broadwell,Skylake-Client
```

Then `Haswell`, the first cpu model supporting both `avx` and `avx2`, will be chosen by libvirt.

In selecting the `custom` mode, along with a *`libvirt.cpu_models`* that matches the oldest of your compute node CPUs, you can ensure that live migration between compute nodes will always be possible. However, you should ensure that the *`libvirt.cpu_models`* you select passes the correct CPU feature flags to the guest.

If you need to further tweak your CPU feature flags in the `custom` mode, see *Set CPU feature flags*.

Note: If *`libvirt.cpu_models`* is configured, the CPU models in the list needs to be compatible with the host CPU. Also, if *`libvirt.cpu_model_extra_flags`* is configured, all flags needs to be compatible with the host CPU. If incompatible CPU models or flags are specified, nova service will raise an error and fail to start.

None (default for all libvirt-driven hypervisors other than KVM & QEMU)

If your `nova.conf` file contains `cpu_mode=none`, libvirt does not specify a CPU model. Instead, the hypervisor chooses the default model.

Set CPU feature flags

Regardless of whether your selected *`libvirt.cpu_mode`* is `host-passthrough`, `host-model`, or `custom`, it is also possible to selectively enable additional feature flags. Suppose your selected `custom` CPU model is `IvyBridge`, which normally does not enable the `pcid` feature flag but you do want to pass `pcid` into your guest instances. In that case, you would set:

```
[libvirt]
cpu_mode = custom
cpu_models = IvyBridge
cpu_model_extra_flags = pcid
```

Nested guest support

You may choose to enable support for nested guests that is, allow your Nova instances to themselves run hardware-accelerated virtual machines with KVM. Doing so requires a module parameter on your KVM kernel module, and corresponding `nova.conf` settings.

Nested guest support in the KVM kernel module

To enable nested KVM guests, your compute node must load the `kvm_intel` or `kvm_amd` module with `nested=1`. You can enable the `nested` parameter permanently, by creating a file named `/etc/modprobe.d/kvm.conf` and populating it with the following content:

```
options kvm_intel nested=1
options kvm_amd nested=1
```

A reboot may be required for the change to become effective.

Nested guest support in `nova.conf`

To support nested guests, you must set your *`libvirt.cpu_mode`* configuration to one of the following options:

Host pass through In this mode, nested virtualization is automatically enabled once the KVM kernel module is loaded with nesting support.

```
[libvirt]
cpu_mode = host-passthrough
```

However, do consider the other implications that *Host pass through* mode has on compute functionality.

Host model In this mode, nested virtualization is automatically enabled once the KVM kernel module is loaded with nesting support, **if** the matching CPU model exposes the `vmx` feature flag to guests by default (you can verify this with `virsh capabilities` on your compute node). If your CPU model does not pass in the `vmx` flag, you can force it with *`libvirt.cpu_model_extra_flags`*:

```
[libvirt]
cpu_mode = host-model
cpu_model_extra_flags = vmx
```

Again, consider the other implications that apply to the *Host model (default for KVM & Qemu)* mode.

Custom In custom mode, the same considerations apply as in host-model mode, but you may *additionally* want to ensure that libvirt passes not only the `vmx`, but also the `pcid` flag to its guests:

```
[libvirt]
cpu_mode = custom
cpu_models = IvyBridge
cpu_model_extra_flags = vmx,pcid
```

Nested guest support limitations

When enabling nested guests, you should be aware of (and inform your users about) certain limitations that are currently inherent to nested KVM virtualization. Most importantly, guests using nested virtualization will, *while nested guests are running*,

- fail to complete live migration;
- fail to resume from suspend.

See the KVM documentation for more information on these limitations.

AMD SEV (Secure Encrypted Virtualization)

Secure Encrypted Virtualization (SEV) is a technology from AMD which enables the memory for a VM to be encrypted with a key unique to the VM. SEV is particularly applicable to cloud computing since it can reduce the amount of trust VMs need to place in the hypervisor and administrator of their host system.

Nova supports SEV from the Train release onwards.

Requirements for SEV

First the operator will need to ensure the following prerequisites are met:

- At least one of the Nova compute hosts must be AMD hardware capable of supporting SEV. It is entirely possible for the compute plane to be a mix of hardware which can and cannot support SEV, although as per the section on *Permanent limitations* below, the maximum number of simultaneously running guests with SEV will be limited by the quantity and quality of SEV-capable hardware available.
- An appropriately configured software stack on those compute hosts, so that the various layers are all SEV ready:
  - kernel >= 4.16
  - QEMU >= 2.12
  - libvirt >= 4.5
  - ovmf >= commit 75b7aa9528bd 2018-07-06

Deploying SEV-capable infrastructure

In order for users to be able to use SEV, the operator will need to perform the following steps:

- Ensure that sufficient memory is reserved on the SEV compute hosts for host-level services to function correctly at all times. This is particularly important when hosting SEV-enabled guests, since they pin pages in RAM, preventing any memory overcommit which may be in normal operation on other compute hosts.

  It is recommended to achieve this by configuring an `rlimit` at the `/machine.slice` top-level `cgroup` on the host, with all VMs placed inside that. (For extreme detail, see this discussion on the spec.)

  An alternative approach is to configure the *`reserved_host_memory_mb`* option in the `[DEFAULT]` section of `nova.conf`, based on the expected maximum number of SEV guests simultaneously running on the host, and the details provided in an earlier version of the AMD SEV spec regarding memory region sizes, which cover how to calculate it correctly.

  See the Memory Locking and Accounting section of the AMD SEV spec and previous discussion for further details.

- A cloud administrator will need to define one or more SEV-enabled flavors *as described in the user guide*, unless it is sufficient for users to define SEV-enabled images.

Additionally the cloud operator should consider the following optional steps:

- Configure the *`libvirt.num_memory_encrypted_guests`* option in `nova.conf` to represent the number of guests an SEV compute node can host concurrently with memory encrypted at the hardware level. For example:

  ```
  [libvirt]
  num_memory_encrypted_guests = 15
  ```

  This option exists because on AMD SEV-capable hardware, the memory controller has a fixed number of slots for holding encryption keys, one per guest. For example, at the time of writing, earlier generations of hardware only have 15 slots, thereby limiting the number of SEV guests which can be run concurrently to 15. Nova needs to track how many slots are available and used in order to avoid attempting to exceed that limit in the hardware.

  At the time of writing (September 2019), work is in progress to allow QEMU and libvirt to expose the number of slots available on SEV hardware; however until this is finished and released, it will not be possible for Nova to programmatically detect the correct value.

  So this configuration option serves as a stop-gap, allowing the cloud operator the option of providing this value manually. It may later be demoted to a fallback value for cases where the limit cannot be detected programmatically, or even removed altogether when Novas minimum QEMU version guarantees that it can always be detected.

Note: When deciding whether to use the default of `None` or manually impose a limit, operators should carefully weigh the benefits vs. the risk. The benefits of using the default are a) immediate convenience since nothing needs to be done now, and b) convenience later when upgrading compute hosts to future versions of Nova, since again nothing will need to be done for the correct limit to be automatically imposed. However the risk is that until auto-detection is implemented, users may be able to attempt to launch guests with encrypted memory on hosts which have already reached the maximum number of guests simultaneously running with encrypted memory. This risk may be mitigated by other limitations which operators can impose, for example if the smallest RAM footprint of any flavor imposes a maximum number of simultaneously running guests which is less than or equal to the SEV limit.

- Configure *`libvirt.hw_machine_type`* on all SEV-capable compute hosts to include `x86_64=q35`, so that all x86_64 images use the `q35` machine type by default. (Currently Nova defaults to the `pc` machine type for the `x86_64` architecture, although it is expected that this will change in the future.)

  Changing the default from `pc` to `q35` makes the creation and configuration of images by users more convenient by removing the need for the `hw_machine_type` property to be set to `q35` on every image for which SEV booting is desired.

  Caution: Consider carefully whether to set this option. It is particularly important since a limitation of the implementation prevents the user from receiving an error message with a helpful explanation if they try to boot an SEV guest when neither this configuration option nor the image property are set to select a `q35` machine type.

  On the other hand, setting it to `q35` may have other undesirable side-effects on other images which were expecting to be booted with `pc`, so it is suggested to set it on a single compute node or aggregate, and perform careful testing of typical images before rolling out the setting to all SEV-capable compute hosts.

Launching SEV instances

Once an operator has covered the above steps, users can launch SEV instances either by requesting a flavor for which the operator set the `hw:mem_encryption` extra spec to `True`, or by using an image with the `hw_mem_encryption` property set to `True`.

These do not inherently cause a preference for SEV-capable hardware, but for now SEV is the only way of fulfilling the requirement for memory encryption. However in the future, support for other hardware-level guest memory encryption technology such as Intel MKTME may be added. If a guest specifically needs to be booted using SEV rather than any other memory encryption technology, it is possible to ensure this by adding `trait:HW_CPU_X86_AMD_SEV=required` to the flavor extra specs or image properties.

In all cases, SEV instances can only be booted from images which have the `hw_firmware_type` property set to `uefi`, and only when the machine type is set to `q35`. This can be set per image by setting the image property `hw_machine_type=q35`, or per compute node by the operator via *`libvirt.hw_machine_type`* as explained above.

Impermanent limitations

The following limitations may be removed in the future as the hardware, firmware, and various layers of software receive new features:

- SEV-encrypted VMs cannot yet be live-migrated or suspended, therefore they will need to be fully shut down before migrating off an SEV host, e.g. if maintenance is required on the host.

- SEV-encrypted VMs cannot contain directly accessible host devices (PCI passthrough). So for example mdev vGPU support will not currently work. However technologies based on vhost-user should work fine.
- The boot disk of SEV-encrypted VMs can only be `virtio`. (`virtio-blk` is typically the default for libvirt disks on x86, but can also be explicitly set e.g. via the image property `hw_disk_bus=virtio`). Valid alternatives for the disk include using `hw_disk_bus=scsi` with `hw_scsi_model=virtio-scsi` , or `hw_disk_bus=sata`.
- QEMU and libvirt cannot yet expose the number of slots available for encrypted guests in the memory controller on SEV hardware. Until this is implemented, it is not possible for Nova to programmatically detect the correct value. As a short-term workaround, operators can optionally manually specify the upper limit of SEV guests for each compute host, via the new *`libvirt.num_memory_encrypted_guests`* configuration option *described above*.

Permanent limitations

The following limitations are expected long-term:

- The number of SEV guests allowed to run concurrently will always be limited. On the first generation of EPYC machines it will be limited to 15 guests; however this limit becomes much higher with the second generation (Rome).
- The operating system running in an encrypted virtual machine must contain SEV support.

Non-limitations

For the sake of eliminating any doubt, the following actions are *not* expected to be limited when SEV encryption is used:

- Cold migration or shelve, since they power off the VM before the operation at which point there is no encrypted memory (although this could change since there is work underway to add support for PMEM)
- Snapshot, since it only snapshots the disk
- `nova evacuate` (despite the name, more akin to resurrection than evacuation), since this is only initiated when the VM is no longer running
- Attaching any volumes, as long as they do not require attaching via an IDE bus
- Use of spice / VNC / serial / RDP consoles
- VM guest virtual NUMA (a.k.a. vNUMA)

For further technical details, see the nova spec for SEV support.

Guest agent support

Use guest agents to enable optional access between compute nodes and guests through a socket, using the QMP protocol.

To enable this feature, you must set `hw_qemu_guest_agent=yes` as a metadata parameter on the image you wish to use to create the guest-agent-capable instances from. You can explicitly disable the feature by setting `hw_qemu_guest_agent=no` in the image metadata.

KVM performance tweaks

The VHostNet kernel module improves network performance. To load the kernel module, run the following command as root:

```
# modprobe vhost_net
```

Troubleshoot KVM

Trying to launch a new virtual machine instance fails with the `ERROR` state, and the following error appears in the `/var/log/nova/nova-compute.log` file:

```
libvirtError: internal error no supported architecture for os type 'hvm'
```

This message indicates that the KVM kernel modules were not loaded.

If you cannot start VMs after installation without rebooting, the permissions might not be set correctly. This can happen if you load the KVM module before you install `nova-compute`. To check whether the group is set to `kvm`, run:

```
# ls -l /dev/kvm
```

If it is not set to `kvm`, run:

```
# udevadm trigger
```

QEMU

From the perspective of the Compute service, the QEMU hypervisor is very similar to the KVM hypervisor. Both are controlled through libvirt, both support the same feature set, and all virtual machine images that are compatible with KVM are also compatible with QEMU. The main difference is that QEMU does not support native virtualization. Consequently, QEMU has worse performance than KVM and is a poor choice for a production deployment.

The typical uses cases for QEMU are

- Running on older hardware that lacks virtualization support.
- Running the Compute service inside of a virtual machine for development or testing purposes, where the hypervisor does not support native virtualization for guests.

To enable QEMU, add these settings to `nova.conf`:

```
compute_driver = libvirt.LibvirtDriver

[libvirt]
virt_type = qemu
```

For some operations you may also have to install the **`guestmount`** utility:

On Ubuntu:

```
# apt-get install guestmount
```

On Red Hat Enterprise Linux, Fedora, or CentOS:

```
# yum install libguestfs-tools
```

On openSUSE:

```
# zypper install guestfs-tools
```

The QEMU hypervisor supports the following virtual machine image formats:

- Raw
- QEMU Copy-on-write (qcow2)
- VMware virtual machine disk format (vmdk)

XenServer (and other XAPI based Xen variants)

Deprecated since version 20.0.0: The xenapi driver is deprecated and may be removed in a future release. The driver is not tested by the OpenStack project nor does it have clear maintainer(s) and thus its quality can not be ensured. If you are using the driver in production please let us know in freenode IRC and/or the openstack-discuss mailing list.

Todo: os-xenapi version is 0.3.1 currently. This document should be modified according to the new version. This todo has been reported as bug 1718606.

This section describes XAPI managed hypervisors, and how to use them with OpenStack.

Terminology

Xen

A hypervisor that provides the fundamental isolation between virtual machines. Xen is open source (GPLv2) and is managed by XenProject.org, a cross-industry organization and a Linux Foundation Collaborative project.

Xen is a component of many different products and projects. The hypervisor itself is very similar across all these projects, but the way that it is managed can be different, which can cause confusion if youre not clear which toolstack you are using. Make sure you know what toolstack you want before you get started. If you want to use Xen with libvirt in OpenStack Compute refer to *Xen via libvirt*.

XAPI

XAPI is one of the toolstacks that could control a Xen based hypervisor. XAPIs role is similar to libvirts in the KVM world. The API provided by XAPI is called XenAPI. To learn more about the provided interface, look at XenAPI Object Model Overview for definitions of XAPI specific terms such as SR, VDI, VIF and PIF.

OpenStack has a compute driver which talks to XAPI, therefore all XAPI managed servers could be used with OpenStack.

XenAPI

XenAPI is the API provided by XAPI. This name is also used by the python library that is a client for XAPI. A set of packages to use XenAPI on existing distributions can be built using the xenserver/buildroot project.

XenServer

An Open Source virtualization platform that delivers all features needed for any server and datacenter implementation including the Xen hypervisor and XAPI for the management. For more information and product downloads, visit xenserver.org.

XCP

XCP is not supported anymore. XCP project recommends all XCP users to upgrade to the latest version of XenServer by visiting xenserver.org.

Privileged and unprivileged domains

A Xen host runs a number of virtual machines, VMs, or domains (the terms are synonymous on Xen). One of these is in charge of running the rest of the system, and is known as domain 0, or dom0. It is the first domain to boot after Xen, and owns the storage and networking hardware, the device drivers, and the primary control software. Any other VM is unprivileged, and is known as a domU or guest. All customer VMs are unprivileged, but you should note that on XenServer (and other XenAPI using hypervisors), the OpenStack Compute service (`nova-compute`) also runs in a domU. This gives a level of security isolation between the privileged system software and the OpenStack software (much of which is customer-facing). This architecture is described in more detail later.

Paravirtualized versus hardware virtualized domains

A Xen virtual machine can be paravirtualized (PV) or hardware virtualized (HVM). This refers to the interaction between Xen, domain 0, and the guest VMs kernel. PV guests are aware of the fact that they are virtualized and will co-operate with Xen and domain 0; this gives them better performance characteristics. HVM guests are not aware of their environment, and the hardware has to pretend that they are running on an unvirtualized machine. HVM guests do not need to modify the guest operating system, which is essential when running Windows.

In OpenStack, customer VMs may run in either PV or HVM mode. However, the OpenStack domU (thats the one running `nova-compute`) must be running in PV mode.

xapi pool

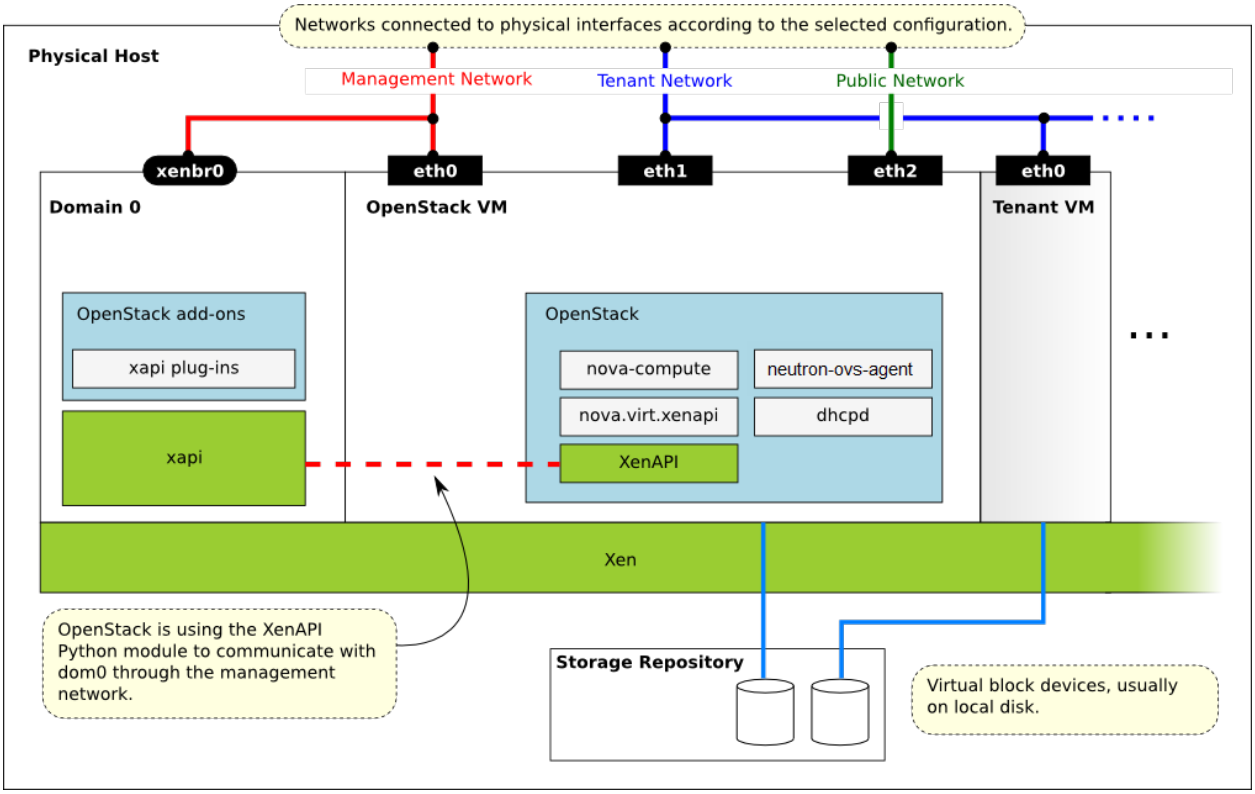
A resource pool comprises multiple XenServer host installations, bound together into a single managed entity which can host virtual machines. When combined with shared storage, VMs could dynamically move between XenServer hosts, with minimal downtime since no block copying is needed.

XenAPI deployment architecture

A basic OpenStack deployment on a XAPI-managed server, assuming that the network provider is neutron network, looks like this:

Key things to note:

- The hypervisor: Xen
- Domain 0: runs XAPI and some small pieces from OpenStack, the XAPI plug-ins.



- OpenStack VM: The `Compute` service runs in a paravirtualized virtual machine, on the host under management. Each host runs a local instance of `Compute`. It is also running neutron plugin-agent (`neutron-openvswitch-agent`) to perform local vSwitch configuration.
- OpenStack Compute uses the XenAPI Python library to talk to XAPI, and it uses the Management Network to reach from the OpenStack VM to Domain 0.

Some notes on the networking:

- The above diagram assumes DHCP networking.
- There are three main OpenStack networks:
  - Management network: RabbitMQ, MySQL, inter-host communication, and compute-XAPI communication. Please note that the VM images are downloaded by the XenAPI plug-ins, so make sure that the OpenStack Image service is accessible through this network. It usually means binding those services to the management interface.
  - Tenant network: controlled by neutron, this is used for tenant traffic.
  - Public network: floating IPs, public API endpoints.
- The networks shown here must be connected to the corresponding physical networks within the data center. In the simplest case, three individual physical network cards could be used. It is also possible to use VLANs to separate these networks. Please note, that the selected configuration must be in line with the networking model selected for the cloud. (In case of VLAN networking, the physical channels have to be able to forward the tagged traffic.)
- With the Networking service, you should enable Linux bridge in `Dom0` which is used for Compute service. `nova-compute` will create Linux bridges for security group and `neutron-openvswitch-agent` in Compute node will apply security group rules on these Linux bridges. To implement this, you need to remove `/etc/modprobe.d/blacklist-bridge*` in `Dom0`.

Further reading

Here are some of the resources available to learn more about Xen:

- Citrix XenServer official documentation
- What is Xen? by XenProject.org
- Xen Hypervisor project
- Xapi project
- Further XenServer and OpenStack information

Install XenServer

Before you can run OpenStack with XenServer, you must install the hypervisor on an appropriate server.

Note: Xen is a type 1 hypervisor: When your server starts, Xen is the first software that runs. Consequently, you must install XenServer before you install the operating system where you want to run OpenStack code. You then install `nova-compute` into a dedicated virtual machine on the host.

Use the following link to download XenServers installation media:

- http://xenserver.org/open-source-virtualization-download.html

When you install many servers, you might find it easier to perform PXE boot installations. You can also package any post-installation changes that you want to make to your XenServer by following the instructions of creating your own XenServer supplemental pack.

Important: When using `[xenserver]image_handler=direct_vhd` (the default), make sure you use the EXT type of storage repository (SR). Features that require access to VHD files (such as copy on write, snapshot and migration) do not work when you use the LVM SR. Storage repository (SR) is a XAPI-specific term relating to the physical storage where virtual disks are stored.

On the XenServer installation screen, choose the *XenDesktop Optimized* option. If you use an answer file, make sure you use `srtype="ext"` in the `installation` tag of the answer file.

Post-installation steps

The following steps need to be completed after the hypervisors installation:

1. For resize and migrate functionality, enable password-less SSH authentication and set up the `/images` directory on dom0.
2. Install the XAPI plug-ins.
3. To support AMI type images, you must set up `/boot/guest` symlink/directory in dom0.
4. Create a paravirtualized virtual machine that can run `nova-compute`.
5. Install and configure `nova-compute` in the above virtual machine.
6. To support live migration requiring no block device migration, you should add the current host to a xapi pool using shared storage. You need to know the pool master ip address, user name and password:

```
xe pool-join master-address=MASTER_IP master-username=root master-password=MASTER_
↪PASSWORD
```

Install XAPI plug-ins

When you use a XAPI managed hypervisor, you can install a Python script (or any executable) on the host side, and execute that through XenAPI. These scripts are called plug-ins. The OpenStack related XAPI plug-ins live in OpenStack os-xenapi code repository. These plug-ins have to be copied to dom0s filesystem, to the appropriate directory, where XAPI can find them. It is important to ensure that the version of the plug-ins are in line with the OpenStack Compute installation you are using.

The plugins should typically be copied from the Nova installation running in the Computes DomU (`pip show os-xenapi` to find its location), but if you want to download the latest version the following procedure can be used.

Manually installing the plug-ins

1. Create temporary files/directories:

```
$ OS_XENAPI_TARBALL=$(mktemp)
$ OS_XENAPI_SOURCES=$(mktemp -d)
```

2. Get the source from the openstack.org archives. The example assumes the latest release is used, and the XenServer host is accessible as xenserver. Match those parameters to your setup.

```
$ OS_XENAPI_URL=https://tarballs.openstack.org/os-xenapi/os-xenapi-0.1.1.tar.gz
$ wget -qO "$OS_XENAPI_TARBALL" "$OS_XENAPI_URL"
$ tar xvf "$OS_XENAPI_TARBALL" -d "$OS_XENAPI_SOURCES"
```

3. Copy the plug-ins to the hypervisor:

```
$ PLUGINPATH=$(find $OS_XENAPI_SOURCES -path '*/xapi.d/plugins' -type d -print)
$ tar -czf - -C "$PLUGINPATH" ./ |
> ssh root@xenserver tar -xozf - -C /etc/xapi.d/plugins
```

4. Remove temporary files/directories:</para>

```
$ rm "$OS_XENAPI_TARBALL"
$ rm -rf "$OS_XENAPI_SOURCES"
```

Prepare for AMI type images

To support AMI type images in your OpenStack installation, you must create the `/boot/guest` directory on dom0. One of the OpenStack XAPI plugins will extract the kernel and ramdisk from AKI and ARI images and put them to that directory.

OpenStack maintains the contents of this directory and its size should not increase during normal operation. However, in case of power failures or accidental shutdowns, some files might be left over. To prevent these files from filling up dom0s filesystem, set up this directory as a symlink that points to a subdirectory of the local SR.

Run these commands in dom0 to achieve this setup:

```
# LOCAL_SR=$(xe sr-list name-label="Local storage" --minimal)
# LOCALPATH="/var/run/sr-mount/$LOCAL_SR/os-guest-kernels"
# mkdir -p "$LOCALPATH"
# ln -s "$LOCALPATH" /boot/guest
```

Modify dom0 for resize/migration support

To resize servers with XenServer you must:

- Establish a root trust between all hypervisor nodes of your deployment:

  To do so, generate an ssh key-pair with the **ssh-keygen** command. Ensure that each of your dom0s `authorized_keys` file (located in `/root/.ssh/authorized_keys`) contains the public key fingerprint (located in `/root/.ssh/id_rsa.pub`).

- Provide a `/images` mount point to the dom0 for your hypervisor:

  dom0 space is at a premium so creating a directory in dom0 is potentially dangerous and likely to fail especially when you resize large servers. The least you can do is to symlink `/images` to your local storage SR. The following instructions work for an English-based installation of XenServer and in the case of ext3-based SR (with which the resize functionality is known to work correctly).

```
# LOCAL_SR=$(xe sr-list name-label="Local storage" --minimal)
# IMG_DIR="/var/run/sr-mount/$LOCAL_SR/images"
# mkdir -p "$IMG_DIR"
# ln -s "$IMG_DIR" /images
```

XenAPI configuration reference

The following section discusses some commonly changed options when using the XenAPI driver. The table below provides a complete reference of all configuration options available for configuring XAPI with OpenStack.

The recommended way to use XAPI with OpenStack is through the XenAPI driver. To enable the XenAPI driver, add the following configuration options to `/etc/nova/nova.conf` and restart `OpenStack Compute`:

```
compute_driver = xenapi.XenAPIDriver
[xenserver]
connection_url = http://your_xenapi_management_ip_address
connection_username = root
connection_password = your_password
ovs_integration_bridge = br-int
```

These connection details are used by OpenStack Compute service to contact your hypervisor and are the same details you use to connect XenCenter, the XenServer management console, to your XenServer node.

Note: The `connection_url` is generally the management network IP address of the XenServer.

Networking configuration

The Networking service in the Compute node is running `neutron-openvswitch-agent`. This manages `dom0`s OVS. You should refer to the openvswitch_agent.ini sample for details, however there are several specific items to look out for.

```
[agent]
minimize_polling = False
root_helper_daemon = xenapi_root_helper

[ovs]
```

(continues on next page)

(continued from previous page)

```
of_listen_address = management_ip_address
ovsdb_connection = tcp:your_xenapi_management_ip_address:6640
bridge_mappings = <physical_network>:<physical_bridge>, ...
integration_bridge = br-int

[xenapi]
connection_url = http://your_xenapi_management_ip_address
connection_username = root
connection_password = your_pass_word
```

Note: The `ovsdb_connection` is the connection string for the native OVSDB backend, you need to enable port 6640 in dom0.

Agent

The agent is a piece of software that runs on the instances, and communicates with OpenStack. In case of the XenAPI driver, the agent communicates with OpenStack through XenStore (see the Xen Project Wiki for more information on XenStore).

If you dont have the guest agent on your VMs, it takes a long time for OpenStack Compute to detect that the VM has successfully started. Generally a large timeout is required for Windows instances, but you may want to adjust: `agent_version_timeout` within the `[xenserver]` section.

VNC proxy address

Assuming you are talking to XAPI through a management network, and XenServer is on the address: 10.10.1.34 specify the same address for the vnc proxy address: `server_proxyclient_address=10.10.1.34`

Storage

You can specify which Storage Repository to use with nova by editing the following flag. To use the local-storage setup by the default installer:

```
sr_matching_filter = "other-config:i18n-key=local-storage"
```

Another alternative is to use the default storage (for example if you have attached NFS or any other shared storage):

```
sr_matching_filter = "default-sr:true"
```

Use different image handler

We support three different implementations for glance image handler. You can choose a specific image handler based on the demand:

- `direct_vhd`: This image handler will call XAPI plugins to directly process the VHD files in XenServer SR(Storage Repository). So this handler only works when the hosts SR type is file system based e.g. ext, nfs.
- `vdi_local_dev`: This image handler uploads `tgz` compressed raw disk images to the glance image service.

- `vdi_remote_stream`: With this image handler, the image data streams between XenServer and the glance image service. As it uses the remote APIs supported by XAPI, this plugin works for all SR types supported by XenServer.

`direct_vhd` is the default image handler. If want to use a different image handler, you can change the config setting of `image_handler` within the `[xenserver]` section. For example, the following config setting is to use `vdi_remote_stream` as the image handler:

```
[xenserver]
image_handler=vdi_remote_stream
```

Xen via libvirt

OpenStack Compute supports the Xen Project Hypervisor (or Xen). Xen can be integrated with OpenStack Compute via the libvirt toolstack or via the XAPI toolstack. This section describes how to set up OpenStack Compute with Xen and libvirt. For information on how to set up Xen with XAPI refer to *XenServer (and other XAPI based Xen variants)*.

Installing Xen with libvirt

At this stage we recommend using the baseline that we use for the Xen Project OpenStack CI Loop, which contains the most recent stability fixes to both Xen and libvirt.

Xen 4.5.1 (or newer) and libvirt 1.2.15 (or newer) contain the minimum required OpenStack improvements for Xen. Although libvirt 1.2.15 works with Xen, libvirt 1.3.2 or newer is recommended. The necessary Xen changes have also been backported to the Xen 4.4.3 stable branch. Please check with the Linux and FreeBSD distros you are intending to use as Dom 0, whether the relevant version of Xen and libvirt are available as installable packages.

The latest releases of Xen and libvirt packages that fulfil the above minimum requirements for the various openSUSE distributions can always be found and installed from the Open Build Service Virtualization project. To install these latest packages, add the Virtualization repository to your software management stack and get the newest packages from there. More information about the latest Xen and libvirt packages are available here and here.

Alternatively, it is possible to use the Ubuntu LTS 14.04 Xen Package **4.4.1-0ubuntu0.14.04.4** (Xen 4.4.1) and apply the patches outlined here. You can also use the Ubuntu LTS 14.04 libvirt package **1.2.2 libvirt_1.2.2-0ubuntu13.1.7** as baseline and update it to libvirt version 1.2.15, or 1.2.14 with the patches outlined here applied. Note that this will require rebuilding these packages partly from source.

For further information and latest developments, you may want to consult the Xen Projects mailing lists for OpenStack related issues and questions.

Configuring Xen with libvirt

To enable Xen via libvirt, ensure the following options are set in `/etc/nova/nova.conf` on all hosts running the `nova-compute` service.

```
compute_driver = libvirt.LibvirtDriver

[libvirt]
virt_type = xen
```

Additional configuration options

Use the following as a guideline for configuring Xen for use in OpenStack:

1. **Dom0 memory**: Set it between 1GB and 4GB by adding the following parameter to the Xen Boot Options in the grub.conf file.

```
dom0_mem=1024M
```

Note: The above memory limits are suggestions and should be based on the available compute host resources. For large hosts that will run many hundreds of instances, the suggested values may need to be higher.

Note: The location of the grub.conf file depends on the host Linux distribution that you are using. Please refer to the distro documentation for more details (see Dom 0 for more resources).

2. **Dom0 vcpus**: Set the virtual CPUs to 4 and employ CPU pinning by adding the following parameters to the Xen Boot Options in the grub.conf file.

```
dom0_max_vcpus=4 dom0_vcpus_pin
```

Note: Note that the above virtual CPU limits are suggestions and should be based on the available compute host resources. For large hosts, that will run many hundred of instances, the suggested values may need to be higher.

3. **PV vs HVM guests**: A Xen virtual machine can be paravirtualized (PV) or hardware virtualized (HVM). The virtualization mode determines the interaction between Xen, Dom 0, and the guest VMs kernel. PV guests are aware of the fact that they are virtualized and will co-operate with Xen and Dom 0. The choice of virtualization mode determines performance characteristics. For an overview of Xen virtualization modes, see Xen Guest Types.

   In OpenStack, customer VMs may run in either PV or HVM mode. The mode is a property of the operating system image used by the VM, and is changed by adjusting the image metadata stored in the Image service. The image metadata can be changed using the **openstack** commands.

   To choose one of the HVM modes (HVM, HVM with PV Drivers or PVHVM), use **openstack** to set the `vm_mode` property to `hvm`.

   To choose one of the HVM modes (HVM, HVM with PV Drivers or PVHVM), use one of the following two commands:

```
$ openstack image set --property vm_mode=hvm IMAGE
```

   To chose PV mode, which is supported by NetBSD, FreeBSD and Linux, use one of the following two commands

```
$ openstack image set --property vm_mode=xen IMAGE
```

Note: The default for virtualization mode in nova is PV mode.

4. **Image formats**: Xen supports raw, qcow2 and vhd image formats. For more information on image formats, refer to the OpenStack Virtual Image Guide and the Storage Options Guide on the Xen Project Wiki.

5. **Image metadata**: In addition to the `vm_mode` property discussed above, the `hypervisor_type` property is another important component of the image metadata, especially if your cloud contains mixed hypervisor compute nodes. Setting the `hypervisor_type` property allows the nova scheduler to select a compute node

running the specified hypervisor when launching instances of the image. Image metadata such as `vm_mode`, `hypervisor_type`, architecture, and others can be set when importing the image to the Image service. The metadata can also be changed using the **`openstack`** commands:

```
$ openstack image set --property hypervisor_type=xen vm_mode=hvm IMAGE
```

For more more information on image metadata, refer to the OpenStack Virtual Image Guide.

6. **Libguestfs file injection**: OpenStack compute nodes can use libguestfs to inject files into an instances image prior to launching the instance. libguestfs uses libvirts QEMU driver to start a qemu process, which is then used to inject files into the image. When using libguestfs for file injection, the compute node must have the libvirt qemu driver installed, in addition to the Xen driver. In RPM based distributions, the qemu driver is provided by the `libvirt-daemon-qemu` package. In Debian and Ubuntu, the qemu driver is provided by the `libvirt-bin` package.

Troubleshoot Xen with libvirt

Important log files: When an instance fails to start, or when you come across other issues, you should first consult the following log files:

- `/var/log/nova/nova-compute.log`
- `/var/log/libvirt/libxl/libxl-driver.log`,
- `/var/log/xen/qemu-dm-${instancename}.log`,
- `/var/log/xen/xen-hotplug.log`,
- `/var/log/xen/console/guest-${instancename}` (to enable see Enabling Guest Console Logs)
- Host Console Logs (read Enabling and Retrieving Host Console Logs).

If you need further help you can ask questions on the mailing lists xen-users@, wg-openstack@ or raise a bug against Xen.

Known issues

- **Networking**: Xen via libvirt is currently only supported with nova-network. Fixes for a number of bugs are currently being worked on to make sure that Xen via libvirt will also work with OpenStack Networking (neutron).

Todo: Is this still true?

- **Live migration**: Live migration is supported in the libvirt libxl driver since version 1.2.5. However, there were a number of issues when used with OpenStack, in particular with libvirt migration protocol compatibility. It is worth mentioning that libvirt 1.3.0 addresses most of these issues. We do however recommend using libvirt 1.3.2, which is fully supported and tested as part of the Xen Project CI loop. It addresses live migration monitoring related issues and adds support for peer-to-peer migration mode, which nova relies on.
- **Live migration monitoring**: On compute nodes running Kilo or later, live migration monitoring relies on libvirt APIs that are only implemented from libvirt version 1.3.1 onwards. When attempting to live migrate, the migration monitoring thread would crash and leave the instance state as MIGRATING. If you experience such an issue and you are running on a version released before libvirt 1.3.1, make sure you backport libvirt commits ad71665 and b7b4391 from upstream.

Additional information and resources

The following section contains links to other useful resources.

- wiki.xenproject.org/wiki/OpenStack - OpenStack Documentation on the Xen Project wiki
- wiki.xenproject.org/wiki/OpenStack_CI_Loop_for_Xen-Libvirt - Information about the Xen Project OpenStack CI Loop
- wiki.xenproject.org/wiki/OpenStack_via_DevStack - How to set up OpenStack via DevStack
- Mailing lists for OpenStack related issues and questions - This list is dedicated to coordinating bug fixes and issues across Xen, libvirt and OpenStack and the CI loop.

LXC (Linux containers)

LXC (also known as Linux containers) is a virtualization technology that works at the operating system level. This is different from hardware virtualization, the approach used by other hypervisors such as KVM, Xen, and VMware. LXC (as currently implemented using libvirt in the Compute service) is not a secure virtualization technology for multi-tenant environments (specifically, containers may affect resource quotas for other containers hosted on the same machine). Additional containment technologies, such as AppArmor, may be used to provide better isolation between containers, although this is not the case by default. For all these reasons, the choice of this virtualization technology is not recommended in production.

If your compute hosts do not have hardware support for virtualization, LXC will likely provide better performance than QEMU. In addition, if your guests must access specialized hardware, such as GPUs, this might be easier to achieve with LXC than other hypervisors.

Note: Some OpenStack Compute features might be missing when running with LXC as the hypervisor. See the hypervisor support matrix for details.

To enable LXC, ensure the following options are set in `/etc/nova/nova.conf` on all hosts running the `nova-compute` service.

```
compute_driver = libvirt.LibvirtDriver

[libvirt]
virt_type = lxc
```

On Ubuntu, enable LXC support in OpenStack by installing the `nova-compute-lxc` package.

VMware vSphere

Introduction

OpenStack Compute supports the VMware vSphere product family and enables access to advanced features such as vMotion, High Availability, and Dynamic Resource Scheduling (DRS).

This section describes how to configure VMware-based virtual machine images for launch. The VMware driver supports vCenter version 5.5.0 and later.

The VMware vCenter driver enables the `nova-compute` service to communicate with a VMware vCenter server that manages one or more ESX host clusters. The driver aggregates the ESX hosts in each cluster to present one large hypervisor entity for each cluster to the Compute scheduler. Because individual ESX hosts are not exposed to the

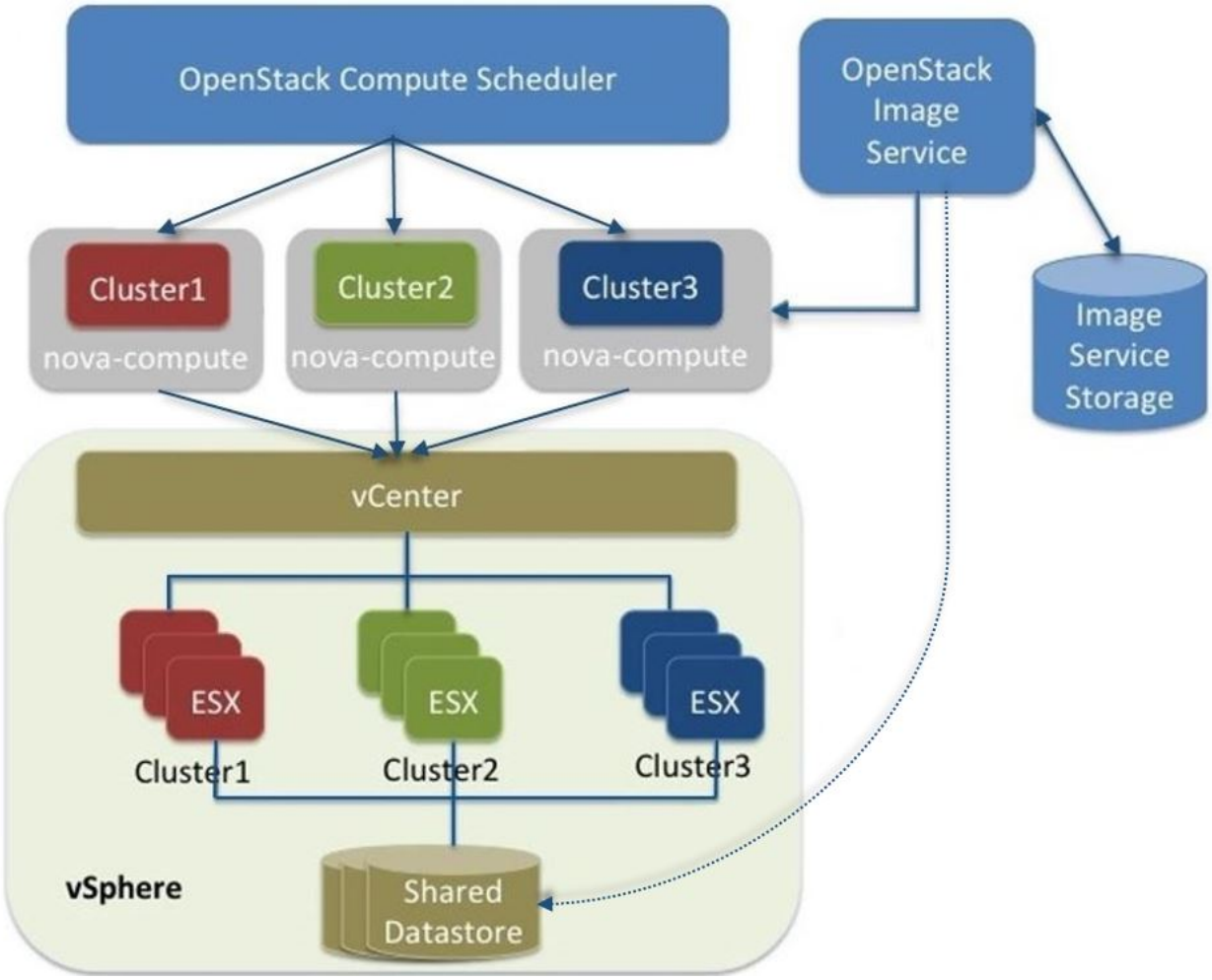
scheduler, Compute schedules to the granularity of clusters and vCenter uses DRS to select the actual ESX host within the cluster. When a virtual machine makes its way into a vCenter cluster, it can use all vSphere features.

The following sections describe how to configure the VMware vCenter driver.

High-level architecture

The following diagram shows a high-level view of the VMware driver architecture:

VMware driver architecture



As the figure shows, the OpenStack Compute Scheduler sees three hypervisors that each correspond to a cluster in vCenter. `nova-compute` contains the VMware driver. You can run with multiple `nova-compute` services. It is recommended to run with one `nova-compute` service per ESX cluster thus ensuring that while Compute schedules at the granularity of the `nova-compute` service it is also in effect able to schedule at the cluster level. In turn the VMware driver inside `nova-compute` interacts with the vCenter APIs to select an appropriate ESX host within the cluster. Internally, vCenter uses DRS for placement.

The VMware vCenter driver also interacts with the Image service to copy VMDK images from the Image service back-end store. The dotted line in the figure represents VMDK images being copied from the OpenStack Image service to

the vSphere data store. VMDK images are cached in the data store so the copy operation is only required the first time that the VMDK image is used.

After OpenStack boots a VM into a vSphere cluster, the VM becomes visible in vCenter and can access vSphere advanced features. At the same time, the VM is visible in the OpenStack dashboard and you can manage it as you would any other OpenStack VM. You can perform advanced vSphere operations in vCenter while you configure OpenStack resources such as VMs through the OpenStack dashboard.

The figure does not show how networking fits into the architecture. Both `nova-network` and the OpenStack Networking Service are supported. For details, see *Networking with VMware vSphere*.

Configuration overview

To get started with the VMware vCenter driver, complete the following high-level steps:

1. Configure vCenter. See *Prerequisites and limitations*.
2. Configure the VMware vCenter driver in the `nova.conf` file. See *VMware vCenter driver*.
3. Load desired VMDK images into the Image service. See *Images with VMware vSphere*.
4. Configure networking with either `nova-network` or the Networking service. See *Networking with VMware vSphere*.

Prerequisites and limitations

Use the following list to prepare a vSphere environment that runs with the VMware vCenter driver:

Copying VMDK files In vSphere 5.1, copying large image files (for example, 12 GB and greater) from the Image service can take a long time. To improve performance, VMware recommends that you upgrade to VMware vCenter Server 5.1 Update 1 or later. For more information, see the Release Notes.

DRS For any cluster that contains multiple ESX hosts, enable DRS and enable fully automated placement.

Shared storage Only shared storage is supported and data stores must be shared among all hosts in a cluster. It is recommended to remove data stores not intended for OpenStack from clusters being configured for OpenStack.

Clusters and data stores Do not use OpenStack clusters and data stores for other purposes. If you do, OpenStack displays incorrect usage information.

Networking The networking configuration depends on the desired networking model. See *Networking with VMware vSphere*.

Security groups If you use the VMware driver with OpenStack Networking and the NSX plug-in, security groups are supported. If you use `nova-network`, security groups are not supported.

Note: The NSX plug-in is the only plug-in that is validated for vSphere.

VNC The port range 5900 - 6105 (inclusive) is automatically enabled for VNC connections on every ESX host in all clusters under OpenStack control.

Note: In addition to the default VNC port numbers (5900 to 6000) specified in the above document, the following ports are also used: 6101, 6102, and 6105.

You must modify the ESXi firewall configuration to allow the VNC ports. Additionally, for the firewall modifications to persist after a reboot, you must create a custom vSphere Installation Bundle (VIB) which is then

installed onto the running ESXi host or added to a custom image profile used to install ESXi hosts. For details about how to create a VIB for persisting the firewall configuration modifications, see Knowledge Base.

Note: The VIB can be downloaded from openstack-vmwareapi-team/Tools.

To use multiple vCenter installations with OpenStack, each vCenter must be assigned to a separate availability zone. This is required as the OpenStack Block Storage VMDK driver does not currently work across multiple vCenter installations.

VMware vCenter service account

OpenStack integration requires a vCenter service account with the following minimum permissions. Apply the permissions to the `Datacenter` root object, and select the *Propagate to Child Objects* option.

Table 1: vCenter permissions tree

All Privileges			
	Datastore		
		Allocate space	
		Browse datastore	
		Low level file operation	
		Remove file	
	Extension		
		Register extension	
	Folder		
		Create folder	
	Host		
		Configuration	
			Maintenance
			Network configuration
			Storage partition configuration
	Network		
		Assign network	
	Resource		
		Assign virtual machine to resource pool	
		Migrate powered off virtual machine	
		Migrate powered on virtual machine	
	Virtual Machine		
		Configuration	
			Add existing disk
			Add new disk
			Add or remove device
			Advanced
			CPU count
			Change resource
			Disk change tracking
			Host USB device
			Memory

Continued on next page

Table 1 – continued from previous page

All Privileges			
			Modify device settings
			Raw device
			Remove disk
			Rename
			Set annotation
			Swapfile placement
		Interaction	
			Configure CD media
			Power Off
			Power On
			Reset
			Suspend
		Inventory	
			Create from existing
			Create new
			Move
			Remove
			Unregister
		Provisioning	
			Clone virtual machine
			Customize
			Create template from virtual machine
		Snapshot management	
			Create snapshot
			Remove snapshot
	Profile-driven storage		
		Profile-driven storage view	
	Sessions		
			Validate session
			View and stop sessions
	vApp		
		Export	
		Import	

VMware vCenter driver

Use the VMware vCenter driver (VMwareVCDriver) to connect OpenStack Compute with vCenter. This recommended configuration enables access through vCenter to advanced vSphere features like vMotion, High Availability, and Dynamic Resource Scheduling (DRS).

VMwareVCDriver configuration options

Add the following VMware-specific configuration options to the `nova.conf` file:

```
[DEFAULT]
compute_driver = vmwareapi.VMwareVCDriver

[vmware]
host_ip = <vCenter hostname or IP address>
host_username = <vCenter username>
host_password = <vCenter password>
cluster_name = <vCenter cluster name>
datastore_regex = <optional datastore regex>
```

Note:

- Clusters: The vCenter driver can support only a single cluster. Clusters and data stores used by the vCenter driver should not contain any VMs other than those created by the driver.
- Data stores: The `datastore_regex` setting specifies the data stores to use with Compute. For example, `datastore_regex="nas.*"` selects all the data stores that have a name starting with nas. If this line is omitted, Compute uses the first data store returned by the vSphere API. It is recommended not to use this field and instead remove data stores that are not intended for OpenStack.
- Reserved host memory: The `reserved_host_memory_mb` option value is 512 MB by default. However, VMware recommends that you set this option to 0 MB because the vCenter driver reports the effective memory available to the virtual machines.
- The vCenter driver generates instance name by instance ID. Instance name template is ignored.
- The minimum supported vCenter version is 5.5.0. Starting in the OpenStack Ocata release any version lower than 5.5.0 will be logged as a warning. In the OpenStack Pike release this will be enforced.

A `nova-compute` service can control one or more clusters containing multiple ESXi hosts, making `nova-compute` a critical service from a high availability perspective. Because the host that runs `nova-compute` can fail while the vCenter and ESX still run, you must protect the `nova-compute` service against host failures.

Note: Many `nova.conf` options are relevant to libvirt but do not apply to this driver.

Images with VMware vSphere

The vCenter driver supports images in the VMDK format. Disks in this format can be obtained from VMware Fusion or from an ESX environment. It is also possible to convert other formats, such as qcow2, to the VMDK format using the `qemu-img` utility. After a VMDK disk is available, load it into the Image service. Then, you can use it with the VMware vCenter driver. The following sections provide additional details on the supported disks and the commands used for conversion and upload.

Supported image types

Upload images to the OpenStack Image service in VMDK format. The following VMDK disk types are supported:

- `VMFS Flat Disks` (includes thin, thick, zeroedthick, and eagerzeroedthick). Note that once a VMFS thin disk is exported from VMFS to a non-VMFS location, like the OpenStack Image service, it becomes a preallocated flat disk. This impacts the transfer time from the Image service to the data store when the full preallocated flat disk, rather than the thin disk, must be transferred.

- `Monolithic Sparse disks`. Sparse disks get imported from the Image service into ESXi as thin provisioned disks. Monolithic Sparse disks can be obtained from VMware Fusion or can be created by converting from other virtual disk formats using the `qemu-img` utility.
- `Stream-optimized disks`. Stream-optimized disks are compressed sparse disks. They can be obtained from VMware vCenter/ESXi when exporting vm to ovf/ova template.

The following table shows the `vmware_disktype` property that applies to each of the supported VMDK disk types:

Table 2: OpenStack Image service disk type settings

vmware_disktype property	VMDK disk type
sparse	Monolithic Sparse
thin	VMFS flat, thin provisioned
preallocated (default)	VMFS flat, thick/zeroedthick/eagerzeroedthick
streamOptimized	Compressed Sparse

The `vmware_disktype` property is set when an image is loaded into the Image service. For example, the following command creates a Monolithic Sparse image by setting `vmware_disktype` to `sparse`:

```
$ openstack image create \
  --disk-format vmdk \
  --container-format bare \
  --property vmware_disktype="sparse" \
  --property vmware_ostype="ubuntu64Guest" \
  ubuntu-sparse < ubuntuLTS-sparse.vmdk
```

Note: Specifying `thin` does not provide any advantage over `preallocated` with the current version of the driver. Future versions might restore the thin properties of the disk after it is downloaded to a vSphere data store.

The following table shows the `vmware_ostype` property that applies to each of the supported guest OS:

Note: If a glance image has a `vmware_ostype` property which does not correspond to a valid VMware guestId, VM creation will fail, and a warning will be logged.

Table 3: OpenStack Image service OS type settings

vmware_ostype property	Retail Name
asianux3_64Guest	Asianux Server 3 (64 bit)
asianux3Guest	Asianux Server 3
asianux4_64Guest	Asianux Server 4 (64 bit)
asianux4Guest	Asianux Server 4
darwin64Guest	Darwin 64 bit
darwinGuest	Darwin
debian4_64Guest	Debian GNU/Linux 4 (64 bit)
debian4Guest	Debian GNU/Linux 4
debian5_64Guest	Debian GNU/Linux 5 (64 bit)
debian5Guest	Debian GNU/Linux 5
dosGuest	MS-DOS
freebsd64Guest	FreeBSD x64
freebsdGuest	FreeBSD
mandrivaGuest	Mandriva Linux

Continued on next page

Table 3 – continued from previous page

vmware_ostype property	Retail Name
netware4Guest	Novell NetWare 4
netware5Guest	Novell NetWare 5.1
netware6Guest	Novell NetWare 6.x
nld9Guest	Novell Linux Desktop 9
oesGuest	Open Enterprise Server
openServer5Guest	SCO OpenServer 5
openServer6Guest	SCO OpenServer 6
opensuse64Guest	openSUSE (64 bit)
opensuseGuest	openSUSE
os2Guest	OS/2
other24xLinux64Guest	Linux 2.4x Kernel (64 bit) (experimental)
other24xLinuxGuest	Linux 2.4x Kernel
other26xLinux64Guest	Linux 2.6x Kernel (64 bit) (experimental)
other26xLinuxGuest	Linux 2.6x Kernel (experimental)
otherGuest	Other Operating System
otherGuest64	Other Operating System (64 bit) (experimental)
otherLinux64Guest	Linux (64 bit) (experimental)
otherLinuxGuest	Other Linux
redhatGuest	Red Hat Linux 2.1
rhel2Guest	Red Hat Enterprise Linux 2
rhel3_64Guest	Red Hat Enterprise Linux 3 (64 bit)
rhel3Guest	Red Hat Enterprise Linux 3
rhel4_64Guest	Red Hat Enterprise Linux 4 (64 bit)
rhel4Guest	Red Hat Enterprise Linux 4
rhel5_64Guest	Red Hat Enterprise Linux 5 (64 bit) (experimental)
rhel5Guest	Red Hat Enterprise Linux 5
rhel6_64Guest	Red Hat Enterprise Linux 6 (64 bit)
rhel6Guest	Red Hat Enterprise Linux 6
sjdsGuest	Sun Java Desktop System
sles10_64Guest	SUSE Linux Enterprise Server 10 (64 bit) (experimental)
sles10Guest	SUSE Linux Enterprise Server 10
sles11_64Guest	SUSE Linux Enterprise Server 11 (64 bit)
sles11Guest	SUSE Linux Enterprise Server 11
sles64Guest	SUSE Linux Enterprise Server 9 (64 bit)
slesGuest	SUSE Linux Enterprise Server 9
solaris10_64Guest	Solaris 10 (64 bit) (experimental)
solaris10Guest	Solaris 10 (32 bit) (experimental)
solaris6Guest	Solaris 6
solaris7Guest	Solaris 7
solaris8Guest	Solaris 8
solaris9Guest	Solaris 9
suse64Guest	SUSE Linux (64 bit)
suseGuest	SUSE Linux
turboLinux64Guest	Turbolinux (64 bit)
turboLinuxGuest	Turbolinux
ubuntu64Guest	Ubuntu Linux (64 bit)
ubuntuGuest	Ubuntu Linux
unixWare7Guest	SCO UnixWare 7
win2000AdvServGuest	Windows 2000 Advanced Server

Continued on next page

Table 3 – continued from previous page

vmware_ostype property	Retail Name
win2000ProGuest	Windows 2000 Professional
win2000ServGuest	Windows 2000 Server
win31Guest	Windows 3.1
win95Guest	Windows 95
win98Guest	Windows 98
windows7_64Guest	Windows 7 (64 bit)
windows7Guest	Windows 7
windows7Server64Guest	Windows Server 2008 R2 (64 bit)
winLonghorn64Guest	Windows Longhorn (64 bit) (experimental)
winLonghornGuest	Windows Longhorn (experimental)
winMeGuest	Windows Millennium Edition
winNetBusinessGuest	Windows Small Business Server 2003
winNetDatacenter64Guest	Windows Server 2003, Datacenter Edition (64 bit) (experimental)
winNetDatacenterGuest	Windows Server 2003, Datacenter Edition
winNetEnterprise64Guest	Windows Server 2003, Enterprise Edition (64 bit)
winNetEnterpriseGuest	Windows Server 2003, Enterprise Edition
winNetStandard64Guest	Windows Server 2003, Standard Edition (64 bit)
winNetEnterpriseGuest	Windows Server 2003, Enterprise Edition
winNetStandard64Guest	Windows Server 2003, Standard Edition (64 bit)
winNetStandardGuest	Windows Server 2003, Standard Edition
winNetWebGuest	Windows Server 2003, Web Edition
winNTGuest	Windows NT 4
winVista64Guest	Windows Vista (64 bit)
winVistaGuest	Windows Vista
winXPHomeGuest	Windows XP Home Edition
winXPPro64Guest	Windows XP Professional Edition (64 bit)
winXPProGuest	Windows XP Professional

Convert and load images

Using the `qemu-img` utility, disk images in several formats (such as, qcow2) can be converted to the VMDK format.

For example, the following command can be used to convert a qcow2 Ubuntu Trusty cloud image:

```
$ qemu-img convert -f qcow2 ~/Downloads/trusty-server-cloudimg-amd64-disk1.img \
  -O vmdk trusty-server-cloudimg-amd64-disk1.vmdk
```

VMDK disks converted through `qemu-img` are `always` monolithic sparse VMDK disks with an IDE adapter type. Using the previous example of the Ubuntu Trusty image after the `qemu-img` conversion, the command to upload the VMDK disk should be something like:

```
$ openstack image create \
  --container-format bare --disk-format vmdk \
  --property vmware_disktype="sparse" \
  --property vmware_adaptertype="ide" \
  trusty-cloud < trusty-server-cloudimg-amd64-disk1.vmdk
```

Note that the `vmware_disktype` is set to `sparse` and the `vmware_adaptertype` is set to `ide` in the previous command.

If the image did not come from the `qemu-img` utility, the `vmware_disktype` and `vmware_adaptertype` might be different. To determine the image adapter type from an image file, use the following command and look for

the `ddb.adapterType=` line:

```
$ head -20 <vmdk file name>
```

Assuming a preallocated disk type and an iSCSI lsiLogic adapter type, the following command uploads the VMDK disk:

```
$ openstack image create \
  --disk-format vmdk \
  --container-format bare \
  --property vmware_adaptertype="lsiLogic" \
  --property vmware_disktype="preallocated" \
  --property vmware_ostype="ubuntu64Guest" \
  ubuntu-thick-scsi < ubuntuLTS-flat.vmdk
```

Currently, OS boot VMDK disks with an IDE adapter type cannot be attached to a virtual SCSI controller and likewise disks with one of the SCSI adapter types (such as, busLogic, lsiLogic, lsiLogicsas, paraVirtual) cannot be attached to the IDE controller. Therefore, as the previous examples show, it is important to set the `vmware_adaptertype` property correctly. The default adapter type is lsiLogic, which is SCSI, so you can omit the `vmware_adaptertype` property if you are certain that the image adapter type is lsiLogic.

Tag VMware images

In a mixed hypervisor environment, OpenStack Compute uses the `hypervisor_type` tag to match images to the correct hypervisor type. For VMware images, set the hypervisor type to `vmware`. Other valid hypervisor types include: `hyperv`, `ironic`, `lxc`, `qemu`, `uml`, and `xen`. Note that `qemu` is used for both QEMU and KVM hypervisor types.

```
$ openstack image create \
  --disk-format vmdk \
  --container-format bare \
  --property vmware_adaptertype="lsiLogic" \
  --property vmware_disktype="preallocated" \
  --property hypervisor_type="vmware" \
  --property vmware_ostype="ubuntu64Guest" \
  ubuntu-thick-scsi < ubuntuLTS-flat.vmdk
```

Optimize images

Monolithic Sparse disks are considerably faster to download but have the overhead of an additional conversion step. When imported into ESX, sparse disks get converted to VMFS flat thin provisioned disks. The download and conversion steps only affect the first launched instance that uses the sparse disk image. The converted disk image is cached, so subsequent instances that use this disk image can simply use the cached version.

To avoid the conversion step (at the cost of longer download times) consider converting sparse disks to thin provisioned or preallocated disks before loading them into the Image service.

Use one of the following tools to pre-convert sparse disks.

vSphere CLI tools Sometimes called the remote CLI or rCLI.

Assuming that the sparse disk is made available on a data store accessible by an ESX host, the following command converts it to preallocated format:

```
vmkfstools --server=ip_of_some_ESX_host -i \
  /vmfs/volumes/datastore1/sparse.vmdk \
  /vmfs/volumes/datastore1/converted.vmdk
```

Note that the vifs tool from the same CLI package can be used to upload the disk to be converted. The vifs tool can also be used to download the converted disk if necessary.

vmkfstools directly on the ESX host If the SSH service is enabled on an ESX host, the sparse disk can be uploaded to the ESX data store through scp and the vmkfstools local to the ESX host can use used to perform the conversion. After you log in to the host through ssh, run this command:

```
vmkfstools -i /vmfs/volumes/datastore1/sparse.vmdk /vmfs/volumes/datastore1/
↪converted.vmdk
```

vmware-vdiskmanager `vmware-vdiskmanager` is a utility that comes bundled with VMware Fusion and VMware Workstation. The following example converts a sparse disk to preallocated format:

```
'/Applications/VMware Fusion.app/Contents/Library/vmware-vdiskmanager' -r sparse.
↪vmdk -t 4 converted.vmdk
```

In the previous cases, the converted vmdk is actually a pair of files:

- The descriptor file `converted.vmdk`.
- The actual virtual disk data file `converted-flat.vmdk`.

The file to be uploaded to the Image service is `converted-flat.vmdk`.

Image handling

The ESX hypervisor requires a copy of the VMDK file in order to boot up a virtual machine. As a result, the vCenter OpenStack Compute driver must download the VMDK via HTTP from the Image service to a data store that is visible to the hypervisor. To optimize this process, the first time a VMDK file is used, it gets cached in the data store. A cached image is stored in a folder named after the image ID. Subsequent virtual machines that need the VMDK use the cached version and dont have to copy the file again from the Image service.

Even with a cached VMDK, there is still a copy operation from the cache location to the hypervisor file directory in the shared data store. To avoid this copy, boot the image in linked_clone mode. To learn how to enable this mode, see *Configuration reference*.

Note: You can also use the `img_linked_clone` property (or legacy property `vmware_linked_clone`) in the Image service to override the linked_clone mode on a per-image basis.

If spawning a virtual machine image from ISO with a VMDK disk, the image is created and attached to the virtual machine as a blank disk. In that case `img_linked_clone` property for the image is just ignored.

If multiple compute nodes are running on the same host, or have a shared file system, you can enable them to use the same cache folder on the back-end data store. To configure this action, set the `cache_prefix` option in the `nova.conf` file. Its value stands for the name prefix of the folder where cached images are stored.

Note: This can take effect only if compute nodes are running on the same host, or have a shared file system.

You can automatically purge unused images after a specified period of time. To configure this action, set these options in the `DEFAULT` section in the `nova.conf` file:

remove_unused_base_images Set this option to `True` to specify that unused images should be removed after the duration specified in the `remove_unused_original_minimum_age_seconds` option. The default is `True`.

remove_unused_original_minimum_age_seconds Specifies the duration in seconds after which an unused image is purged from the cache. The default is `86400` (24 hours).

Networking with VMware vSphere

The VMware driver supports networking with the `nova-network` service or the Networking Service. Depending on your installation, complete these configuration steps before you provision VMs:

1. **The nova-network service with the FlatManager or FlatDHCPManager**. Create a port group with the same name as the `flat_network_bridge` value in the `nova.conf` file. The default value is `br100`. If you specify another value, the new value must be a valid Linux bridge identifier that adheres to Linux bridge naming conventions.

   All VM NICs are attached to this port group.

   Ensure that the flat interface of the node that runs the `nova-network` service has a path to this network.

   Note: When configuring the port binding for this port group in vCenter, specify `ephemeral` for the port binding type. For more information, see Choosing a port binding type in ESX/ESXi in the VMware Knowledge Base.

   ---

2. **The nova-network service with the VlanManager**. Set the `vlan_interface` configuration option to match the ESX host interface that handles VLAN-tagged VM traffic.

   OpenStack Compute automatically creates the corresponding port groups.

3. If you are using the OpenStack Networking Service: Before provisioning VMs, create a port group with the same name as the `vmware.integration_bridge` value in `nova.conf` (default is `br-int`). All VM NICs are attached to this port group for management by the OpenStack Networking plug-in.

Volumes with VMware vSphere

The VMware driver supports attaching volumes from the Block Storage service. The VMware VMDK driver for OpenStack Block Storage is recommended and should be used for managing volumes based on vSphere data stores. For more information about the VMware VMDK driver, see Cinders manual on the VMDK Driver (TODO: this has not yet been imported and published). Also an iSCSI volume driver provides limited support and can be used only for attachments.

Configuration reference

To customize the VMware driver, use the configuration option settings below.

Table 4: Description of VMware configuration options

Configuration option = Default value	Description
[vmware]	
api_retry_count = 10	(Integer) Number of times VMware vCenter server API must be retried on connection failures, e.g. socket error, etc.
ca_file = None	(String) Specifies the CA bundle file to be used in verifying the vCenter server certificate.
cache_prefix = None	(String) This option adds a prefix to the folder where cached images are stored This is not the full path - just a folder prefix. This should only be used when a datastore cache is shared between compute nodes. --- **Note:** This should only be used when the compute nodes are running on same host or they have a shared file system. --- Possible values: • Any string representing the cache prefix to the folder
cluster_name = None	(String) Name of a VMware Cluster ComputeResource.
console_delay_seconds = None	(Integer) Set this value if affected by an increased network latency causing repeated characters when typing in a remote console.
datastore_regex = None	(String) Regular expression pattern to match the name of datastore. The datastore_regex setting specifies the datastores to use with Compute. For example, datastore_regex=nas.* selects all the data stores that have a name starting with nas. --- **Note:** If no regex is given, it just picks the datastore with the most freespace. --- Possible values: • Any matching regular expression to a datastore must be given
host_ip = None	(String) Hostname or IP address for connection to VMware vCenter host.
host_password = None	(String) Password for connection to VMware vCenter host.
host_port = 443	(Port number) Port for connection to VMware vCenter host.
host_username = None	(String) Username for connection to VMware vCenter host.
insecure = False	(Boolean) If true, the vCenter server certificate is not verified. If false, then the default CA truststore is used for verification. Related options: • ca_file: This option is ignored if ca_file is set.
integration_bridge = None	(String) This option should be configured only when using the NSX-MH Neutron plugin. This is the name of the integration bridge on the ESXi server or host. This should not be set for any other Neutron plugin. Hence the default value is not set.

Troubleshooting

Operators can troubleshoot VMware specific failures by correlating OpenStack logs to vCenter logs. Every RPC call which is made by an OpenStack driver has an `opID` which can be traced in the vCenter logs. For example consider the following excerpt from a `nova-compute` log:

```
Aug 15 07:31:09 localhost nova-compute[16683]: DEBUG oslo_vmware.service [-] Invoking␣
↪Folder.CreateVM_Task with opID=oslo.vmware-debb6064-690e-45ac-b0ae-1b94a9638d1f {
↪{(pid=16683) request_handler /opt/stack/oslo.vmware/oslo_vmware/service.py:355}}
```

In this case the `opID` is `oslo.vmware-debb6064-690e-45ac-b0ae-1b94a9638d1f` and we can grep the vCenter log (usually `/var/log/vmware/vpxd/vpxd.log`) for it to find if anything went wrong with the `CreateVM` operation.

Hyper-V virtualization platform

Todo: This is really installation guide material and should probably be moved.

It is possible to use Hyper-V as a compute node within an OpenStack Deployment. The `nova-compute` service runs as `openstack-compute`, a 32-bit service directly upon the Windows platform with the Hyper-V role enabled. The necessary Python components as well as the `nova-compute` service are installed directly onto the Windows platform. Windows Clustering Services are not needed for functionality within the OpenStack infrastructure. The use of the Windows Server 2012 platform is recommend for the best experience and is the platform for active development. The following Windows platforms have been tested as compute nodes:

- Windows Server 2012
- Windows Server 2012 R2 Server and Core (with the Hyper-V role enabled)
- Hyper-V Server

Hyper-V configuration

The only OpenStack services required on a Hyper-V node are `nova-compute` and `neutron-hyperv-agent`. Regarding the resources needed for this host you have to consider that Hyper-V will require 16 GB - 20 GB of disk space for the OS itself, including updates. Two NICs are required, one connected to the management network and one to the guest data network.

The following sections discuss how to prepare the Windows Hyper-V node for operation as an OpenStack compute node. Unless stated otherwise, any configuration information should work for the Windows 2012 and 2012 R2 platforms.

Local storage considerations

The Hyper-V compute node needs to have ample storage for storing the virtual machine images running on the compute nodes. You may use a single volume for all, or partition it into an OS volume and VM volume.

Configure NTP

Network time services must be configured to ensure proper operation of the OpenStack nodes. To set network time on your Windows host you must run the following commands:

```
C:\>net stop w32time
C:\>w32tm /config "/manualpeerlist:pool.ntp.org,0x8" /syncfromflags:MANUAL
C:\>net start w32time
```

Keep in mind that the node will have to be time synchronized with the other nodes of your OpenStack environment, so it is important to use the same NTP server. Note that in case of an Active Directory environment, you may do this only for the AD Domain Controller.

Configure Hyper-V virtual switching

Information regarding the Hyper-V virtual Switch can be found in the Hyper-V Virtual Switch Overview.

To quickly enable an interface to be used as a Virtual Interface the following PowerShell may be used:

```
PS C:\> $if = Get-NetIPAddress -IPAddress 192* | Get-NetIPInterface
PS C:\> New-VMSwitch -NetAdapterName $if.ifAlias -Name YOUR_BRIDGE_NAME -
↪AllowManagementOS $false
```

Note: It is very important to make sure that when you are using a Hyper-V node with only 1 NIC the -AllowManagementOS option is set on `True`, otherwise you will lose connectivity to the Hyper-V node.

Enable iSCSI initiator service

To prepare the Hyper-V node to be able to attach to volumes provided by cinder you must first make sure the Windows iSCSI initiator service is running and started automatically.

```
PS C:\> Set-Service -Name MSiSCSI -StartupType Automatic
PS C:\> Start-Service MSiSCSI
```

Configure shared nothing live migration

Detailed information on the configuration of live migration can be found in this guide

The following outlines the steps of shared nothing live migration.

1. The target host ensures that live migration is enabled and properly configured in Hyper-V.
2. The target host checks if the image to be migrated requires a base VHD and pulls it from the Image service if not already available on the target host.
3. The source host ensures that live migration is enabled and properly configured in Hyper-V.
4. The source host initiates a Hyper-V live migration.
5. The source host communicates to the manager the outcome of the operation.

The following three configuration options are needed in order to support Hyper-V live migration and must be added to your `nova.conf` on the Hyper-V compute node:

- This is needed to support shared nothing Hyper-V live migrations. It is used in `nova/compute/manager.py`.

  ```
  instances_shared_storage = False
  ```

- This flag is needed to support live migration to hosts with different CPU features. This flag is checked during instance creation in order to limit the CPU features used by the VM.

```
limit_cpu_features = True
```

- This option is used to specify where instances are stored on disk.

```
instances_path = DRIVELETTER:\PATH\TO\YOUR\INSTANCES
```

Additional Requirements:

- Hyper-V 2012 R2 or Windows Server 2012 R2 with Hyper-V role enabled
- A Windows domain controller with the Hyper-V compute nodes as domain members
- The instances_path command-line option/flag needs to be the same on all hosts
- The `openstack-compute` service deployed with the setup must run with domain credentials. You can set the service credentials with:

```
C:\>sc config openstack-compute obj="DOMAIN\username" password="password"
```

How to setup live migration on Hyper-V

To enable shared nothing live migration, run the 3 instructions below on each Hyper-V host:

```
PS C:\> Enable-VMMigration
PS C:\> Set-VMMigrationNetwork IP_ADDRESS
PS C:\> Set-VMHost -VirtualMachineMigrationAuthenticationTypeKerberos
```

Note: Replace the `IP_ADDRESS` with the address of the interface which will provide live migration.

Additional Reading

This article clarifies the various live migration options in Hyper-V:

Hyper-V Live Migration of Yesterday

Install nova-compute using OpenStack Hyper-V installer

In case you want to avoid all the manual setup, you can use Cloudbase Solutions installer. You can find it here:

HyperVNovaCompute_Beta download

The tool installs an independent Python environment in order to avoid conflicts with existing applications, and dynamically generates a `nova.conf` file based on the parameters provided by you.

The tool can also be used for an automated and unattended mode for deployments on a massive number of servers. More details about how to use the installer and its features can be found here:

Cloudbase

Requirements

Python

Python 2.7 32bit must be installed as most of the libraries are not working properly on the 64bit version.

Setting up Python prerequisites

1. Download and install Python 2.7 using the MSI installer from here:

   python-2.7.3.msi download

```
PS C:\> $src = "https://www.python.org/ftp/python/2.7.3/python-2.7.3.msi"
PS C:\> $dest = "$env:temp\python-2.7.3.msi"
PS C:\> Invoke-WebRequest -Uri $src -OutFile $dest
PS C:\> Unblock-File $dest
PS C:\> Start-Process $dest
```

2. Make sure that the `Python` and `Python\Scripts` paths are set up in the `PATH` environment variable.

```
PS C:\> $oldPath = [System.Environment]::GetEnvironmentVariable("Path")
PS C:\> $newPath = $oldPath + ";C:\python27\;C:\python27\Scripts\"
PS C:\> [System.Environment]::SetEnvironmentVariable("Path", $newPath, [System.
↪EnvironmentVariableTarget]::User
```

Python dependencies

The following packages need to be downloaded and manually installed:

`setuptools` https://pypi.python.org/packages/2.7/s/setuptools/setuptools-0.6c11.win32-py2.7.exe

`pip` https://pip.pypa.io/en/latest/installing/

`PyMySQL` http://codegood.com/download/10/

`PyWin32` https://sourceforge.net/projects/pywin32/files/pywin32/Build%20217/pywin32-217.win32-py2.7.exe

`Greenlet` http://www.lfd.uci.edu/~gohlke/pythonlibs/#greenlet

`PyCryto` http://www.voidspace.org.uk/downloads/pycrypto26/pycrypto-2.6.win32-py2.7.exe

The following packages must be installed with pip:

- `ecdsa`
- `amqp`
- `wmi`

```
PS C:\> pip install ecdsa
PS C:\> pip install amqp
PS C:\> pip install wmi
```

Other dependencies

`qemu-img` is required for some of the image related operations. You can get it from here: http://qemu.weilnetz.de/. You must make sure that the `qemu-img` path is set in the PATH environment variable.

Some Python packages need to be compiled, so you may use MinGW or Visual Studio. You can get MinGW from here: http://sourceforge.net/projects/mingw/. You must configure which compiler is to be used for this purpose by using the `distutils.cfg` file in `$Python27\Lib\distutils`, which can contain:

```
[build]
compiler = mingw32
```

As a last step for setting up MinGW, make sure that the MinGW binaries directories are set up in PATH.

Install nova-compute

Download the nova code

1. Use Git to download the necessary source code. The installer to run Git on Windows can be downloaded here:

   https://github.com/msysgit/msysgit/releases/download/Git-1.9.2-preview20140411/Git-1.9.2-preview20140411.exe

2. Download the installer. Once the download is complete, run the installer and follow the prompts in the installation wizard. The default should be acceptable for the purposes of this guide.

   ```
   PS C:\> $src = "https://github.com/msysgit/msysgit/releases/download/Git-1.9.2-
   ↪preview20140411/Git-1.9.2-preview20140411.exe"
   PS C:\> $dest = "$env:temp\Git-1.9.2-preview20140411.exe"
   PS C:\> Invoke-WebRequest -Uri $src -OutFile $dest
   PS C:\> Unblock-File $dest
   PS C:\> Start-Process $dest
   ```

3. Run the following to clone the nova code.

   ```
   PS C:\> git.exe clone https://opendev.org/openstack/nova
   ```

Install nova-compute service

To install `nova-compute`, run:

```
PS C:\> cd c:\nova
PS C:\> python setup.py install
```

Configure nova-compute

The `nova.conf` file must be placed in `C:\etc\nova` for running OpenStack on Hyper-V. Below is a sample `nova.conf` for Windows:

```
[DEFAULT]
auth_strategy = keystone
image_service = nova.image.glance.GlanceImageService
compute_driver = nova.virt.hyperv.driver.HyperVDriver
volume_api_class = nova.volume.cinder.API
fake_network = true
instances_path = C:\Program Files (x86)\OpenStack\Instances
use_cow_images = true
```

(continues on next page)

(continued from previous page)

```
force_config_drive = false
injected_network_template = C:\Program Files (x86)\OpenStack\Nova\etc\interfaces.
↪template
policy_file = C:\Program Files (x86)\OpenStack\Nova\etc\policy.json
mkisofs_cmd = C:\Program Files (x86)\OpenStack\Nova\bin\mkisofs.exe
allow_resize_to_same_host = true
running_deleted_instance_action = reap
running_deleted_instance_poll_interval = 120
resize_confirm_window = 5
resume_guests_state_on_host_boot = true
rpc_response_timeout = 1800
lock_path = C:\Program Files (x86)\OpenStack\Log\
rpc_backend = nova.openstack.common.rpc.impl_kombu
rabbit_host = IP_ADDRESS
rabbit_port = 5672
rabbit_userid = guest
rabbit_password = Passw0rd
logdir = C:\Program Files (x86)\OpenStack\Log\
logfile = nova-compute.log
instance_usage_audit = true
instance_usage_audit_period = hour
use_neutron = True
[glance]
api_servers = http://IP_ADDRESS:9292
[neutron]
endpoint_override = http://IP_ADDRESS:9696
auth_strategy = keystone
project_name = service
username = neutron
password = Passw0rd
auth_url = http://IP_ADDRESS:5000/v3
auth_type = password
[hyperv]
vswitch_name = newVSwitch0
limit_cpu_features = false
config_drive_inject_password = false
qemu_img_cmd = C:\Program Files (x86)\OpenStack\Nova\bin\qemu-img.exe
config_drive_cdrom = true
dynamic_memory_ratio = 1
enable_instance_metrics_collection = true
[rdp]
enabled = true
html5_proxy_base_url = https://IP_ADDRESS:4430
```

Prepare images for use with Hyper-V

Hyper-V currently supports only the VHD and VHDX file format for virtual machine instances. Detailed instructions for installing virtual machines on Hyper-V can be found here:

Create Virtual Machines

Once you have successfully created a virtual machine, you can then upload the image to *glance* using the *openstack-client*:

```
PS C:\> openstack image create --name "VM_IMAGE_NAME" --property hypervisor_
↪type=hyperv --public \
          --container-format bare --disk-format vhd
```

Note: VHD and VHDX files sizes can be bigger than their maximum internal size, as such you need to boot instances using a flavor with a slightly bigger disk size than the internal size of the disk file. To create VHDs, use the following PowerShell cmdlet:

```
PS C:\> New-VHD DISK_NAME.vhd -SizeBytes VHD_SIZE
```

Inject interfaces and routes

The `interfaces.template` file describes the network interfaces and routes available on your system and how to activate them. You can specify the location of the file with the `injected_network_template` configuration option in `/etc/nova/nova.conf`.

```
injected_network_template = PATH_TO_FILE
```

A default template exists in `nova/virt/interfaces.template`.

Run Compute with Hyper-V

To start the `nova-compute` service, run this command from a console in the Windows server:

```
PS C:\> C:\Python27\python.exe c:\Python27\Scripts\nova-compute --config-file␣
↪c:\etc\nova\nova.conf
```

Troubleshoot Hyper-V configuration

- I ran the **`nova-manage service list`** command from my controller; however, Im not seeing smiley faces for Hyper-V compute nodes, what do I do?

  Verify that you are synchronized with a network time source. For instructions about how to configure NTP on your Hyper-V compute node, see *Configure NTP*.

- How do I restart the compute service?

  ```
  PS C:\> net stop nova-compute && net start nova-compute
  ```

- How do I restart the iSCSI initiator service?

  ```
  PS C:\> net stop msiscsi && net start msiscsi
  ```

Virtuozzo

Virtuozzo 7.0.0 (or newer), or its community edition OpenVZ, provides both types of virtualization: Kernel Virtual Machines and OS Containers. The type of instance to span is chosen depending on the `hw_vm_type` property of an image.

Note: Some OpenStack Compute features may be missing when running with Virtuozzo as the hypervisor. See *Feature Support Matrix* for details.

To enable Virtuozzo Containers, set the following options in `/etc/nova/nova.conf` on all hosts running the `nova-compute` service.

```
compute_driver = libvirt.LibvirtDriver
force_raw_images = False

[libvirt]
virt_type = parallels
images_type = ploop
connection_uri = parallels:///system
inject_partition = -2
```

To enable Virtuozzo Virtual Machines, set the following options in `/etc/nova/nova.conf` on all hosts running the `nova-compute` service.

```
compute_driver = libvirt.LibvirtDriver

[libvirt]
virt_type = parallels
images_type = qcow2
connection_uri = parallels:///system
```

PowerVM

Introduction

OpenStack Compute supports the PowerVM hypervisor through NovaLink. In the NovaLink architecture, a thin NovaLink virtual machine running on the Power system manages virtualization for that system. The `nova-compute` service can be installed on the NovaLink virtual machine and configured to use the PowerVM compute driver. No external management element (e.g. Hardware Management Console) is needed.

Configuration

In order to function properly, the `nova-compute` service must be executed by a member of the `pvm_admin` group. Use the `usermod` command to add the user. For example, to add the `stacker` user to the `pvm_admin` group, execute:

```
sudo usermod -a -G pvm_admin stacker
```

The user must re-login for the change to take effect.

To enable the PowerVM compute driver, set the following configuration option in the `/etc/nova/nova.conf` file:

```
[Default]
compute_driver = powervm.PowerVMDriver
```

The PowerVM driver supports two types of storage for ephemeral disks: `localdisk` or `ssp`. If `localdisk` is selected, you must specify which volume group should be used. E.g.:

```
[powervm]
disk_driver = localdisk
volume_group_name = openstackvg
```

Note: Using the `rootvg` volume group is strongly discouraged since `rootvg` is used by the management partition and filling this will cause failures.

The PowerVM driver also supports configuring the default amount of physical processor compute power (known as proc units) which will be given to each vCPU. This value will be used if the requested flavor does not specify the `powervm:proc_units` extra-spec. A factor value of 1.0 means a whole physical processor, whereas 0.05 means 1/20th of a physical processor. E.g.:

```
[powervm]
proc_units_factor = 0.1
```

Volume Support

Volume support is provided for the PowerVM virt driver via Cinder. Currently, the only supported volume protocol is vSCSI Fibre Channel. Attach, detach, and extend are the operations supported by the PowerVM vSCSI FC volume adapter. Boot from volume is not yet supported.

zVM

z/VM System Requirements

- The appropriate APARs installed, the current list of which can be found: z/VM OpenStack Cloud Information (http://www.vm.ibm.com/sysman/osmntlvl.html).

Note: IBM z Systems hardware requirements are based on both the applications and the load on the system.

Active Engine Guide

Active engine is used as an initial configuration and management tool during deployed machine startup. Currently the z/VM driver uses `zvmguestconfigure` and `cloud-init` as a two stage active engine.

Installation and Configuration of zvmguestconfigure

Cloudlib4zvm supports initiating changes to a Linux on z Systems virtual machine while Linux is shut down or the virtual machine is logged off. The changes to Linux are implemented using an activation engine (AE) that is run when Linux is booted the next time. The first active engine, `zvmguestconfigure`, must be installed in the Linux on z Systems virtual server so it can process change request files transmitted by the cloudlib4zvm service to the reader of the virtual machine as a class X file.

Note: An additional activation engine, cloud-init, should be installed to handle OpenStack related tailoring of the system. The cloud-init AE relies on tailoring performed by `zvmguestconfigure`.

Installation and Configuration of cloud-init

OpenStack uses cloud-init as its activation engine. Some Linux distributions include cloud-init either already installed or available to be installed. If your distribution does not include cloud-init, you can download the code from https://launchpad.net/cloud-init/+download. After installation, if you issue the following shell command and no errors occur, cloud-init is installed correctly:

```
cloud-init init --local
```

Installation and configuration of cloud-init differs among different Linux distributions, and cloud-init source code may change. This section provides general information, but you may have to tailor cloud-init to meet the needs of your Linux distribution. You can find a community-maintained list of dependencies at http://ibm.biz/cloudinitLoZ.

As of the Rocky release, the z/VM OpenStack support has been tested with cloud-init 0.7.4 and 0.7.5 for RHEL6.x and SLES11.x, 0.7.6 for RHEL7.x and SLES12.x, and 0.7.8 for Ubuntu 16.04.

During cloud-init installation, some dependency packages may be required. You can use zypper and python setuptools to easily resolve these dependencies. See https://pypi.python.org/pypi/setuptools for more information.

Image guide

This guideline will describe the requirements and steps to create and configure images for use with z/VM.

Image Requirements

- The following Linux distributions are supported for deploy:
  - RHEL 6.2, 6.3, 6.4, 6.5, 6.6, and 6.7
  - RHEL 7.0, 7.1 and 7.2
  - SLES 11.2, 11.3, and 11.4
  - SLES 12 and SLES 12.1
  - Ubuntu 16.04
- A supported root disk type for snapshot/spawn. The following are supported:
  - FBA
  - ECKD
- An image deployed on a compute node must match the disk type supported by that compute node, as configured by the `zvm_diskpool_type` property in the zvmsdk.conf configuration file in zvm cloud connector A compute node supports deployment on either an ECKD or FBA image, but not both at the same time. If you wish to switch image types, you need to change the `zvm_diskpool_type` and `zvm_diskpool` properties in the zvmsdk.conf file, accordingly. Then restart the nova-compute service to make the changes take effect.
- If you deploy an instance with an ephemeral disk, both the root disk and the ephemeral disk will be created with the disk type that was specified by `zvm_diskpool_type` property in the zvmsdk.conf file. That property can specify either ECKD or FBA.
- The network interfaces must be IPv4 interfaces.
- Image names should be restricted to the UTF-8 subset, which corresponds to the ASCII character set. In addition, special characters such as `/`, `\`, `$`, `%`, `@` should not be used. For the FBA disk type vm, capture and deploy is supported only for an FBA disk with a single partition. Capture and deploy is not supported for the FBA disk type vm on a CMS formatted FBA disk.

- The virtual server/Linux instance used as the source of the new image should meet the following criteria:
  1. The root filesystem must not be on a logical volume.
  2. The minidisk on which the root filesystem resides should be a minidisk of the same type as desired for a subsequent deploy (for example, an ECKD disk image should be captured for a subsequent deploy to an ECKD disk).
  3. The minidisks should not be a full-pack minidisk, since cylinder 0 on full-pack minidisks is reserved, and should be defined with virtual address 0100.
  4. The root disk should have a single partition.
  5. The image being captured should not have any network interface cards (NICs) defined below virtual address 1100.

In addition to the specified criteria, the following recommendations allow for efficient use of the image:

- The minidisk on which the root filesystem resides should be defined as a multiple of full gigabytes in size (for example, 1GB or 2GB). OpenStack specifies disk sizes in full gigabyte values, whereas z/VM handles disk sizes in other ways (cylinders for ECKD disks, blocks for FBA disks, and so on). See the appropriate online information if you need to convert cylinders or blocks to gigabytes; for example: http://www.mvsforums.com/helpboards/viewtopic.php?t=8316.
- During subsequent deploys of the image, the OpenStack code will ensure that a disk image is not copied to a disk smaller than the source disk, as this would result in loss of data. The disk specified in the flavor should therefore be equal to or slightly larger than the source virtual machines root disk.

OpenStack Compute supports many hypervisors, which might make it difficult for you to choose one. Most installations use only one hypervisor. However, you can use *ComputeFilter* and *ImagePropertiesFilter* to schedule different hypervisors within the same installation. The following links help you choose a hypervisor. See *Feature Support Matrix* for a detailed list of features and support across the hypervisors.

The following hypervisors are supported:

- KVM - Kernel-based Virtual Machine. The virtual disk formats that it supports is inherited from QEMU since it uses a modified QEMU program to launch the virtual machine. The supported formats include raw images, the qcow2, and VMware formats.
- LXC - Linux Containers (through libvirt), used to run Linux-based virtual machines.
- QEMU - Quick EMUlator, generally only used for development purposes.
- VMware vSphere 5.1.0 and newer - Runs VMware-based Linux and Windows images through a connection with a vCenter server.
- Xen (using libvirt) - Xen Project Hypervisor using libvirt as management interface into `nova-compute` to run Linux, Windows, FreeBSD and NetBSD virtual machines.
- XenServer - XenServer, Xen Cloud Platform (XCP) and other XAPI based Xen variants runs Linux or Windows virtual machines. You must install the `nova-compute` service in a para-virtualized VM.
- Hyper-V - Server virtualization with Microsoft Hyper-V, use to run Windows, Linux, and FreeBSD virtual machines. Runs `nova-compute` natively on the Windows virtualization platform.
- Virtuozzo 7.0.0 and newer - OS Containers and Kernel-based Virtual Machines supported via libvirt virt_type=parallels. The supported formats include ploop and qcow2 images.
- PowerVM - Server virtualization with IBM PowerVM for AIX, IBM i, and Linux workloads on the Power Systems platform.
- zVM - Server virtualization on z Systems and IBM LinuxONE, it can run Linux, z/OS and more.
- UML - User-Mode Linux is a safe, secure way of running Linux versions and Linux processes.

- Ironic - OpenStack project which provisions bare metal (as opposed to virtual) machines.

Compute schedulers

Compute uses the `nova-scheduler` service to determine how to dispatch compute requests. For example, the `nova-scheduler` service determines on which host a VM should launch. In the context of filters, the term `host` means a physical node that has a `nova-compute` service running on it. You can configure the scheduler through a variety of options.

Compute is configured with the following default scheduler options in the `/etc/nova/nova.conf` file:

```
[scheduler]
driver = filter_scheduler

[filter_scheduler]
available_filters = nova.scheduler.filters.all_filters
enabled_filters = AvailabilityZoneFilter, ComputeFilter, ComputeCapabilitiesFilter,␣
↪ImagePropertiesFilter, ServerGroupAntiAffinityFilter, ServerGroupAffinityFilter
```

By default, the scheduler `driver` is configured as a filter scheduler, as described in the next section. In the default configuration, this scheduler considers hosts that meet all the following criteria:

- Are in the requested availability zone (`AvailabilityZoneFilter`).
- Can service the request (`ComputeFilter`).
- Satisfy the extra specs associated with the instance type (`ComputeCapabilitiesFilter`).
- Satisfy any architecture, hypervisor type, or virtual machine mode properties specified on the instances image properties (`ImagePropertiesFilter`).
- Are on a different host than other instances of a group (if requested) (`ServerGroupAntiAffinityFilter`).
- Are in a set of group hosts (if requested) (`ServerGroupAffinityFilter`).

The scheduler chooses a new host when an instance is migrated.

When evacuating instances from a host, the scheduler service honors the target host defined by the administrator on the **`nova evacuate`** command. If a target is not defined by the administrator, the scheduler determines the target host. For information about instance evacuation, see *Evacuate instances*.

Prefiltering

As of the Rocky release, the scheduling process includes a prefilter step to increase the efficiency of subsequent stages. These prefilters are largely optional, and serve to augment the request that is sent to placement to reduce the set of candidate compute hosts based on attributes that placement is able to answer for us ahead of time. In addition to the prefilters listed here, also see *Tenant Isolation with Placement* and *Availability Zones with Placement*.

Compute Image Type Support

Starting in the Train release, there is a prefilter available for excluding compute nodes that do not support the `disk_format` of the image used in a boot request. This behavior is enabled by setting *`[scheduler]/query_placement_for_image_type_support=True`*. For example, the libvirt driver, when using ceph as an ephemeral backend, does not support `qcow2` images (without an expensive conversion step). In this case (and especially if you have a mix of ceph and non-ceph backed computes), enabling this feature will ensure that the scheduler does not send requests to boot a `qcow2` image to computes backed by ceph.

Compute Disabled Status Support

Starting in the Train release, there is a mandatory pre-filter which will exclude disabled compute nodes similar to (but does not fully replace) the *ComputeFilter*. Compute node resource providers with the `COMPUTE_STATUS_DISABLED` trait will be excluded as scheduling candidates. The trait is managed by the `nova-compute` service and should mirror the `disabled` status on the related compute service record in the os-services API. For example, if a compute services status is `disabled`, the related compute node resource provider(s) for that service should have the `COMPUTE_STATUS_DISABLED` trait. When the service status is `enabled` the `COMPUTE_STATUS_DISABLED` trait shall be removed.

If the compute service is down when the status is changed, the trait will be synchronized by the compute service when it is restarted. Similarly, if an error occurs when trying to add or remove the trait on a given resource provider, the trait will be synchronized when the `update_available_resource` periodic task runs - which is controlled by the *`update_resources_interval`* configuration option.

Isolate Aggregates

Starting in the Train release, there is an optional placement pre-request filter *Filtering hosts by isolating aggregates* When enabled, the traits required in the servers flavor and image must be at least those required in an aggregates metadata in order for the server to be eligible to boot on hosts in that aggregate.

Filter scheduler

The filter scheduler (`nova.scheduler.filter_scheduler.FilterScheduler`) is the default scheduler for scheduling virtual machine instances. It supports filtering and weighting to make informed decisions on where a new instance should be created.

When the filter scheduler receives a request for a resource, it first applies filters to determine which hosts are eligible for consideration when dispatching a resource. Filters are binary: either a host is accepted by the filter, or it is rejected. Hosts that are accepted by the filter are then processed by a different algorithm to decide which hosts to use for that request, described in the *Weights* section.

Filtering

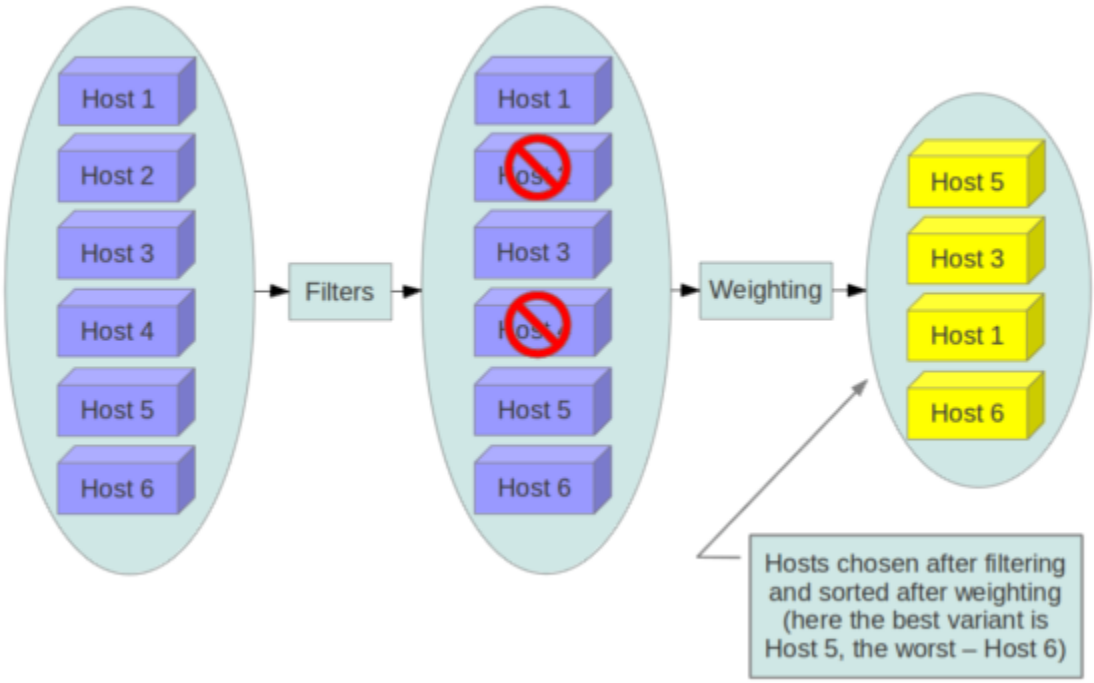
The `available_filters` configuration option in `nova.conf` provides the Compute service with the list of the filters that are available for use by the scheduler. The default setting specifies all of the filters that are included with the Compute service:

```
[filter_scheduler]
available_filters = nova.scheduler.filters.all_filters
```

This configuration option can be specified multiple times. For example, if you implemented your own custom filter in Python called `myfilter.MyFilter` and you wanted to use both the built-in filters and your custom filter, your `nova.conf` file would contain:

```
[filter_scheduler]
available_filters = nova.scheduler.filters.all_filters
available_filters = myfilter.MyFilter
```

The *`filter_scheduler.enabled_filters`* configuration option in `nova.conf` defines the list of filters that are applied by the `nova-scheduler` service.



Compute filters

The following sections describe the available compute filters.

AggregateCoreFilter

Deprecated since version 20.0.0: `AggregateCoreFilter` is deprecated since the 20.0.0 Train release. As of the introduction of the placement service in Ocata, the behavior of this filter *has changed* and no longer should be used. In the 18.0.0 Rocky release nova automatically mirrors host aggregates to placement aggregates. In the 19.0.0 Stein release initial allocation ratios support was added which allows management of the allocation ratios via the placement API in addition to the existing capability to manage allocation ratios via the nova config. See *Allocation ratios* for details.

Filters host by CPU core numbers with a per-aggregate `cpu_allocation_ratio` value. If the per-aggregate value is not found, the value falls back to the global setting. If the host is in more than one aggregate and more than one value is found, the minimum value will be used. For information about how to use this filter, see *Host aggregates and availability zones*.

Note the `cpu_allocation_ratio` *bug 1804125* restriction.

AggregateDiskFilter

Deprecated since version 20.0.0: `AggregateDiskFilter` is deprecated since the 20.0.0 Train release. As of the introduction of the placement service in Ocata, the behavior of this filter *has changed* and no longer should be used. In the 18.0.0 Rocky release nova automatically mirrors host aggregates to placement aggregates. In the 19.0.0 Stein release initial allocation ratios support was added which allows management of the allocation ratios via the placement

API in addition to the existing capability to manage allocation ratios via the nova config. See *Allocation ratios* for details.

Filters host by disk allocation with a per-aggregate `disk_allocation_ratio` value. If the per-aggregate value is not found, the value falls back to the global setting. If the host is in more than one aggregate and more than one value is found, the minimum value will be used. For information about how to use this filter, see *Host aggregates and availability zones*.

Note the `disk_allocation_ratio` *bug 1804125* restriction.

AggregateImagePropertiesIsolation

Matches properties defined in an images metadata against those of aggregates to determine host matches:

- If a host belongs to an aggregate and the aggregate defines one or more metadata that matches an images properties, that host is a candidate to boot the images instance.
- If a host does not belong to any aggregate, it can boot instances from all images.

For example, the following aggregate `myWinAgg` has the Windows operating system as metadata (named windows):

```
$ openstack aggregate show myWinAgg
+-------------------+----------------------------+
| Field             | Value                      |
+-------------------+----------------------------+
| availability_zone | zone1                      |
| created_at        | 2017-01-01T15:36:44.000000 |
| deleted           | False                      |
| deleted_at        | None                       |
| hosts             | [u'sf-devel']              |
| id                | 1                          |
| name              | myWinAgg                   |
| properties        | os_distro='windows'        |
| updated_at        | None                       |
+-------------------+----------------------------+
```

In this example, because the following Win-2012 image has the `windows` property, it boots on the `sf-devel` host (all other filters being equal):

```
$ openstack image show Win-2012
+------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | ee1eca47dc88f4879d8a229cc70a07c6                     |
| container_format | bare                                                 |
| created_at       | 2016-12-13T09:30:30Z                                 |
| disk_format      | qcow2                                                |
| ...              |                                                      |
| name             | Win-2012                                             |
| ...              |                                                      |
| properties       | os_distro='windows'                                  |
| ...              |                                                      |
```

You can configure the `AggregateImagePropertiesIsolation` filter by using the following options in the `nova.conf` file:

```
# Considers only keys matching the given namespace (string).
# Multiple values can be given, as a comma-separated list.
```

(continues on next page)

(continued from previous page)

```
aggregate_image_properties_isolation_namespace = <None>

# Separator used between the namespace and keys (string).
aggregate_image_properties_isolation_separator = .
```

Note: This filter has limitations as described in bug 1677217 which are addressed in placement *Filtering hosts by isolating aggregates* request filter.

AggregateInstanceExtraSpecsFilter

Matches properties defined in extra specs for an instance type against admin-defined properties on a host aggregate. Works with specifications that are scoped with `aggregate_instance_extra_specs`. Multiple values can be given, as a comma-separated list. For backward compatibility, also works with non-scoped specifications; this action is highly discouraged because it conflicts with *ComputeCapabilitiesFilter* filter when you enable both filters. For information about how to use this filter, see the *Host aggregates and availability zones* section.

AggregateIoOpsFilter

Filters host by disk allocation with a per-aggregate `max_io_ops_per_host` value. If the per-aggregate value is not found, the value falls back to the global setting. If the host is in more than one aggregate and more than one value is found, the minimum value will be used. For information about how to use this filter, see *Host aggregates and availability zones*. See also *IoOpsFilter*.

AggregateMultiTenancyIsolation

Ensures hosts in tenant-isolated *Host aggregates and availability zones* will only be available to a specified set of tenants. If a host is in an aggregate that has the `filter_tenant_id` metadata key, the host can build instances from only that tenant or comma-separated list of tenants. A host can be in different aggregates. If a host does not belong to an aggregate with the metadata key, the host can build instances from all tenants. This does not restrict the tenant from creating servers on hosts outside the tenant-isolated aggregate.

For example, consider there are two available hosts for scheduling, HostA and HostB. HostB is in an aggregate isolated to tenant X. A server create request from tenant X will result in either HostA *or* HostB as candidates during scheduling. A server create request from another tenant Y will result in only HostA being a scheduling candidate since HostA is not part of the tenant-isolated aggregate.

Note: There is a known limitation with the number of tenants that can be isolated per aggregate using this filter. This limitation does not exist, however, for the *Tenant Isolation with Placement* filtering capability added in the 18.0.0 Rocky release.

AggregateNumInstancesFilter

Filters host by number of instances with a per-aggregate `max_instances_per_host` value. If the per-aggregate value is not found, the value falls back to the global setting. If the host is in more than one aggregate and thus more than one value is found, the minimum value will be used. For information about how to use this filter, see *Host aggregates and availability zones*. See also *NumInstancesFilter*.

AggregateRamFilter

Deprecated since version 20.0.0: `AggregateRamFilter` is deprecated since the 20.0.0 Train release. As of the introduction of the placement service in Ocata, the behavior of this filter *has changed* and no longer should be used. In the 18.0.0 Rocky release nova automatically mirrors host aggregates to placement aggregates. In the 19.0.0 Stein release initial allocation ratios support was added which allows management of the allocation ratios via the placement API in addition to the existing capability to manage allocation ratios via the nova config. See *Allocation ratios* for details.

Filters host by RAM allocation of instances with a per-aggregate `ram_allocation_ratio` value. If the per-aggregate value is not found, the value falls back to the global setting. If the host is in more than one aggregate and thus more than one value is found, the minimum value will be used. For information about how to use this filter, see *Host aggregates and availability zones*.

Note the `ram_allocation_ratio` *bug 1804125* restriction.

AggregateTypeAffinityFilter

This filter passes hosts if no `instance_type` key is set or the `instance_type` aggregate metadata value contains the name of the `instance_type` requested. The value of the `instance_type` metadata entry is a string that may contain either a single `instance_type` name or a comma-separated list of `instance_type` names, such as `m1.nano` or `m1.nano,m1.small`. For information about how to use this filter, see *Host aggregates and availability zones*.

AllHostsFilter

This is a no-op filter. It does not eliminate any of the available hosts.

AvailabilityZoneFilter

Filters hosts by availability zone. You must enable this filter for the scheduler to respect availability zones in requests.

ComputeCapabilitiesFilter

Matches properties defined in extra specs for an instance type against compute capabilities. If an extra specs key contains a colon (`:`), anything before the colon is treated as a namespace and anything after the colon is treated as the key to be matched. If a namespace is present and is not `capabilities`, the filter ignores the namespace. For backward compatibility, also treats the extra specs key as the key to be matched if no namespace is present; this action is highly discouraged because it conflicts with *AggregateInstanceExtraSpecsFilter* filter when you enable both filters.

Some virt drivers support reporting CPU traits to the Placement service. With that feature available, you should consider using traits in flavors instead of ComputeCapabilitiesFilter, because traits provide consistent naming for CPU features in some virt drivers and querying traits is efficient. For more detail, please see Support Matrix, *Required traits*, *Forbidden traits* and Report CPU features to the Placement service.

Also refer to *Compute capabilities as traits*.

ComputeFilter

Passes all hosts that are operational and enabled.

In general, you should always enable this filter.

DifferentHostFilter

Schedules the instance on a different host from a set of instances. To take advantage of this filter, the requester must pass a scheduler hint, using `different_host` as the key and a list of instance UUIDs as the value. This filter is the opposite of the `SameHostFilter`. Using the **`openstack server create`** command, use the `--hint` flag. For example:

```
$ openstack server create --image cedef40a-ed67-4d10-800e-17455edce175 \
  --flavor 1 --hint different_host=a0cf03a5-d921-4877-bb5c-86d26cf818e1 \
  --hint different_host=8c19174f-4220-44f0-824a-cd1eeef10287 server-1
```

With the API, use the `os:scheduler_hints` key. For example:

```
{
    "server": {
        "name": "server-1",
        "imageRef": "cedef40a-ed67-4d10-800e-17455edce175",
        "flavorRef": "1"
    },
    "os:scheduler_hints": {
        "different_host": [
            "a0cf03a5-d921-4877-bb5c-86d26cf818e1",
            "8c19174f-4220-44f0-824a-cd1eeef10287"
        ]
    }
}
```

ImagePropertiesFilter

Filters hosts based on properties defined on the instances image. It passes hosts that can support the specified image properties contained in the instance. Properties include the architecture, hypervisor type, hypervisor version (for Xen hypervisor type only), and virtual machine mode.

For example, an instance might require a host that runs an ARM-based processor, and QEMU as the hypervisor. You can decorate an image with these properties by using:

```
$ openstack image set --architecture arm --property hypervisor_type=qemu \
  img-uuid
```

The image properties that the filter checks for are:

`architecture` describes the machine architecture required by the image. Examples are `i686`, `x86_64`, `arm`, and `ppc64`.

`hypervisor_type` describes the hypervisor required by the image. Examples are `xen`, `qemu`, and `xenapi`.

> **Note:** `qemu` is used for both QEMU and KVM hypervisor types.

`hypervisor_version_requires` describes the hypervisor version required by the image. The property is supported for Xen hypervisor type only. It can be used to enable support for multiple hypervisor versions, and to prevent instances with newer Xen tools from being provisioned on an older version of a hypervisor. If available, the property value is compared to the hypervisor version of the compute host.

To filter the hosts by the hypervisor version, add the `hypervisor_version_requires` property on the image as metadata and pass an operator and a required hypervisor version as its value:

```
$ openstack image set --property hypervisor_type=xen --property \
  hypervisor_version_requires=">=4.3" img-uuid
```

vm_mode describes the hypervisor application binary interface (ABI) required by the image. Examples are `xen` for Xen 3.0 paravirtual ABI, `hvm` for native ABI, `uml` for User Mode Linux paravirtual ABI, `exe` for container virt executable ABI.

IsolatedHostsFilter

Allows the admin to define a special (isolated) set of images and a special (isolated) set of hosts, such that the isolated images can only run on the isolated hosts, and the isolated hosts can only run isolated images. The flag `restrict_isolated_hosts_to_isolated_images` can be used to force isolated hosts to only run isolated images.

The logic within the filter depends on the `restrict_isolated_hosts_to_isolated_images` config option, which defaults to True. When True, a volume-backed instance will not be put on an isolated host. When False, a volume-backed instance can go on any host, isolated or not.

The admin must specify the isolated set of images and hosts in the `nova.conf` file using the `isolated_hosts` and `isolated_images` configuration options. For example:

```
[filter_scheduler]
isolated_hosts = server1, server2
isolated_images = 342b492c-128f-4a42-8d3a-c5088cf27d13, ebd267a6-ca86-4d6c-9a0e-
↪bd132d6b7d09
```

IoOpsFilter

The IoOpsFilter filters hosts by concurrent I/O operations on it. Hosts with too many concurrent I/O operations will be filtered out. The `max_io_ops_per_host` option specifies the maximum number of I/O intensive instances allowed to run on a host. A host will be ignored by the scheduler if more than `max_io_ops_per_host` instances in build, resize, snapshot, migrate, rescue or unshelve task states are running on it.

JsonFilter

Warning: This filter is not enabled by default and not comprehensively tested, and thus could fail to work as expected in non-obvious ways. Furthermore, the filter variables are based on attributes of the HostState class which could change from release to release so usage of this filter is generally not recommended. Consider using other filters such as the *ImagePropertiesFilter* or *traits-based scheduling*.

The JsonFilter allows a user to construct a custom filter by passing a scheduler hint in JSON format. The following operators are supported:

- =
- <
- >
- in

- <=
- >=
- not
- or
- and

The filter supports any attribute in the HostState class such as the following variables:

- `$free_ram_mb`
- `$free_disk_mb`
- `$hypervisor_hostname`
- `$total_usable_ram_mb`
- `$vcpus_total`
- `$vcpus_used`

Using the **`openstack server create`** command, use the `--hint` flag:

```
$ openstack server create --image 827d564a-e636-4fc4-a376-d36f7ebe1747 \
  --flavor 1 --hint query='[">=","$free_ram_mb",1024]' server1
```

With the API, use the `os:scheduler_hints` key:

```
{
    "server": {
        "name": "server-1",
        "imageRef": "cedef40a-ed67-4d10-800e-17455edce175",
        "flavorRef": "1"
    },
    "os:scheduler_hints": {
        "query": "[\">=\",\"$free_ram_mb\",1024]"
    }
}
```

MetricsFilter

Filters hosts based on meters `weight_setting`. Only hosts with the available meters are passed so that the metrics weigher will not fail due to these hosts.

NUMATopologyFilter

Filters hosts based on the NUMA topology that was specified for the instance through the use of flavor `extra_specs` in combination with the image properties, as described in detail in the related nova-spec document. Filter will try to match the exact NUMA cells of the instance to those of the host. It will consider the standard over-subscription limits for each host NUMA cell, and provide limits to the compute host accordingly.

Note: If instance has no topology defined, it will be considered for any host. If instance has a topology defined, it will be considered only for NUMA capable hosts.

NumInstancesFilter

Hosts that have more instances running than specified by the `max_instances_per_host` option are filtered out when this filter is in place.

PciPassthroughFilter

The filter schedules instances on a host if the host has devices that meet the device requests in the `extra_specs` attribute for the flavor.

RetryFilter

Deprecated since version 20.0.0: Since the 17.0.0 (Queens) release, the scheduler has provided alternate hosts for rescheduling so the scheduler does not need to be called during a reschedule which makes the `RetryFilter` useless. See the Return Alternate Hosts spec for details.

Filters out hosts that have already been attempted for scheduling purposes. If the scheduler selects a host to respond to a service request, and the host fails to respond to the request, this filter prevents the scheduler from retrying that host for the service request.

This filter is only useful if the *`scheduler.max_attempts`* configuration option is set to a value greater than one.

SameHostFilter

Schedules the instance on the same host as another instance in a set of instances. To take advantage of this filter, the requester must pass a scheduler hint, using `same_host` as the key and a list of instance UUIDs as the value. This filter is the opposite of the `DifferentHostFilter`. Using the **`openstack server create`** command, use the `--hint` flag:

```
$ openstack server create --image cedef40a-ed67-4d10-800e-17455edce175 \
  --flavor 1 --hint same_host=a0cf03a5-d921-4877-bb5c-86d26cf818e1 \
  --hint same_host=8c19174f-4220-44f0-824a-cd1eeef10287 server-1
```

With the API, use the `os:scheduler_hints` key:

```
{
    "server": {
        "name": "server-1",
        "imageRef": "cedef40a-ed67-4d10-800e-17455edce175",
        "flavorRef": "1"
    },
    "os:scheduler_hints": {
        "same_host": [
            "a0cf03a5-d921-4877-bb5c-86d26cf818e1",
            "8c19174f-4220-44f0-824a-cd1eeef10287"
        ]
    }
}
```

ServerGroupAffinityFilter

The ServerGroupAffinityFilter ensures that an instance is scheduled on to a host from a set of group hosts. To take advantage of this filter, the requester must create a server group with an `affinity` policy, and pass a scheduler hint, using `group` as the key and the server group UUID as the value. Using the **`openstack server create`** command, use the `--hint` flag. For example:

```
$ openstack server group create --policy affinity group-1
$ openstack server create --image IMAGE_ID --flavor 1 \
  --hint group=SERVER_GROUP_UUID server-1
```

ServerGroupAntiAffinityFilter

The ServerGroupAntiAffinityFilter ensures that each instance in a group is on a different host. To take advantage of this filter, the requester must create a server group with an `anti-affinity` policy, and pass a scheduler hint, using `group` as the key and the server group UUID as the value. Using the **`openstack server create`** command, use the `--hint` flag. For example:

```
$ openstack server group create --policy anti-affinity group-1
$ openstack server create --image IMAGE_ID --flavor 1 \
  --hint group=SERVER_GROUP_UUID server-1
```

SimpleCIDRAffinityFilter

Schedules the instance based on host IP subnet range. To take advantage of this filter, the requester must specify a range of valid IP address in CIDR format, by passing two scheduler hints:

`build_near_host_ip` The first IP address in the subnet (for example, `192.168.1.1`)

`cidr` The CIDR that corresponds to the subnet (for example, `/24`)

Using the **`openstack server create`** command, use the `--hint` flag. For example, to specify the IP subnet `192.168.1.1/24`:

```
$ openstack server create --image cedef40a-ed67-4d10-800e-17455edce175 \
  --flavor 1 --hint build_near_host_ip=192.168.1.1 --hint cidr=/24 server-1
```

With the API, use the `os:scheduler_hints` key:

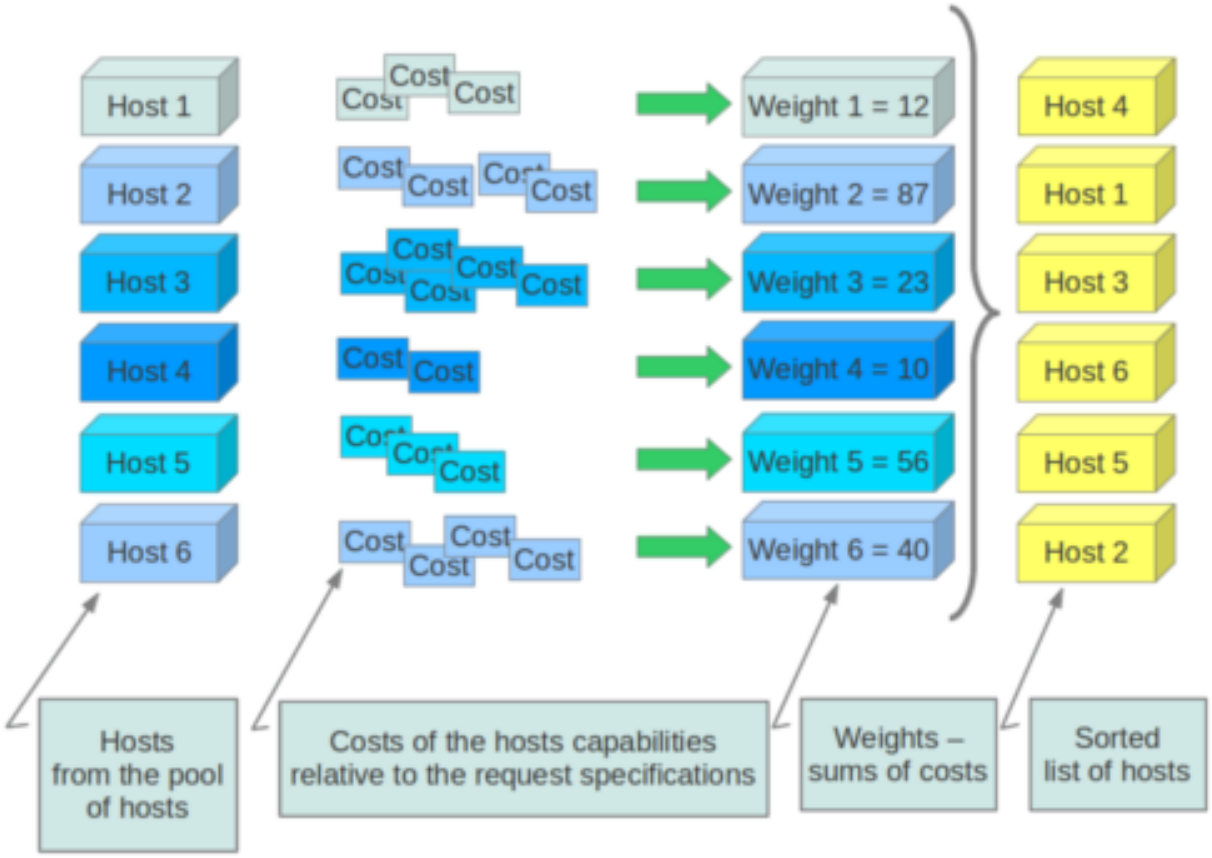
```
{
    "server": {
        "name": "server-1",
        "imageRef": "cedef40a-ed67-4d10-800e-17455edce175",
        "flavorRef": "1"
    },
    "os:scheduler_hints": {
        "build_near_host_ip": "192.168.1.1",
        "cidr": "24"
    }
}
```

Weights

When resourcing instances, the filter scheduler filters and weights each host in the list of acceptable hosts. Each time the scheduler selects a host, it virtually consumes resources on it, and subsequent selections are adjusted accordingly. This process is useful when the customer asks for the same large amount of instances, because weight is computed for each requested instance.

All weights are normalized before being summed up; the host with the largest weight is given the highest priority.

Weighting hosts



Hosts are weighted based on the following options in the `/etc/nova/nova.conf` file:

Table 5: Host weighting options

Section	Option	Description
[DEFAULT]	ram_weight_multiplier	By default, the scheduler spreads instances across all hosts evenly. Set the ram_weight_multiplier option to a negative number if you prefer stacking instead of spreading. Use a floating-point value. If the per aggregate ram_weight_multiplier metadata is set, this multiplier will override the configuration option value.
[DEFAULT]	disk_weight_multiplier	By default, the scheduler spreads instances across all hosts evenly. Set the disk_weight_multiplier option to a negative number if you prefer stacking instead of spreading. Use a floating-point value. If the per aggregate disk_weight_multiplier metadata is set, this multiplier will override the configuration option value.
[DEFAULT]	cpu_weight_multiplier	By default, the scheduler spreads instances across all hosts evenly. Set the cpu_weight_multiplier option to a negative number if you prefer stacking instead of spreading. Use a floating-point value. If the per aggregate cpu_weight_multiplier metadata is set, this multiplier will override the configuration option value.
[DEFAULT]	scheduler_host_subset_size	New instances are scheduled on a host that is chosen randomly from a subset of the N best hosts. This property defines the subset size from which a host is chosen. A value of 1 chooses the first host returned by the weighting functions. This value must be at least 1. A value less than 1 is ignored, and 1 is used instead. Use an integer value.
[DEFAULT]	scheduler_weight_classes	Defaults to nova.scheduler.weights.all_weighers. Hosts are then weighted and sorted with the largest weight winning.
[DEFAULT]	io_ops_weight_multiplier	Multiplier used for weighing host I/O operations. A negative value means a preference to choose light workload compute hosts. If the per aggregate io_ops_weight_multiplier metadata is set, this multiplier will override the configuration option value.
[filter_scheduler]	soft_affinity_weight_multiplier	Multiplier used for weighing hosts for group soft-affinity. Only a positive value is allowed.
[filter_scheduler]	soft_anti_affinity_weight_multiplier	Multiplier used for weighing hosts for group soft-anti-affinity. Only a positive value is allowed. If the per aggregate soft_anti_affinity_weight_multiplier metadata is set, this multiplier will override the configuration option value.
[filter_scheduler]	build_failure_weight_multiplier	Multiplier used for weighing hosts which have recent build failures. A positive value increases the significance of build failures reported by the host recently, making them less likely to be chosen. If the per aggregate build_failure_weight_multiplier metadata is set, this multiplier will override the configuration option value.
[metrics]	weight_multiplier	Multiplier for weighting meters. Use a floating-point value. If the per aggregate metrics_weight_multiplier metadata is set, this multiplier will override the configuration option value.
[metrics]	weight_setting	Determines how meters are weighted. Use a comma-separated list of metricName=ratio. For example: name1=1.0, name2=-1.0 results in: name1.value * 1.0 + name2.value * -1.0
[metrics]	required	Specifies how to treat unavailable meters: • True - Raises an exception. To avoid the raised exception, you

For example:

```
[DEFAULT]
scheduler_host_subset_size = 1
scheduler_weight_classes = nova.scheduler.weights.all_weighers
ram_weight_multiplier = 1.0
io_ops_weight_multiplier = 2.0
soft_affinity_weight_multiplier = 1.0
soft_anti_affinity_weight_multiplier = 1.0
[metrics]
weight_multiplier = 1.0
weight_setting = name1=1.0, name2=-1.0
required = false
weight_of_unavailable = -10000.0
```

Utilization aware scheduling

It is possible to schedule VMs using advanced scheduling decisions. These decisions are made based on enhanced usage statistics encompassing data like memory cache utilization, memory bandwidth utilization, or network bandwidth utilization. This is disabled by default. The administrator can configure how the metrics are weighted in the configuration file by using the `weight_setting` configuration option in the `nova.conf` configuration file. For example to configure metric1 with ratio1 and metric2 with ratio2:

```
weight_setting = "metric1=ratio1, metric2=ratio2"
```

Host aggregates and availability zones

Host aggregates are a mechanism for partitioning hosts in an OpenStack cloud, or a region of an OpenStack cloud, based on arbitrary characteristics. Examples where an administrator may want to do this include where a group of hosts have additional hardware or performance characteristics.

Host aggregates are not explicitly exposed to users. Instead administrators map flavors to host aggregates. Administrators do this by setting metadata on a host aggregate, and matching flavor extra specifications. The scheduler then endeavors to match user requests for instance of the given flavor to a host aggregate with the same key-value pair in its metadata. Compute nodes can be in more than one host aggregate. Weight multipliers can be controlled on a per-aggregate basis by setting the desired `xxx_weight_multiplier` aggregate metadata. Administrators are able to optionally expose a host aggregate as an availability zone. Availability zones are different from host aggregates in that they are explicitly exposed to the user, and hosts can only be in a single availability zone. Administrators can configure a default availability zone where instances will be scheduled when the user fails to specify one.

Command-line interface

The **nova** command-line client supports the following aggregate-related commands.

nova aggregate-list Print a list of all aggregates.

nova aggregate-create <name> [<availability-zone>] Create a new aggregate named `<name>`, and optionally in availability zone `[<availability-zone>]` if specified. The command returns the ID of the newly created aggregate. Hosts can be made available to multiple host aggregates. Be careful when adding a host to an additional host aggregate when the host is also in an availability zone. Pay attention when using the **nova aggregate-set-metadata** and **nova aggregate-update** commands to avoid user confusion when they boot instances in different availability zones. An error occurs if you cannot add a particular host to an aggregate zone for which it is not intended.

nova aggregate-delete <aggregate> Delete an aggregate with its <id> or <name>.

nova aggregate-show <aggregate> Show details of the aggregate with its <id> or <name>.

nova aggregate-add-host <aggregate> <host> Add host with name <host> to aggregate with its <id> or <name>.

nova aggregate-remove-host <aggregate> <host> Remove the host with name <host> from the aggregate with its <id> or <name>.

nova aggregate-set-metadata <aggregate> <key=value> [<key=value>] Add or update metadata (key-value pairs) associated with the aggregate with its <id> or <name>.

nova aggregate-update [–name <name>] [–availability-zone <availability-zone>] <aggregate> Update the name and/or availability zone for the aggregate.

nova host-list List all hosts by service. It has been deprecated since microversion 2.43. Use **nova hypervisor-list** instead.

nova hypervisor-list [–matching <hostname>] [–marker <marker>] [–limit <limit>] List hypervisors.

nova host-update [–status <enable|disable>] [–maintenance <enable|disable>] <hostname> Put/resume host into/from maintenance. It has been deprecated since microversion 2.43. To enable or disable a service, use **nova service-enable** or **nova service-disable** instead.

nova service-enable <id> Enable the service.

nova service-disable [–reason <reason>] <id> Disable the service.

Note: Only administrators can access these commands. If you try to use these commands and the user name and tenant that you use to access the Compute service do not have the admin role or the appropriate privileges, these errors occur:

```
ERROR: Policy doesn't allow compute_extension:aggregates to be performed. (HTTP 403)␣
↪(Request-ID: req-299fbff6-6729-4cef-93b2-e7e1f96b4864)
```

```
ERROR: Policy doesn't allow compute_extension:hosts to be performed. (HTTP 403)␣
↪(Request-ID: req-ef2400f6-6776-4ea3-b6f1-7704085c27d1)
```

Configure scheduler to support host aggregates

One common use case for host aggregates is when you want to support scheduling instances to a subset of compute hosts because they have a specific capability. For example, you may want to allow users to request compute hosts that have SSD drives if they need access to faster disk I/O, or access to compute hosts that have GPU cards to take advantage of GPU-accelerated code.

To configure the scheduler to support host aggregates, the *filter_scheduler.enabled_filters* configuration option must contain the AggregateInstanceExtraSpecsFilter in addition to the other filters used by the scheduler. Add the following line to /etc/nova/nova.conf on the host that runs the nova-scheduler service to enable host aggregates filtering, as well as the other filters that are typically enabled:

```
[filter_scheduler]
enabled_filters=...,AggregateInstanceExtraSpecsFilter
```

Example: Specify compute hosts with SSDs

This example configures the Compute service to enable users to request nodes that have solid-state drives (SSDs). You create a `fast-io` host aggregate in the `nova` availability zone and you add the `ssd=true` key-value pair to the aggregate. Then, you add the `node1`, and `node2` compute nodes to it.

```
$ openstack aggregate create --zone nova fast-io
+-------------------+----------------------------+
| Field             | Value                      |
+-------------------+----------------------------+
| availability_zone | nova                       |
| created_at        | 2016-12-22T07:31:13.013466 |
| deleted           | False                      |
| deleted_at        | None                       |
| id                | 1                          |
| name              | fast-io                    |
| updated_at        | None                       |
+-------------------+----------------------------+

$ openstack aggregate set --property ssd=true 1
+-------------------+----------------------------+
| Field             | Value                      |
+-------------------+----------------------------+
| availability_zone | nova                       |
| created_at        | 2016-12-22T07:31:13.000000 |
| deleted           | False                      |
| deleted_at        | None                       |
| hosts             | []                         |
| id                | 1                          |
| name              | fast-io                    |
| properties        | ssd='true'                 |
| updated_at        | None                       |
+-------------------+----------------------------+

$ openstack aggregate add host 1 node1
+-------------------+-------------------------------------------------------+
| Field             | Value                                                 |
+-------------------+-------------------------------------------------------+
| availability_zone | nova                                                  |
| created_at        | 2016-12-22T07:31:13.000000                            |
| deleted           | False                                                 |
| deleted_at        | None                                                  |
| hosts             | [u'node1']                                            |
| id                | 1                                                     |
| metadata          | {u'ssd': u'true', u'availability_zone': u'nova'}      |
| name              | fast-io                                               |
| updated_at        | None                                                  |
+-------------------+-------------------------------------------------------+

$ openstack aggregate add host 1 node2
+-------------------+-------------------------------------------------------+
| Field             | Value                                                 |
+-------------------+-------------------------------------------------------+
| availability_zone | nova                                                  |
| created_at        | 2016-12-22T07:31:13.000000                            |
| deleted           | False                                                 |
| deleted_at        | None                                                  |
```

(continues on next page)

(continued from previous page)

```
| hosts              | [u'node1', u'node2']                                  |
| id                 | 1                                                     |
| metadata           | {u'ssd': u'true', u'availability_zone': u'nova'}      |
| name               | fast-io                                               |
| updated_at         | None                                                  |
+--------------------+-------------------------------------------------------+
```

Use the **`openstack flavor create`** command to create the `ssd.large` flavor called with an ID of 6, 8 GB of RAM, 80 GB root disk, and 4 vCPUs.

```
$ openstack flavor create --id 6 --ram 8192 --disk 80 --vcpus 4 ssd.large
+----------------------------+-----------+
| Field                      | Value     |
+----------------------------+-----------+
| OS-FLV-DISABLED:disabled   | False     |
| OS-FLV-EXT-DATA:ephemeral  | 0         |
| disk                       | 80        |
| id                         | 6         |
| name                       | ssd.large |
| os-flavor-access:is_public | True      |
| ram                        | 8192      |
| rxtx_factor                | 1.0       |
| swap                       |           |
| vcpus                      | 4         |
+----------------------------+-----------+
```

Once the flavor is created, specify one or more key-value pairs that match the key-value pairs on the host aggregates with scope `aggregate_instance_extra_specs`. In this case, that is the `aggregate_instance_extra_specs:ssd=true` key-value pair. Setting a key-value pair on a flavor is done using the **`openstack flavor set`** command.

```
$ openstack flavor set --property aggregate_instance_extra_specs:ssd=true ssd.large
```

Once it is set, you should see the `extra_specs` property of the `ssd.large` flavor populated with a key of `ssd` and a corresponding value of `true`.

```
$ openstack flavor show ssd.large
+----------------------------+-------------------------------------------+
| Field                      | Value                                     |
+----------------------------+-------------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                     |
| OS-FLV-EXT-DATA:ephemeral  | 0                                         |
| disk                       | 80                                        |
| id                         | 6                                         |
| name                       | ssd.large                                 |
| os-flavor-access:is_public | True                                      |
| properties                 | aggregate_instance_extra_specs:ssd='true' |
| ram                        | 8192                                      |
| rxtx_factor                | 1.0                                       |
| swap                       |                                           |
| vcpus                      | 4                                         |
+----------------------------+-------------------------------------------+
```

Now, when a user requests an instance with the `ssd.large` flavor, the scheduler only considers hosts with the `ssd=true` key-value pair. In this example, these are `node1` and `node2`.

Aggregates in Placement

Aggregates also exist in placement and are not the same thing as host aggregates in nova. These aggregates are defined (purely) as groupings of related resource providers. Since compute nodes in nova are represented in placement as resource providers, they can be added to a placement aggregate as well. For example, get the uuid of the compute node using **`openstack hypervisor list`** and add it to an aggregate in placement using **`openstack resource provider aggregate set`**.

```
$ openstack --os-compute-api-version=2.53 hypervisor list
+--------------------------------------+---------------------+-----------------+------
↪-----------+-------+
| ID                                   | Hypervisor Hostname | Hypervisor Type | Host␣
↪IP          | State |
+--------------------------------------+---------------------+-----------------+------
↪-----------+-------+
| 815a5634-86fb-4e1e-8824-8a631fee3e06 | node1               | QEMU            | 192.
↪168.1.123   | up    |
+--------------------------------------+---------------------+-----------------+------
↪-----------+-------+

$ openstack --os-placement-api-version=1.2 resource provider aggregate set --
↪aggregate df4c74f3-d2c4-4991-b461-f1a678e1d161 815a5634-86fb-4e1e-8824-8a631fee3e06
```

Some scheduling filter operations can be performed by placement for increased speed and efficiency.

Note: The nova-api service attempts (as of nova 18.0.0) to automatically mirror the association of a compute host with an aggregate when an administrator adds or removes a host to/from a nova host aggregate. This should alleviate the need to manually create those association records in the placement API using the `openstack resource provider aggregate set` CLI invocation.

Tenant Isolation with Placement

In order to use placement to isolate tenants, there must be placement aggregates that match the membership and UUID of nova host aggregates that you want to use for isolation. The same key pattern in aggregate metadata used by the *AggregateMultiTenancyIsolation* filter controls this function, and is enabled by setting *[scheduler]/limit_tenants_to_placement_aggregate=True*.

```
$ openstack --os-compute-api-version=2.53 aggregate create myagg
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| availability_zone | None                                 |
| created_at        | 2018-03-29T16:22:23.175884           |
| deleted           | False                                |
| deleted_at        | None                                 |
| id                | 4                                    |
| name              | myagg                                |
| updated_at        | None                                 |
| uuid              | 019e2189-31b3-49e1-aff2-b220ebd91c24 |
+-------------------+--------------------------------------+

$ openstack --os-compute-api-version=2.53 aggregate add host myagg node1
+-------------------+--------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| Field             | Value                                |
+-------------------+--------------------------------------+
| availability_zone | None                                 |
| created_at        | 2018-03-29T16:22:23.175884           |
| deleted           | False                                |
| deleted_at        | None                                 |
| hosts             | [u'node1']                           |
| id                | 4                                    |
| name              | myagg                                |
| updated_at        | None                                 |
| uuid              | 019e2189-31b3-49e1-aff2-b220ebd91c24 |
+-------------------+--------------------------------------+

$ openstack project list -f value | grep 'demo'
9691591f913949818a514f95286a6b90 demo

$ openstack aggregate set --property filter_tenant_
↪id=9691591f913949818a514f95286a6b90 myagg

$ openstack --os-placement-api-version=1.2 resource provider aggregate set --
↪aggregate 019e2189-31b3-49e1-aff2-b220ebd91c24 815a5634-86fb-4e1e-8824-8a631fee3e06
```

Note that the `filter_tenant_id` metadata key can be optionally suffixed with any string for multiple tenants, such as `filter_tenant_id3=$tenantid`.

Availability Zones with Placement

In order to use placement to honor availability zone requests, there must be placement aggregates that match the membership and UUID of nova host aggregates that you assign as availability zones. The same key in aggregate metadata used by the *AvailabilityZoneFilter* filter controls this function, and is enabled by setting *[scheduler]/query_placement_for_availability_zone=True*.

```
$ openstack --os-compute-api-version=2.53 aggregate create myaz
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| availability_zone | None                                 |
| created_at        | 2018-03-29T16:22:23.175884           |
| deleted           | False                                |
| deleted_at        | None                                 |
| id                | 4                                    |
| name              | myaz                                 |
| updated_at        | None                                 |
| uuid              | 019e2189-31b3-49e1-aff2-b220ebd91c24 |
+-------------------+--------------------------------------+

$ openstack --os-compute-api-version=2.53 aggregate add host myaz node1
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| availability_zone | None                                 |
| created_at        | 2018-03-29T16:22:23.175884           |
| deleted           | False                                |
| deleted_at        | None                                 |
| hosts             | [u'node1']                           |
```

(continues on next page)

(continued from previous page)

```
| id                | 4                                    |
| name              | myagg                                |
| updated_at        | None                                 |
| uuid              | 019e2189-31b3-49e1-aff2-b220ebd91c24 |
+-------------------+--------------------------------------+

$ openstack aggregate set --property availability_zone=az002 myaz

$ openstack --os-placement-api-version=1.2 resource provider aggregate set --
↪aggregate 019e2189-31b3-49e1-aff2-b220ebd91c24 815a5634-86fb-4e1e-8824-8a631fee3e06
```

With the above configuration, the *AvailabilityZoneFilter* filter can be disabled in *[filter_scheduler]/enabled_filters* while retaining proper behavior (and doing so with the higher performance of placements implementation).

XenServer hypervisor pools to support live migration

When using the XenAPI-based hypervisor, the Compute service uses host aggregates to manage XenServer Resource pools, which are used in supporting live migration.

Allocation ratios

The following configuration options exist to control allocation ratios per compute node to support over-commit of resources:

- *cpu_allocation_ratio*: allows overriding the VCPU inventory allocation ratio for a compute node
- *ram_allocation_ratio*: allows overriding the MEMORY_MB inventory allocation ratio for a compute node
- *disk_allocation_ratio*: allows overriding the DISK_GB inventory allocation ratio for a compute node

Prior to the 19.0.0 Stein release, if left unset, the `cpu_allocation_ratio` defaults to 16.0, the `ram_allocation_ratio` defaults to 1.5, and the `disk_allocation_ratio` defaults to 1.0.

Starting with the 19.0.0 Stein release, the following configuration options control the initial allocation ratio values for a compute node:

- *initial_cpu_allocation_ratio*: the initial VCPU inventory allocation ratio for a new compute node record, defaults to 16.0
- *initial_ram_allocation_ratio*: the initial MEMORY_MB inventory allocation ratio for a new compute node record, defaults to 1.5
- *initial_disk_allocation_ratio*: the initial DISK_GB inventory allocation ratio for a new compute node record, defaults to 1.0

Scheduling considerations

The allocation ratio configuration is used both during reporting of compute node resource provider inventory to the placement service and during scheduling.

Note: Regarding the *AggregateCoreFilter*, *AggregateDiskFilter* and *AggregateRamFilter*, starting in 15.0.0 (Ocata) there is a behavior change where aggregate-based overcommit ratios will no longer be honored during scheduling for

the FilterScheduler. Instead, overcommit values must be set on a per-compute-node basis in the Nova configuration files.

If you have been relying on per-aggregate overcommit, during your upgrade, you must change to using per-compute-node overcommit ratios in order for your scheduling behavior to stay consistent. Otherwise, you may notice increased NoValidHost scheduling failures as the aggregate-based overcommit is no longer being considered.

See bug 1804125 for more details.

Usage scenarios

Since allocation ratios can be set via nova configuration, host aggregate metadata and the placement API, it can be confusing to know which should be used. This really depends on your scenario. A few common scenarios are detailed here.

1. When the deployer wants to **always** set an override value for a resource on a compute node, the deployer would ensure that the `[DEFAULT]/cpu_allocation_ratio`, `[DEFAULT]/ram_allocation_ratio` and `[DEFAULT]/disk_allocation_ratio` configuration options are set to a non-None value (or greater than 0.0 before the 19.0.0 Stein release). This will make the `nova-compute` service overwrite any externally-set allocation ratio values set via the placement REST API.

2. When the deployer wants to set an **initial** value for a compute node allocation ratio but wants to allow an admin to adjust this afterwards without making any configuration file changes, the deployer would set the `[DEFAULT]/initial_cpu_allocation_ratio`, `[DEFAULT]/initial_ram_allocation_ratio` and `[DEFAULT]/initial_disk_allocation_ratio` configuration options and then manage the allocation ratios using the placement REST API (or osc-placement command line interface). For example:

   ```
   $ openstack resource provider inventory set --resource VCPU:allocation_ratio=1.0␣
   ↪815a5634-86fb-4e1e-8824-8a631fee3e06
   ```

   Note the *bug 1804125* restriction.

3. When the deployer wants to **always** use the placement API to set allocation ratios, then the deployer should ensure that `[DEFAULT]/xxx_allocation_ratio` options are all set to None (the default since 19.0.0 Stein, 0.0 before Stein) and then manage the allocation ratios using the placement REST API (or osc-placement command line interface).

   This scenario is the workaround for bug 1804125.

Hypervisor-specific considerations

Nova provides three configuration options, *reserved_host_cpus*, *reserved_host_memory_mb*, and *reserved_host_disk_mb*, that can be used to set aside some number of resources that will not be consumed by an instance, whether these resources are overcommitted or not. Some virt drivers may benefit from the use of these options to account for hypervisor-specific overhead.

HyperV Hyper-V creates a VM memory file on the local disk when an instance starts. The size of this file corresponds to the amount of RAM allocated to the instance.

You should configure the *reserved_host_disk_mb* config option to account for this overhead, based on the amount of memory available to instances.

XenAPI XenServer memory overhead is proportional to the size of the VM and larger flavor VMs become more efficient with respect to overhead. This overhead can be calculated using the following formula:

```
overhead (MB) = (instance.memory * 0.00781) + (instance.vcpus * 1.5) + 3
```

You should configure the *reserved_host_memory_mb* config option to account for this overhead, based on the size of your hosts and instances. For more information, refer to https://wiki.openstack.org/wiki/XenServer/Overhead.

Cells considerations

By default cells are enabled for scheduling new instances but they can be disabled (new schedulings to the cell are blocked). This may be useful for users while performing cell maintenance, failures or other interventions. It is to be noted that creating pre-disabled cells and enabling/disabling existing cells should either be followed by a restart or SIGHUP of the nova-scheduler service for the changes to take effect.

Command-line interface

The **nova-manage** command-line client supports the cell-disable related commands. To enable or disable a cell, use **nova-manage cell_v2 update_cell** and to create pre-disabled cells, use **nova-manage cell_v2 create_cell**. See the *Nova Cells v2* man page for details on command usage.

Compute capabilities as traits

Starting with the 19.0.0 Stein release, the `nova-compute` service will report certain `COMPUTE_*` traits based on its compute driver capabilities to the placement service. The traits will be associated with the resource provider for that compute service. These traits can be used during scheduling by configuring flavors with *Required traits* or *Forbidden traits*. For example, if you have a host aggregate with a set of compute nodes that support multi-attach volumes, you can restrict a flavor to that aggregate by adding the `trait:COMPUTE_VOLUME_MULTI_ATTACH=required` extra spec to the flavor and then restrict the flavor to the aggregate *as normal*.

Here is an example of a libvirt compute node resource provider that is exposing some CPU features as traits, driver capabilities as traits, and a custom trait denoted by the `CUSTOM_` prefix:

```
$ openstack --os-placement-api-version 1.6 resource provider trait list \
> d9b3dbc4-50e2-42dd-be98-522f6edaab3f --sort-column name
+---------------------------------------+
| name                                  |
+---------------------------------------+
| COMPUTE_DEVICE_TAGGING                |
| COMPUTE_NET_ATTACH_INTERFACE          |
| COMPUTE_NET_ATTACH_INTERFACE_WITH_TAG |
| COMPUTE_TRUSTED_CERTS                 |
| COMPUTE_VOLUME_ATTACH_WITH_TAG        |
| COMPUTE_VOLUME_EXTEND                 |
| COMPUTE_VOLUME_MULTI_ATTACH           |
| CUSTOM_IMAGE_TYPE_RBD                 |
| HW_CPU_X86_MMX                        |
| HW_CPU_X86_SSE                        |
| HW_CPU_X86_SSE2                       |
| HW_CPU_X86_SVM                        |
+---------------------------------------+
```

Rules

There are some rules associated with capability-defined traits.

1. The compute service owns these traits and will add/remove them when the `nova-compute` service starts and when the `update_available_resource` periodic task runs, with run intervals controlled by config option *`update_resources_interval`*.
2. The compute service will not remove any custom traits set on the resource provider externally, such as the `CUSTOM_IMAGE_TYPE_RBD` trait in the example above.
3. If compute-owned traits are removed from the resource provider externally, for example by running `openstack resource provider trait delete <rp_uuid>`, the compute service will add its traits again on restart or SIGHUP.
4. If a compute trait is set on the resource provider externally which is not supported by the driver, for example by adding the `COMPUTE_VOLUME_EXTEND` trait when the driver does not support that capability, the compute service will automatically remove the unsupported trait on restart or SIGHUP.
5. Compute capability traits are standard traits defined in the os-traits library.

Further information on capabilities and traits can be found in the *Technical Reference Deep Dives section*.

Compute log files

The corresponding log file of each Compute service is stored in the `/var/log/nova/` directory of the host on which each service runs.

Table 6: Log files used by Compute services

Log file	Service name (CentOS/Fedora/openSUSE/Red Hat Enterprise Linux/SUSE Linux Enterprise)	Service name (Ubuntu/Debian)
`nova-api.log`	`openstack-nova-api`	`nova-api`
`nova-compute.log`	`openstack-nova-compute`	`nova-compute`
`nova-conductor.log`	`openstack-nova-conductor`	`nova-conductor`
`nova-network.log`[1]	`openstack-nova-network`	`nova-network`
`nova-manage.log`	`nova-manage`	`nova-manage`
`nova-scheduler.log`	`openstack-nova-scheduler`	`nova-scheduler`

Compute service sample configuration files

Files in this section can be found in `/etc/nova`.

api-paste.ini

The Compute service stores its API configuration settings in the `api-paste.ini` file.

```
############
# Metadata #
############
[composite:metadata]
use = egg:Paste#urlmap
/: meta
```

(continues on next page)

[1] The `nova` network service (`openstack-nova-network`/ `nova-network`) only runs in deployments that are not configured to use the Networking service (`neutron`).

(continued from previous page)

```
[pipeline:meta]
pipeline = cors metaapp

[app:metaapp]
paste.app_factory = nova.api.metadata.handler:MetadataRequestHandler.factory

##############
# OpenStack #
##############

[composite:osapi_compute]
use = call:nova.api.openstack.urlmap:urlmap_factory
/: oscomputeversions
# v21 is an exactly feature match for v2, except it has more stringent
# input validation on the wsgi surface (prevents fuzzing early on the
# API). It also provides new features via API microversions which are
# opt into for clients. Unaware clients will receive the same frozen
# v2 API feature set, but with some relaxed validation
/v2: openstack_compute_api_v21_legacy_v2_compatible
/v2.1: openstack_compute_api_v21

[composite:openstack_compute_api_v21]
use = call:nova.api.auth:pipeline_factory_v21
noauth2 = cors http_proxy_to_wsgi compute_req_id faultwrap request_log sizelimit␣
↪osprofiler noauth2 osapi_compute_app_v21
keystone = cors http_proxy_to_wsgi compute_req_id faultwrap request_log sizelimit␣
↪osprofiler authtoken keystonecontext osapi_compute_app_v21

[composite:openstack_compute_api_v21_legacy_v2_compatible]
use = call:nova.api.auth:pipeline_factory_v21
noauth2 = cors http_proxy_to_wsgi compute_req_id faultwrap request_log sizelimit␣
↪osprofiler noauth2 legacy_v2_compatible osapi_compute_app_v21
keystone = cors http_proxy_to_wsgi compute_req_id faultwrap request_log sizelimit␣
↪osprofiler authtoken keystonecontext legacy_v2_compatible osapi_compute_app_v21

[filter:request_log]
paste.filter_factory = nova.api.openstack.requestlog:RequestLog.factory

[filter:compute_req_id]
paste.filter_factory = nova.api.compute_req_id:ComputeReqIdMiddleware.factory

[filter:faultwrap]
paste.filter_factory = nova.api.openstack:FaultWrapper.factory

[filter:noauth2]
paste.filter_factory = nova.api.openstack.auth:NoAuthMiddleware.factory

[filter:osprofiler]
paste.filter_factory = nova.profiler:WsgiMiddleware.factory

[filter:sizelimit]
paste.filter_factory = oslo_middleware:RequestBodySizeLimiter.factory

[filter:http_proxy_to_wsgi]
paste.filter_factory = oslo_middleware.http_proxy_to_wsgi:HTTPProxyToWSGI.factory
```

(continues on next page)

(continued from previous page)

```
[filter:legacy_v2_compatible]
paste.filter_factory = nova.api.openstack:LegacyV2CompatibleWrapper.factory

[app:osapi_compute_app_v21]
paste.app_factory = nova.api.openstack.compute:APIRouterV21.factory

[pipeline:oscomputeversions]
pipeline = cors faultwrap request_log http_proxy_to_wsgi oscomputeversionapp

[app:oscomputeversionapp]
paste.app_factory = nova.api.openstack.compute.versions:Versions.factory

##########
# Shared #
##########

[filter:cors]
paste.filter_factory = oslo_middleware.cors:filter_factory
oslo_config_project = nova

[filter:keystonecontext]
paste.filter_factory = nova.api.auth:NovaKeystoneContext.factory

[filter:authtoken]
paste.filter_factory = keystonemiddleware.auth_token:filter_factory
```

policy.yaml

The `policy.yaml` file defines additional access controls that apply to the Compute service.

```
# Decides what is required for the 'is_admin:True' check to succeed.
#"context_is_admin": "role:admin"

# Default rule for most non-Admin APIs.
#"admin_or_owner": "is_admin:True or project_id:%(project_id)s"

# Default rule for most Admin APIs.
#"admin_api": "is_admin:True"

# Reset the state of a given server
# POST  /servers/{server_id}/action (os-resetState)
#"os_compute_api:os-admin-actions:reset_state": "rule:admin_api"

# Inject network information into the server
# POST  /servers/{server_id}/action (injectNetworkInfo)
#"os_compute_api:os-admin-actions:inject_network_info": "rule:admin_api"

# Reset networking on a server
# POST  /servers/{server_id}/action (resetNetwork)
#"os_compute_api:os-admin-actions:reset_network": "rule:admin_api"

# Change the administrative password for a server
# POST  /servers/{server_id}/action (changePassword)
#"os_compute_api:os-admin-password": "rule:admin_or_owner"
```

(continues on next page)

(continued from previous page)

```
# Create, list, update, and delete guest agent builds
#
# This is XenAPI driver specific. It is used to force the upgrade of
# the XenAPI guest agent on instance boot.
# GET  /os-agents
# POST  /os-agents
# PUT  /os-agents/{agent_build_id}
# DELETE  /os-agents/{agent_build_id}
#"os_compute_api:os-agents": "rule:admin_api"

# Create or replace metadata for an aggregate
# POST  /os-aggregates/{aggregate_id}/action (set_metadata)
#"os_compute_api:os-aggregates:set_metadata": "rule:admin_api"

# Add a host to an aggregate
# POST  /os-aggregates/{aggregate_id}/action (add_host)
#"os_compute_api:os-aggregates:add_host": "rule:admin_api"

# Create an aggregate
# POST  /os-aggregates
#"os_compute_api:os-aggregates:create": "rule:admin_api"

# Remove a host from an aggregate
# POST  /os-aggregates/{aggregate_id}/action (remove_host)
#"os_compute_api:os-aggregates:remove_host": "rule:admin_api"

# Update name and/or availability zone for an aggregate
# PUT  /os-aggregates/{aggregate_id}
#"os_compute_api:os-aggregates:update": "rule:admin_api"

# List all aggregates
# GET  /os-aggregates
#"os_compute_api:os-aggregates:index": "rule:admin_api"

# Delete an aggregate
# DELETE  /os-aggregates/{aggregate_id}
#"os_compute_api:os-aggregates:delete": "rule:admin_api"

# Show details for an aggregate
# GET  /os-aggregates/{aggregate_id}
#"os_compute_api:os-aggregates:show": "rule:admin_api"

# Create an assisted volume snapshot
# POST  /os-assisted-volume-snapshots
#"os_compute_api:os-assisted-volume-snapshots:create": "rule:admin_api"

# Delete an assisted volume snapshot
# DELETE  /os-assisted-volume-snapshots/{snapshot_id}
#"os_compute_api:os-assisted-volume-snapshots:delete": "rule:admin_api"

# List port interfaces or show details of a port interface attached to
# a server
# GET  /servers/{server_id}/os-interface
# GET  /servers/{server_id}/os-interface/{port_id}
#"os_compute_api:os-attach-interfaces": "rule:admin_or_owner"

# Attach an interface to a server
```

(continues on next page)

(continued from previous page)

```
# POST  /servers/{server_id}/os-interface
#"os_compute_api:os-attach-interfaces:create": "rule:admin_or_owner"

# Detach an interface from a server
# DELETE  /servers/{server_id}/os-interface/{port_id}
#"os_compute_api:os-attach-interfaces:delete": "rule:admin_or_owner"

# List availability zone information without host information
# GET  /os-availability-zone
#"os_compute_api:os-availability-zone:list": "rule:admin_or_owner"

# List detailed availability zone information with host information
# GET  /os-availability-zone/detail
#"os_compute_api:os-availability-zone:detail": "rule:admin_api"

# List and show details of bare metal nodes.
#
# These APIs are proxy calls to the Ironic service and are deprecated.
# GET  /os-baremetal-nodes
# GET  /os-baremetal-nodes/{node_id}
#"os_compute_api:os-baremetal-nodes": "rule:admin_api"

# Show console connection information for a given console
# authentication token
# GET  /os-console-auth-tokens/{console_token}
#"os_compute_api:os-console-auth-tokens": "rule:admin_api"

# Show console output for a server
# POST  /servers/{server_id}/action (os-getConsoleOutput)
#"os_compute_api:os-console-output": "rule:admin_or_owner"

# Create a console for a server instance
# POST  /servers/{server_id}/consoles
#"os_compute_api:os-consoles:create": "rule:admin_or_owner"

# Show console details for a server instance
# GET  /servers/{server_id}/consoles/{console_id}
#"os_compute_api:os-consoles:show": "rule:admin_or_owner"

# Delete a console for a server instance
# DELETE  /servers/{server_id}/consoles/{console_id}
#"os_compute_api:os-consoles:delete": "rule:admin_or_owner"

# List all consoles for a server instance
# GET  /servers/{server_id}/consoles
#"os_compute_api:os-consoles:index": "rule:admin_or_owner"

# Create a back up of a server
# POST  /servers/{server_id}/action (createBackup)
#"os_compute_api:os-create-backup": "rule:admin_or_owner"

# Restore a soft deleted server or force delete a server before
# deferred cleanup
# POST  /servers/{server_id}/action (restore)
# POST  /servers/{server_id}/action (forceDelete)
#"os_compute_api:os-deferred-delete": "rule:admin_or_owner"
```

(continues on next page)

(continued from previous page)

```
# Evacuate a server from a failed host to a new host
# POST  /servers/{server_id}/action (evacuate)
#"os_compute_api:os-evacuate": "rule:admin_api"

# Return extended attributes for server.
#
# This rule will control the visibility for a set of servers
# attributes:
#
# - ``OS-EXT-SRV-ATTR:host`` - ``OS-EXT-SRV-ATTR:instance_name`` -
# ``OS-EXT-SRV-ATTR:reservation_id`` (since microversion 2.3) - ``OS-
# EXT-SRV-ATTR:launch_index`` (since microversion 2.3) - ``OS-EXT-SRV-
# ATTR:hostname`` (since microversion 2.3) - ``OS-EXT-SRV-
# ATTR:kernel_id`` (since microversion 2.3) - ``OS-EXT-SRV-
# ATTR:ramdisk_id`` (since microversion 2.3) - ``OS-EXT-SRV-
# ATTR:root_device_name`` (since microversion 2.3) - ``OS-EXT-SRV-
# ATTR:user_data`` (since microversion 2.3)
# GET  /servers/{id}
# GET  /servers/detail
#"os_compute_api:os-extended-server-attributes": "rule:admin_api"

# List available extensions and show information for an extension by
# alias
# GET  /extensions
# GET  /extensions/{alias}
#"os_compute_api:extensions": "rule:admin_or_owner"

# Add flavor access to a tenant
# POST  /flavors/{flavor_id}/action (addTenantAccess)
#"os_compute_api:os-flavor-access:add_tenant_access": "rule:admin_api"

# Remove flavor access from a tenant
# POST  /flavors/{flavor_id}/action (removeTenantAccess)
#"os_compute_api:os-flavor-access:remove_tenant_access": "rule:admin_api"

# List flavor access information
#
# Allows access to the full list of tenants that have access to a
# flavor via an os-flavor-access API.
# GET  /flavors/{flavor_id}/os-flavor-access
#"os_compute_api:os-flavor-access": "rule:admin_or_owner"

# Show an extra spec for a flavor
# GET  /flavors/{flavor_id}/os-extra_specs/{flavor_extra_spec_key}
#"os_compute_api:os-flavor-extra-specs:show": "rule:admin_or_owner"

# Create extra specs for a flavor
# POST  /flavors/{flavor_id}/os-extra_specs/
#"os_compute_api:os-flavor-extra-specs:create": "rule:admin_api"

# Update an extra spec for a flavor
# PUT  /flavors/{flavor_id}/os-extra_specs/{flavor_extra_spec_key}
#"os_compute_api:os-flavor-extra-specs:update": "rule:admin_api"

# Delete an extra spec for a flavor
# DELETE  /flavors/{flavor_id}/os-extra_specs/{flavor_extra_spec_key}
#"os_compute_api:os-flavor-extra-specs:delete": "rule:admin_api"
```

(continues on next page)

(continued from previous page)

```
# List extra specs for a flavor. Starting with microversion 2.47, the
# flavor used for a server is also returned in the response when
# showing server details, updating a server or rebuilding a server.
# Starting with microversion 2.61, extra specs may be returned in
# responses for the flavor resource.
# GET  /flavors/{flavor_id}/os-extra_specs/
# GET  /servers/detail
# GET  /servers/{server_id}
# PUT  /servers/{server_id}
# POST  /servers/{server_id}/action (rebuild)
# POST  /flavors
# GET  /flavors/detail
# GET  /flavors/{flavor_id}
# PUT  /flavors/{flavor_id}
#"os_compute_api:os-flavor-extra-specs:index": "rule:admin_or_owner"

# Create a flavor
# POST  /flavors
#"os_compute_api:os-flavor-manage:create": "rule:admin_api"

# Update a flavor
# PUT  /flavors/{flavor_id}
#"os_compute_api:os-flavor-manage:update": "rule:admin_api"

# Delete a flavor
# DELETE  /flavors/{flavor_id}
#"os_compute_api:os-flavor-manage:delete": "rule:admin_api"

# List floating IP pools. This API is deprecated.
# GET  /os-floating-ip-pools
#"os_compute_api:os-floating-ip-pools": "rule:admin_or_owner"

# Manage a project's floating IPs. These APIs are all deprecated.
# POST  /servers/{server_id}/action (addFloatingIp)
# POST  /servers/{server_id}/action (removeFloatingIp)
# GET  /os-floating-ips
# POST  /os-floating-ips
# GET  /os-floating-ips/{floating_ip_id}
# DELETE  /os-floating-ips/{floating_ip_id}
#"os_compute_api:os-floating-ips": "rule:admin_or_owner"

# List, show and manage physical hosts.
#
# These APIs are all deprecated in favor of os-hypervisors and os-
# services.
# GET  /os-hosts
# GET  /os-hosts/{host_name}
# PUT  /os-hosts/{host_name}
# GET  /os-hosts/{host_name}/reboot
# GET  /os-hosts/{host_name}/shutdown
# GET  /os-hosts/{host_name}/startup
#"os_compute_api:os-hosts": "rule:admin_api"

# Policy rule for hypervisor related APIs.
#
# This rule will be checked for the following APIs:
```

(continues on next page)

(continued from previous page)

```
#
# List all hypervisors, list all hypervisors with details, show
# summary statistics for all hypervisors over all compute nodes, show
# details for a hypervisor, show the uptime of a hypervisor, search
# hypervisor by hypervisor_hostname pattern and list all servers on
# hypervisors that can match the provided hypervisor_hostname pattern.
# GET  /os-hypervisors
# GET  /os-hypervisors/details
# GET  /os-hypervisors/statistics
# GET  /os-hypervisors/{hypervisor_id}
# GET  /os-hypervisors/{hypervisor_id}/uptime
# GET  /os-hypervisors/{hypervisor_hostname_pattern}/search
# GET  /os-hypervisors/{hypervisor_hostname_pattern}/servers
#"os_compute_api:os-hypervisors": "rule:admin_api"

# Add events details in action details for a server.
#
# This check is performed only after the check os_compute_api:os-
# instance-actions passes. Beginning with Microversion 2.51, events
# details are always included; traceback information is provided per
# event if policy enforcement passes. Beginning with Microversion
# 2.62, each event includes a hashed host identifier and, if policy
# enforcement passes, the name of the host.
# GET  /servers/{server_id}/os-instance-actions/{request_id}
#"os_compute_api:os-instance-actions:events": "rule:admin_api"

# List actions and show action details for a server.
# GET  /servers/{server_id}/os-instance-actions
# GET  /servers/{server_id}/os-instance-actions/{request_id}
#"os_compute_api:os-instance-actions": "rule:admin_or_owner"

# List all usage audits and that occurred before a specified time for
# all servers on all compute hosts where usage auditing is configured
# GET  /os-instance_usage_audit_log
# GET  /os-instance_usage_audit_log/{before_timestamp}
#"os_compute_api:os-instance-usage-audit-log": "rule:admin_api"

# Show IP addresses details for a network label of a server
# GET  /servers/{server_id}/ips/{network_label}
#"os_compute_api:ips:show": "rule:admin_or_owner"

# List IP addresses that are assigned to a server
# GET  /servers/{server_id}/ips
#"os_compute_api:ips:index": "rule:admin_or_owner"

# List all keypairs
# GET  /os-keypairs
#"os_compute_api:os-keypairs:index": "rule:admin_api or user_id:%(user_id)s"

# Create a keypair
# POST  /os-keypairs
#"os_compute_api:os-keypairs:create": "rule:admin_api or user_id:%(user_id)s"

# Delete a keypair
# DELETE  /os-keypairs/{keypair_name}
#"os_compute_api:os-keypairs:delete": "rule:admin_api or user_id:%(user_id)s"
```

(continues on next page)

(continued from previous page)

```
# Show details of a keypair
# GET  /os-keypairs/{keypair_name}
#"os_compute_api:os-keypairs:show": "rule:admin_api or user_id:%(user_id)s"

# Show rate and absolute limits for the project
# GET  /limits
#"os_compute_api:limits": "rule:admin_or_owner"

# Lock a server
# POST  /servers/{server_id}/action (lock)
#"os_compute_api:os-lock-server:lock": "rule:admin_or_owner"

# Unlock a server
# POST  /servers/{server_id}/action (unlock)
#"os_compute_api:os-lock-server:unlock": "rule:admin_or_owner"

# Unlock a server, regardless who locked the server.
#
# This check is performed only after the check os_compute_api:os-lock-
# server:unlock passes
# POST  /servers/{server_id}/action (unlock)
#"os_compute_api:os-lock-server:unlock:unlock_override": "rule:admin_api"

# Cold migrate a server to a host
# POST  /servers/{server_id}/action (migrate)
#"os_compute_api:os-migrate-server:migrate": "rule:admin_api"

# Live migrate a server to a new host without a reboot
# POST  /servers/{server_id}/action (os-migrateLive)
#"os_compute_api:os-migrate-server:migrate_live": "rule:admin_api"

# List migrations
# GET  /os-migrations
#"os_compute_api:os-migrations:index": "rule:admin_api"

# Add or remove a fixed IP address from a server.
#
# These APIs are proxy calls to the Network service. These are all
# deprecated.
# POST  /servers/{server_id}/action (addFixedIp)
# POST  /servers/{server_id}/action (removeFixedIp)
#"os_compute_api:os-multinic": "rule:admin_or_owner"

# Create and delete a network, add and disassociate a network from a
# project.
#
# These APIs are only available with nova-network which is deprecated.
# POST  /os-networks
# POST  /os-networks/add
# DELETE  /os-networks/{network_id}
# POST  /os-networks/{network_id}/action (disassociate)
#"os_compute_api:os-networks": "rule:admin_api"

# List networks for the project and show details for a network.
#
# These APIs are proxy calls to the Network service. These are all
# deprecated.
```

(continues on next page)

(continued from previous page)

```
# GET  /os-networks
# GET  /os-networks/{network_id}
#"os_compute_api:os-networks:view": "rule:admin_or_owner"

# Associate or disassociate a network from a host or project.
#
# These APIs are only available with nova-network which is deprecated.
# POST  /os-networks/{network_id}/action (disassociate_host)
# POST  /os-networks/{network_id}/action (disassociate_project)
# POST  /os-networks/{network_id}/action (associate_host)
#"os_compute_api:os-networks-associate": "rule:admin_api"

# Pause a server
# POST  /servers/{server_id}/action (pause)
#"os_compute_api:os-pause-server:pause": "rule:admin_or_owner"

# Unpause a paused server
# POST  /servers/{server_id}/action (unpause)
#"os_compute_api:os-pause-server:unpause": "rule:admin_or_owner"

# List quotas for specific quota classs
# GET  /os-quota-class-sets/{quota_class}
#"os_compute_api:os-quota-class-sets:show": "is_admin:True or quota_class:%(quota_
↪class)s"

# Update quotas for specific quota class
# PUT  /os-quota-class-sets/{quota_class}
#"os_compute_api:os-quota-class-sets:update": "rule:admin_api"

# Update the quotas
# PUT  /os-quota-sets/{tenant_id}
#"os_compute_api:os-quota-sets:update": "rule:admin_api"

# List default quotas
# GET  /os-quota-sets/{tenant_id}/defaults
#"os_compute_api:os-quota-sets:defaults": "@"

# Show a quota
# GET  /os-quota-sets/{tenant_id}
#"os_compute_api:os-quota-sets:show": "rule:admin_or_owner"

# Revert quotas to defaults
# DELETE  /os-quota-sets/{tenant_id}
#"os_compute_api:os-quota-sets:delete": "rule:admin_api"

# Show the detail of quota
# GET  /os-quota-sets/{tenant_id}/detail
#"os_compute_api:os-quota-sets:detail": "rule:admin_or_owner"

# Generate a URL to access remove server console
# POST  /servers/{server_id}/action (os-getRDPConsole)
# POST  /servers/{server_id}/action (os-getSerialConsole)
# POST  /servers/{server_id}/action (os-getSPICEConsole)
# POST  /servers/{server_id}/action (os-getVNCConsole)
# POST  /servers/{server_id}/remote-consoles
#"os_compute_api:os-remote-consoles": "rule:admin_or_owner"
```

(continues on next page)

(continued from previous page)

```
# Rescue/unrescue a server
# POST  /servers/{server_id}/action (rescue)
# POST  /servers/{server_id}/action (unrescue)
#"os_compute_api:os-rescue": "rule:admin_or_owner"

# List, show information for, create, or delete default security group
# rules.
#
# These APIs are only available with nova-network which is now
# deprecated.
# GET  /os-security-group-default-rules
# GET  /os-security-group-default-rules/{security_group_default_rule_id}
# POST  /os-security-group-default-rules
# DELETE  /os-security-group-default-rules/{security_group_default_rule_id}
#"os_compute_api:os-security-group-default-rules": "rule:admin_api"

# List, show, add, or remove security groups.
#
# APIs which are directly related to security groups resource are
# deprecated: Lists, shows information for, creates, updates and
# deletes security groups. Creates and deletes security group rules.
# All these APIs are deprecated.
#
# APIs which are related to server resource are not deprecated: Lists
# Security Groups for a server. Add Security Group to a server and
# remove security group from a server.
# GET  /os-security-groups
# GET  /os-security-groups/{security_group_id}
# POST  /os-security-groups
# PUT  /os-security-groups/{security_group_id}
# DELETE  /os-security-groups/{security_group_id}
# GET  /servers/{server_id}/os-security-groups
# POST  /servers/{server_id}/action (addSecurityGroup)
# POST  /servers/{server_id}/action (removeSecurityGroup)
#"os_compute_api:os-security-groups": "rule:admin_or_owner"

# Show the usage data for a server
# GET  /servers/{server_id}/diagnostics
#"os_compute_api:os-server-diagnostics": "rule:admin_api"

# Create one or more external events
# POST  /os-server-external-events
#"os_compute_api:os-server-external-events:create": "rule:admin_api"

# Create a new server group
# POST  /os-server-groups
#"os_compute_api:os-server-groups:create": "rule:admin_or_owner"

# Delete a server group
# DELETE  /os-server-groups/{server_group_id}
#"os_compute_api:os-server-groups:delete": "rule:admin_or_owner"

# List all server groups
# GET  /os-server-groups
#"os_compute_api:os-server-groups:index": "rule:admin_or_owner"

# Show details of a server group
```

(continues on next page)

(continued from previous page)

```
# GET  /os-server-groups/{server_group_id}
#"os_compute_api:os-server-groups:show": "rule:admin_or_owner"

# List all metadata of a server
# GET  /servers/{server_id}/metadata
#"os_compute_api:server-metadata:index": "rule:admin_or_owner"

# Show metadata for a server
# GET  /servers/{server_id}/metadata/{key}
#"os_compute_api:server-metadata:show": "rule:admin_or_owner"

# Create metadata for a server
# POST  /servers/{server_id}/metadata
#"os_compute_api:server-metadata:create": "rule:admin_or_owner"

# Replace metadata for a server
# PUT  /servers/{server_id}/metadata
#"os_compute_api:server-metadata:update_all": "rule:admin_or_owner"

# Update metadata from a server
# PUT  /servers/{server_id}/metadata/{key}
#"os_compute_api:server-metadata:update": "rule:admin_or_owner"

# Delete metadata from a server
# DELETE  /servers/{server_id}/metadata/{key}
#"os_compute_api:server-metadata:delete": "rule:admin_or_owner"

# Show and clear the encrypted administrative password of a server
# GET  /servers/{server_id}/os-server-password
# DELETE  /servers/{server_id}/os-server-password
#"os_compute_api:os-server-password": "rule:admin_or_owner"

# Delete all the server tags
# DELETE  /servers/{server_id}/tags
#"os_compute_api:os-server-tags:delete_all": "rule:admin_or_owner"

# List all tags for given server
# GET  /servers/{server_id}/tags
#"os_compute_api:os-server-tags:index": "rule:admin_or_owner"

# Replace all tags on specified server with the new set of tags.
# PUT  /servers/{server_id}/tags
#"os_compute_api:os-server-tags:update_all": "rule:admin_or_owner"

# Delete a single tag from the specified server
# DELETE  /servers/{server_id}/tags/{tag}
#"os_compute_api:os-server-tags:delete": "rule:admin_or_owner"

# Add a single tag to the server if server has no specified tag
# PUT  /servers/{server_id}/tags/{tag}
#"os_compute_api:os-server-tags:update": "rule:admin_or_owner"

# Check tag existence on the server.
# GET  /servers/{server_id}/tags/{tag}
#"os_compute_api:os-server-tags:show": "rule:admin_or_owner"

# Show the NUMA topology data for a server
```

(continues on next page)

(continued from previous page)

```
# GET  /servers/{server_id}/topology
#"compute:server:topology:index": "rule:admin_or_owner"

# Show the NUMA topology data for a server with host NUMA ID and CPU
# pinning information
# GET  /servers/{server_id}/topology
#"compute:server:topology:host:index": "rule:admin_api"

# List all servers
# GET  /servers
#"os_compute_api:servers:index": "rule:admin_or_owner"

# List all servers with detailed information
# GET  /servers/detail
#"os_compute_api:servers:detail": "rule:admin_or_owner"

# List all servers for all projects
# GET  /servers
#"os_compute_api:servers:index:get_all_tenants": "rule:admin_api"

# List all servers with detailed information for all projects
# GET  /servers/detail
#"os_compute_api:servers:detail:get_all_tenants": "rule:admin_api"

# Allow all filters when listing servers
# GET  /servers
# GET  /servers/detail
#"os_compute_api:servers:allow_all_filters": "rule:admin_api"

# Show a server
# GET  /servers/{server_id}
#"os_compute_api:servers:show": "rule:admin_or_owner"

# Show a server with additional host status information
# GET  /servers/{server_id}
# GET  /servers/detail
#"os_compute_api:servers:show:host_status": "rule:admin_api"

# Create a server
# POST  /servers
#"os_compute_api:servers:create": "rule:admin_or_owner"

# Create a server on the specified host and/or node.
#
# In this case, the server is forced to launch on the specified host
# and/or node by bypassing the scheduler filters unlike the
# ``compute:servers:create:requested_destination`` rule.
# POST  /servers
#"os_compute_api:servers:create:forced_host": "rule:admin_api"

# Create a server on the requested compute service host and/or
# hypervisor_hostname.
#
# In this case, the requested host and/or hypervisor_hostname is
# validated by the scheduler filters unlike the
# ``os_compute_api:servers:create:forced_host`` rule.
# POST  /servers
```

(continues on next page)

(continued from previous page)

```
#"compute:servers:create:requested_destination": "rule:admin_api"

# Create a server with the requested volume attached to it
# POST  /servers
#"os_compute_api:servers:create:attach_volume": "rule:admin_or_owner"

# Create a server with the requested network attached to it
# POST  /servers
#"os_compute_api:servers:create:attach_network": "rule:admin_or_owner"

# Create a server with trusted image certificate IDs
# POST  /servers
#"os_compute_api:servers:create:trusted_certs": "rule:admin_or_owner"

# This rule controls the compute API validation behavior of creating a
# server with a flavor that has 0 disk, indicating the server should
# be volume-backed.
#
# For a flavor with disk=0, the root disk will be set to exactly the
# size of the image used to deploy the instance. However, in this case
# the filter_scheduler cannot select the compute host based on the
# virtual image size. Therefore, 0 should only be used for volume
# booted instances or for testing purposes.
#
# WARNING: It is a potential security exposure to enable this policy
# rule if users can upload their own images since repeated attempts to
# create a disk=0 flavor instance with a large image can exhaust the
# local disk of the compute (or shared storage cluster). See bug
# https://bugs.launchpad.net/nova/+bug/1739646 for details.
# POST  /servers
#"os_compute_api:servers:create:zero_disk_flavor": "rule:admin_api"

# Attach an unshared external network to a server
# POST  /servers
# POST  /servers/{server_id}/os-interface
#"network:attach_external_network": "is_admin:True"

# Delete a server
# DELETE  /servers/{server_id}
#"os_compute_api:servers:delete": "rule:admin_or_owner"

# Update a server
# PUT  /servers/{server_id}
#"os_compute_api:servers:update": "rule:admin_or_owner"

# Confirm a server resize
# POST  /servers/{server_id}/action (confirmResize)
#"os_compute_api:servers:confirm_resize": "rule:admin_or_owner"

# Revert a server resize
# POST  /servers/{server_id}/action (revertResize)
#"os_compute_api:servers:revert_resize": "rule:admin_or_owner"

# Reboot a server
# POST  /servers/{server_id}/action (reboot)
#"os_compute_api:servers:reboot": "rule:admin_or_owner"
```

(continues on next page)

(continued from previous page)

```
# Resize a server
# POST  /servers/{server_id}/action (resize)
#"os_compute_api:servers:resize": "rule:admin_or_owner"

# Rebuild a server
# POST  /servers/{server_id}/action (rebuild)
#"os_compute_api:servers:rebuild": "rule:admin_or_owner"

# Rebuild a server with trusted image certificate IDs
# POST  /servers/{server_id}/action (rebuild)
#"os_compute_api:servers:rebuild:trusted_certs": "rule:admin_or_owner"

# Create an image from a server
# POST  /servers/{server_id}/action (createImage)
#"os_compute_api:servers:create_image": "rule:admin_or_owner"

# Create an image from a volume backed server
# POST  /servers/{server_id}/action (createImage)
#"os_compute_api:servers:create_image:allow_volume_backed": "rule:admin_or_owner"

# Start a server
# POST  /servers/{server_id}/action (os-start)
#"os_compute_api:servers:start": "rule:admin_or_owner"

# Stop a server
# POST  /servers/{server_id}/action (os-stop)
#"os_compute_api:servers:stop": "rule:admin_or_owner"

# Trigger crash dump in a server
# POST  /servers/{server_id}/action (trigger_crash_dump)
#"os_compute_api:servers:trigger_crash_dump": "rule:admin_or_owner"

# Show details for an in-progress live migration for a given server
# GET  /servers/{server_id}/migrations/{migration_id}
#"os_compute_api:servers:migrations:show": "rule:admin_api"

# Force an in-progress live migration for a given server to complete
# POST  /servers/{server_id}/migrations/{migration_id}/action (force_complete)
#"os_compute_api:servers:migrations:force_complete": "rule:admin_api"

# Delete(Abort) an in-progress live migration
# DELETE  /servers/{server_id}/migrations/{migration_id}
#"os_compute_api:servers:migrations:delete": "rule:admin_api"

# Lists in-progress live migrations for a given server
# GET  /servers/{server_id}/migrations
#"os_compute_api:servers:migrations:index": "rule:admin_api"

# List all running Compute services in a region, enables or disable
# scheduling for a Compute service, logs disabled Compute service
# information, set or unset forced_down flag for the compute service
# and delete a Compute service
# GET  /os-services
# PUT  /os-services/enable
# PUT  /os-services/disable
# PUT  /os-services/disable-log-reason
# PUT  /os-services/force-down
```

(continues on next page)

(continued from previous page)

```
# PUT  /os-services/{service_id}
# DELETE  /os-services/{service_id}
#"os_compute_api:os-services": "rule:admin_api"

# Shelve server
# POST  /servers/{server_id}/action (shelve)
#"os_compute_api:os-shelve:shelve": "rule:admin_or_owner"

# Unshelve (restore) shelved server
# POST  /servers/{server_id}/action (unshelve)
#"os_compute_api:os-shelve:unshelve": "rule:admin_or_owner"

# Shelf-offload (remove) server
# POST  /servers/{server_id}/action (shelveOffload)
#"os_compute_api:os-shelve:shelve_offload": "rule:admin_api"

# Show usage statistics for a specific tenant
# GET  /os-simple-tenant-usage/{tenant_id}
#"os_compute_api:os-simple-tenant-usage:show": "rule:admin_or_owner"

# List per tenant usage statistics for all tenants
# GET  /os-simple-tenant-usage
#"os_compute_api:os-simple-tenant-usage:list": "rule:admin_api"

# Resume suspended server
# POST  /servers/{server_id}/action (resume)
#"os_compute_api:os-suspend-server:resume": "rule:admin_or_owner"

# Suspend server
# POST  /servers/{server_id}/action (suspend)
#"os_compute_api:os-suspend-server:suspend": "rule:admin_or_owner"

# Create, list, show information for, and delete project networks.
#
# These APIs are proxy calls to the Network service. These are all
# deprecated.
# GET  /os-tenant-networks
# POST  /os-tenant-networks
# GET  /os-tenant-networks/{network_id}
# DELETE  /os-tenant-networks/{network_id}
#"os_compute_api:os-tenant-networks": "rule:admin_or_owner"

# Show rate and absolute limits for the project.
#
# This policy only checks if the user has access to the requested
# project limits. And this check is performed only after the check
# os_compute_api:limits passes
# GET  /limits
#"os_compute_api:os-used-limits": "rule:admin_api"

# Manage volumes for use with the Compute API.
#
# Lists, shows details, creates, and deletes volumes and snapshots.
# These APIs are proxy calls to the Volume service. These are all
# deprecated.
# GET  /os-volumes
# POST  /os-volumes
```

(continues on next page)

(continued from previous page)

```
# GET  /os-volumes/detail
# GET  /os-volumes/{volume_id}
# DELETE  /os-volumes/{volume_id}
# GET  /os-snapshots
# POST  /os-snapshots
# GET  /os-snapshots/detail
# GET  /os-snapshots/{snapshot_id}
# DELETE  /os-snapshots/{snapshot_id}
#"os_compute_api:os-volumes": "rule:admin_or_owner"

# List volume attachments for an instance
# GET  /servers/{server_id}/os-volume_attachments
#"os_compute_api:os-volumes-attachments:index": "rule:admin_or_owner"

# Attach a volume to an instance
# POST  /servers/{server_id}/os-volume_attachments
#"os_compute_api:os-volumes-attachments:create": "rule:admin_or_owner"

# Show details of a volume attachment
# GET  /servers/{server_id}/os-volume_attachments/{volume_id}
#"os_compute_api:os-volumes-attachments:show": "rule:admin_or_owner"

# Update a volume attachment
# PUT  /servers/{server_id}/os-volume_attachments/{volume_id}
#"os_compute_api:os-volumes-attachments:update": "rule:admin_api"

# Detach a volume from an instance
# DELETE  /servers/{server_id}/os-volume_attachments/{volume_id}
#"os_compute_api:os-volumes-attachments:delete": "rule:admin_or_owner"
```

rootwrap.conf

The `rootwrap.conf` file defines configuration values used by the rootwrap script when the Compute service needs to escalate its privileges to those of the root user.

It is also possible to disable the root wrapper, and default to sudo only. Configure the `disable_rootwrap` option in the `[workaround]` section of the `nova.conf` configuration file.

```
# Configuration for nova-rootwrap
# This file should be owned by (and only-writeable by) the root user

[DEFAULT]
# List of directories to load filter definitions from (separated by ',').
# These directories MUST all be only writeable by root !
filters_path=/etc/nova/rootwrap.d,/usr/share/nova/rootwrap

# List of directories to search executables in, in case filters do not
# explicitly specify a full path (separated by ',')
# If not specified, defaults to system PATH environment variable.
# These directories MUST all be only writeable by root !
exec_dirs=/sbin,/usr/sbin,/bin,/usr/bin,/usr/local/sbin,/usr/local/bin

# Enable logging to syslog
# Default value is False
use_syslog=False
```

(continues on next page)

(continued from previous page)

```
# Which syslog facility to use.
# Valid values include auth, authpriv, syslog, local0, local1...
# Default value is 'syslog'
syslog_log_facility=syslog

# Which messages to log.
# INFO means log all usage
# ERROR means only log unsuccessful attempts
syslog_log_level=ERROR
```

4.3.8 Configure live migrations

Migration enables an administrator to move a virtual machine instance from one compute host to another. A typical scenario is planned maintenance on the source host, but migration can also be useful to redistribute the load when many VM instances are running on a specific physical machine.

This document covers live migrations using the *KVM-libvirt* and *XenServer* hypervisors.

Note: Not all Compute service hypervisor drivers support live-migration, or support all live-migration features. Similarly not all compute service features are supported.

Consult *Feature Support Matrix* to determine which hypervisors support live-migration.

See the *Configuration Guide* for details on hypervisor configuration settings.

The migration types are:

- **Non-live migration**, also known as cold migration or simply migration.

  The instance is shut down, then moved to another hypervisor and restarted. The instance recognizes that it was rebooted, and the application running on the instance is disrupted.

  This section does not cover cold migration.

- **Live migration**

  The instance keeps running throughout the migration. This is useful when it is not possible or desirable to stop the application running on the instance.

  Live migrations can be classified further by the way they treat instance storage:

  - **Shared storage-based live migration**. The instance has ephemeral disks that are located on storage shared between the source and destination hosts.
  - **Block live migration**, or simply block migration. The instance has ephemeral disks that are not shared between the source and destination hosts. Block migration is incompatible with read-only devices such as CD-ROMs and Configuration Drive (config_drive).
  - **Volume-backed live migration**. Instances use volumes rather than ephemeral disks.

  Block live migration requires copying disks from the source to the destination host. It takes more time and puts more load on the network. Shared-storage and volume-backed live migration does not copy disks.

Note: In a multi-cell cloud, instances can be live migrated to a different host in the same cell, but not across cells.

The following sections describe how to configure your hosts for live migrations using the KVM and XenServer hypervisors.

KVM-libvirt

General configuration

To enable any type of live migration, configure the compute hosts according to the instructions below:

1. Set the following parameters in `nova.conf` on all compute hosts:

   - `server_listen=0.0.0.0`

     You must not make the VNC server listen to the IP address of its compute host, since that addresses changes when the instance is migrated.

     Important: Since this setting allows VNC clients from any IP address to connect to instance consoles, you must take additional measures like secure networks or firewalls to prevent potential attackers from gaining access to instances.

   - `instances_path` must have the same value for all compute hosts. In this guide, the value `/var/lib/nova/instances` is assumed.

2. Ensure that name resolution on all compute hosts is identical, so that they can connect each other through their hostnames.

   If you use `/etc/hosts` for name resolution and enable SELinux, ensure that `/etc/hosts` has the correct SELinux context:

   ```
   # restorecon /etc/hosts
   ```

3. Enable password-less SSH so that root on one compute host can log on to any other compute host without providing a password. The `libvirtd` daemon, which runs as root, uses the SSH protocol to copy the instance to the destination and cant know the passwords of all compute hosts.

   You may, for example, compile roots public SSH keys on all compute hosts into an `authorized_keys` file and deploy that file to the compute hosts.

4. Configure the firewalls to allow libvirt to communicate between compute hosts.

   By default, libvirt uses the TCP port range from 49152 to 49261 for copying memory and disk contents. Compute hosts must accept connections in this range.

   For information about ports used by libvirt, see the libvirt documentation.

   Important: Be mindful of the security risks introduced by opening ports.

Securing live migration streams

If your compute nodes have at least libvirt 4.4.0 and QEMU 2.11.0, it is strongly recommended to secure all your live migration streams by taking advantage of the QEMU-native TLS feature. This requires a pre-existing PKI (Public Key Infrastructure) setup. For further details on how to set this all up, refer to the *Secure live migration with QEMU-native TLS* document.

Block migration, volume-based live migration

If your environment satisfies the requirements for QEMU-native TLS, then block migration requires some setup; refer to the above section, *Securing live migration streams*, for details. Otherwise, no additional configuration is required for block migration and volume-backed live migration.

Be aware that block migration adds load to the network and storage subsystems.

Shared storage

Compute hosts have many options for sharing storage, for example NFS, shared disk array LUNs, Ceph or GlusterFS.

The next steps show how a regular Linux system might be configured as an NFS v4 server for live migration. For detailed information and alternative ways to configure NFS on Linux, see instructions for Ubuntu, RHEL and derivatives or SLES and OpenSUSE.

1. Ensure that UID and GID of the nova user are identical on the compute hosts and the NFS server.
2. Create a directory with enough disk space for all instances in the cloud, owned by user nova. In this guide, we assume `/var/lib/nova/instances`.
3. Set the execute/search bit on the `instances` directory:

   ```
   $ chmod o+x /var/lib/nova/instances
   ```

   This allows qemu to access the `instances` directory tree.
4. Export `/var/lib/nova/instances` to the compute hosts. For example, add the following line to `/etc/exports`:

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

   The asterisk permits access to any NFS client. The option `fsid=0` exports the instances directory as the NFS root.

After setting up the NFS server, mount the remote filesystem on all compute hosts.

1. Assuming the NFS servers hostname is `nfs-server`, add this line to `/etc/fstab` to mount the NFS root:

   ```
   nfs-server:/ /var/lib/nova/instances nfs4 defaults 0 0
   ```
2. Test NFS by mounting the instances directory and check access permissions for the nova user:

   ```
   $ sudo mount -a -v
   $ ls -ld /var/lib/nova/instances/
   drwxr-xr-x. 2 nova nova 6 Mar 14 21:30 /var/lib/nova/instances/
   ```

Advanced configuration for KVM and QEMU

Live migration copies the instances memory from the source to the destination compute host. After a memory page has been copied, the instance may write to it again, so that it has to be copied again. Instances that frequently write to different memory pages can overwhelm the memory copy process and prevent the live migration from completing.

This section covers configuration settings that can help live migration of memory-intensive instances succeed.

1. **Live migration completion timeout**

The Compute service aborts a migration when it has been running for too long. The timeout is calculated based on the instance size, which is the instances memory size in GiB. In the case of block migration, the size of ephemeral storage in GiB is added.

The timeout in seconds is the instance size multiplied by the configurable parameter *libvirt.live_migration_completion_timeout*, whose default is 800. For example, shared-storage live migration of an instance with 8GiB memory will time out after 6400 seconds.

2. **Instance downtime**

   Near the end of the memory copy, the instance is paused for a short time so that the remaining few pages can be copied without interference from instance memory writes. The Compute service initializes this time to a small value that depends on the instance size, typically around 50 milliseconds. When it notices that the memory copy does not make sufficient progress, it increases the time gradually.

   You can influence the instance downtime algorithm with the help of three configuration variables on the compute hosts:

   ```
   live_migration_downtime = 500
   live_migration_downtime_steps = 10
   live_migration_downtime_delay = 75
   ```

   `live_migration_downtime` sets the maximum permitted downtime for a live migration, in *milliseconds*. The default is 500.

   `live_migration_downtime_steps` sets the total number of adjustment steps until `live_migration_downtime` is reached. The default is 10 steps.

   `live_migration_downtime_delay` sets the time interval between two adjustment steps in *seconds*. The default is 75.

3. **Auto-convergence**

   One strategy for a successful live migration of a memory-intensive instance is slowing the instance down. This is called auto-convergence. Both libvirt and QEMU implement this feature by automatically throttling the instances CPU when memory copy delays are detected.

   Auto-convergence is disabled by default. You can enable it by setting `live_migration_permit_auto_converge=true`.

   Caution: Before enabling auto-convergence, make sure that the instances application tolerates a slow-down.

   Be aware that auto-convergence does not guarantee live migration success.

4. **Post-copy**

   Live migration of a memory-intensive instance is certain to succeed when you enable post-copy. This feature, implemented by libvirt and QEMU, activates the virtual machine on the destination host before all of its memory has been copied. When the virtual machine accesses a page that is missing on the destination host, the resulting page fault is resolved by copying the page from the source host.

   Post-copy is disabled by default. You can enable it by setting `live_migration_permit_post_copy=true`.

   When you enable both auto-convergence and post-copy, auto-convergence remains disabled.

   Caution: The page faults introduced by post-copy can slow the instance down.

When the network connection between source and destination host is interrupted, page faults cannot be resolved anymore and the instance is rebooted.

The full list of live migration configuration parameters is documented in the *Nova Configuration Options*

XenServer

Shared storage

Prerequisites

- **Compatible XenServer hypervisors**.

  For more information, see the Requirements for Creating Resource Pools section of the XenServer Administrators Guide.

- **Shared storage**.

  An NFS export, visible to all XenServer hosts.

  Note: For the supported NFS versions, see the NFS and SMB section of the XenServer Administrators Guide.

To use shared storage live migration with XenServer hypervisors, the hosts must be joined to a XenServer pool.

Using shared storage live migrations with XenServer Hypervisors

1. Add an NFS VHD storage to your master XenServer, and set it as the default storage repository. For more information, see NFS VHD in the XenServer Administrators Guide.
2. Configure all compute nodes to use the default storage repository (`sr`) for pool operations. Add this line to your `nova.conf` configuration files on all compute nodes:

```
sr_matching_filter=default-sr:true
```

3. To add a host to a pool, you need to know the pool master ip address, user name and password. Run below command on the XenServer host:

```
$ xe pool-join master-address=MASTER_IP master-username=root master-
↪password=MASTER_PASSWORD
```

Note: The added compute node and the host will shut down to join the host to the XenServer pool. The operation will fail if any server other than the compute node is running or suspended on the host.

Block migration

- **Compatible XenServer hypervisors**.

  The hypervisors must support the Storage XenMotion feature. See your XenServer manual to make sure your edition has this feature.

Note:

- To use block migration, you must use the `--block-migrate` parameter with the live migration command.
- Block migration works only with EXT local storage storage repositories, and the server must not have any volumes attached.

VMware

vSphere configuration

Enable vMotion on all ESX hosts which are managed by Nova by following the instructions in this KB article.

4.3.9 Compute service node firewall requirements

Console connections for virtual machines, whether direct or through a proxy, are received on ports `5900` to `5999`. The firewall on each Compute service node must allow network traffic on these ports.

This procedure modifies the iptables firewall to allow incoming connections to the Compute services.

Configuring the service-node firewall

1. Log in to the server that hosts the Compute service, as root.
2. Edit the `/etc/sysconfig/iptables` file, to add an INPUT rule that allows TCP traffic on ports from `5900` to `5999`. Make sure the new rule appears before any INPUT rules that REJECT traffic:

```
-A INPUT -p tcp -m multiport --dports 5900:5999 -j ACCEPT
```

3. Save the changes to the `/etc/sysconfig/iptables` file, and restart the `iptables` service to pick up the changes:

```
$ service iptables restart
```

4. Repeat this process for each Compute service node.

4.3.10 Evacuate instances

If a hardware malfunction or other error causes a cloud compute node to fail, you can evacuate instances to make them available again.

To preserve user data on the server disk, configure shared storage on the target host. When you evacuate the instance, Compute detects whether shared storage is available on the target host. Also, you must validate that the current VM host is not operational. Otherwise, the evacuation fails.

There are two different ways to evacuate instances from a failed compute node. The first one using the **`nova evacuate`** command can be used to evacuate a single instance from a failed node. In some cases where the node in question hosted many instances it might be easier to use **`nova host-evacuate`** to evacuate them all in one shot.

Evacuate a single instance

The procedure below explains how to evacuate a single instance from a failed compute node. Please be aware that these steps describe a post failure scenario and should not be used if the instance is still up and running.

1. To find a host for the evacuated instance, list all hosts:

```
$ openstack host list
```

2. Evacuate the instance. You can use the `--password PWD` option to pass the instance password to the command. If you do not specify a password, the command generates and prints one after it finishes successfully. The following command evacuates a server from a failed host to `HOST_B`.

```
$ nova evacuate EVACUATED_SERVER_NAME HOST_B
```

   The command rebuilds the instance from the original image or volume and returns a password. The command preserves the original configuration, which includes the instance ID, name, uid, IP address, and so on.

```
+-----------+--------------+
| Property  |    Value     |
+-----------+--------------+
| adminPass | kRAJpErnT4xZ |
+-----------+--------------+
```

   Optionally you can omit the `HOST_B` parameter and let the scheduler choose a new target host.

3. To preserve the user disk data on the evacuated server, deploy Compute with a shared file system. To configure your system, see *Configure live migrations*. The following example does not change the password.

```
$ nova evacuate EVACUATED_SERVER_NAME HOST_B --on-shared-storage
```

Note: Starting with the 2.14 compute API version, one no longer needs to specify `--on-shared-storage` even if the server is on a compute host which is using shared storage. The compute service will automatically detect if it is running on shared storage.

Evacuate all instances

The procedure below explains how to evacuate all instances from a failed compute node. Please note that this method should not be used if the host still has instances up and running.

1. To find a host for the evacuated instances, list all hosts:

```
$ openstack host list
```

2. Evacuate all instances from `FAILED_HOST` to `TARGET_HOST`:

```
$ nova host-evacuate --target_host TARGET_HOST FAILED_HOST
```

   The option `--target_host` is optional and can be omitted to let the scheduler decide where to place the instances.

   The above argument `FAILED_HOST` can also be a pattern to search for instead of an exact hypervisor hostname but it is recommended to use a fully qualified domain name to make sure no hypervisor host is getting evacuated by mistake. As long as you are not using a pattern you might want to use the `--strict` flag which got introduced in version 10.2.0 to make sure nova matches the `FAILED_HOST` exactly.

Note:

```
+------+--------+--------------+
| Name | Status | Task State   |
+------+--------+--------------+
| vm_1 | ACTIVE | powering-off |
+------------------------------+
```

If the instance task state is not None, evacuation will be possible. However, depending on the ongoing operation, there may be clean up required in other services which the instance was using, such as neutron, cinder, glance, or the storage backend.

4.3.11 Manage Flavors

Admin users can use the **`openstack flavor`** command to customize and manage flavors. To see information for this command, run:

```
$ openstack flavor --help
Command "flavor" matches:
  flavor create
  flavor delete
  flavor list
  flavor set
  flavor show
  flavor unset
```

Note: Configuration rights can be delegated to additional users by redefining the access controls for `os_compute_api:os-flavor-manage:create`, `os_compute_api:os-flavor-manage:update` and `os_compute_api:os-flavor-manage:delete` in `/etc/nova/policy.json` on the `nova-api` server.

Note: Flavor customization can be limited by the hypervisor in use. For example the libvirt driver enables quotas on CPUs available to a VM, disk tuning, bandwidth I/O, watchdog behavior, random number generator device control, and instance VIF traffic control.

For information on the flavors and flavor extra specs, refer to *Flavors*.

Create a flavor

1. List flavors to show the ID and name, the amount of memory, the amount of disk space for the root partition and for the ephemeral partition, the swap, and the number of virtual CPUs for each flavor:

   ```
   $ openstack flavor list
   ```

2. To create a flavor, specify a name, ID, RAM size, disk size, and the number of vCPUs for the flavor, as follows:

   ```
   $ openstack flavor create FLAVOR_NAME --id FLAVOR_ID \
       --ram RAM_IN_MB --disk ROOT_DISK_IN_GB --vcpus NUMBER_OF_VCPUS
   ```

Note: Unique ID (integer or UUID) for the new flavor. If specifying auto, a UUID will be automatically generated.

Here is an example that creates a public `m1.extra_tiny` flavor that automatically gets an ID assigned, with 256 MB memory, no disk space, and one VCPU.

```
$ openstack flavor create --public m1.extra_tiny --id auto \
    --ram 256 --disk 0 --vcpus 1
```

3. If an individual user or group of users needs a custom flavor that you do not want other projects to have access to, you can create a private flavor.

```
$ openstack flavor create --private m1.extra_tiny --id auto \
    --ram 256 --disk 0 --vcpus 1
```

After you create a flavor, assign it to a project by specifying the flavor name or ID and the project ID:

```
$ openstack flavor set --project PROJECT_ID m1.extra_tiny
```

For a list of optional parameters, run this command:

```
$ openstack help flavor create
```

4. In addition, you can set or unset properties, commonly referred to as extra specs, for the existing flavor. The `extra_specs` metadata keys can influence the instance directly when it is launched. If a flavor sets the `quota:vif_outbound_peak=65536` extra spec, the instances outbound peak bandwidth I/O should be less than or equal to 512 Mbps. There are several aspects that can work for an instance including *CPU limits*, *Disk tuning*, *Bandwidth I/O*, *Watchdog behavior*, and *Random-number generator*. For information about available metadata keys, see *Flavors*.

   For a list of optional parameters, run this command:

```
$ openstack flavor set --help
```

Modify a flavor

Only the description of flavors can be modified (starting from microversion 2.55). To modify the description of a flavor, specify the flavor name or ID and a new description as follows:

```
$ openstack --os-compute-api-version 2.55 flavor set --description <DESCRIPTION>
↪<FLAVOR>
```

Note: The only field that can be updated is the description field. Nova has historically intentionally not included an API to update a flavor because that would be confusing for instances already created with that flavor. Needing to change any other aspect of a flavor requires deleting and/or creating a new flavor.

Nova stores a serialized version of the flavor associated with an instance record in the `instance_extra` table. While nova supports updating flavor extra_specs it does not update the embedded flavor in existing instances. Nova does not update the embedded flavor as the extra_specs change may invalidate the current placement of the instance or alter the compute context that has been created for the instance by the virt driver. For this reason admins should avoid updating extra_specs for flavors used by existing instances. A resize can be used to update existing instances if required but as a resize performs a cold migration it is not transparent to a tenant.

Delete a flavor

To delete a flavor, specify the flavor name or ID as follows:

```
$ openstack flavor delete FLAVOR
```

Default Flavors

Previous versions of nova typically deployed with default flavors. This was removed from Newton. The following table lists the default flavors for Mitaka and earlier.

Flavor	VCPUs	Disk (in GB)	RAM (in MB)
m1.tiny	1	1	512
m1.small	1	20	2048
m1.medium	2	40	4096
m1.large	4	80	8192
m1.xlarge	8	160	16384

4.3.12 Image Caching

What is Image Caching?

In order to understand what image caching is and why it is beneficial, it helps to be familiar with the process by which an instance is booted from a given base image. When a new instance is created on a compute node, the following general steps are performed by the compute manager in conjunction with the virt driver:

1. Download the base image from glance
2. Copy or COW the base image to create a new root disk image for the instance
3. Boot the instance using the new root disk image

The first step involves downloading the entire base image to the local disk on the compute node, which could involve many gigabytes of network traffic, storage, and many minutes of latency between the start of the boot process and actually running the instance. When the virt driver supports image caching, step #1 above may be skipped if the base image is already present on the compute node. This is most often the case when another instance has been booted on that node from the same base image recently. If present, the download operation can be skipped, which greatly reduces the time-to-boot for the second and subsequent instances that use the same base image, as well as avoids load on the glance server and the network connection.

By default, the compute node will periodically scan the images it has cached, looking for base images that are not used by any instances on the node that are older than a configured lifetime (24 hours by default). Those unused images are deleted from the cache directory until they are needed again.

For more information about configuring image cache behavior, see the documentation for the following configuration options:

- *image_cache_subdirectory_name*
- *image_cache_manager_interval*
- *remove_unused_base_images*
- *remove_unused_original_minimum_age_seconds*

Note: Some ephemeral backend drivers may not use or need image caching, or may not behave in the same way as others. For example, when using the `rbd` backend with the `libvirt` driver and a shared pool with glance, images are COWd at the storage level and thus need not be downloaded (and thus cached) at the compute node at all.

Image Caching Resource Accounting

Generally the size of the image cache is not part of the data Nova includes when reporting available or consumed disk space. This means that when `nova-compute` reports 100G of total disk space, the scheduler will assume that 100G of instances may be placed there. Usually disk is the most plentiful resource and thus the last to be exhausted, so this is often not problematic. However, if many instances are booted from distinct images, all of which need to be cached in addition to the disk space used by the instances themselves, Nova may overcommit the disk unintentionally by failing to consider the size of the image cache.

There are two approaches to addressing this situation:

1. **Mount the image cache as a separate filesystem**. This will cause Nova to report the amount of disk space available purely to instances, independent of how much is consumed by the cache. Nova will continue to disregard the size of the image cache and, if the cache space is exhausted, builds will fail. However, available disk space for instances will be correctly reported by `nova-compute` and accurately considered by the scheduler.
2. **Enable optional reserved disk amount behavior**. The configuration workaround *`workarounds.reserve_disk_resource_for_image_cache`* will cause `nova-compute` to periodically update the reserved disk amount to include the statically configured value, as well as the amount currently consumed by the image cache. This will cause the scheduler to see the available disk space decrease as the image cache grows. This is not updated synchronously and thus is not a perfect solution, but should vastly increase the schedulers visibility resulting in better decisions. (Note this solution is currently libvirt-specific)

As above, not all backends and virt drivers use image caching, and thus a third option may be to consider alternative infrastructure to eliminate this problem altogether.

4.3.13 Live-migrate instances

Live-migrating an instance means moving its virtual machine to a different OpenStack Compute server while the instance continues running. Before starting a live-migration, review the chapter *Configure live migrations*. It covers the configuration settings required to enable live-migration, but also reasons for migrations and non-live-migration options.

The instructions below cover shared-storage and volume-backed migration. To block-migrate instances, add the command-line option `-block-migrate` to the **`nova live-migration`** command, and `--block-migration` to the **`openstack server migrate`** command.

Manual selection of the destination host

1. Obtain the ID of the instance you want to migrate:

```
$ openstack server list

+--------------------------------------+------+--------+-----------------+--------
↪----+
| ID                                   | Name | Status | Networks        | Image␣
↪Name |
+--------------------------------------+------+--------+-----------------+--------
↪----+
```

(continues on next page)

(continued from previous page)

```
| d1df1b5a-70c4-4fed-98b7-423362f2c47c | vm1   | ACTIVE | private=a.b.c.d | ...    ␣
↪    |
| d693db9e-a7cf-45ef-a7c9-b3ecb5f22645 | vm2   | ACTIVE | private=e.f.g.h | ...    ␣
↪    |
+--------------------------------------+------+--------+-----------------+--------
↪----+
```

2. Determine on which host the instance is currently running. In this example, `vm1` is running on `HostB`:

```
$ openstack server show d1df1b5a-70c4-4fed-98b7-423362f2c47c

+----------------------+--------------------------------------+
| Field                | Value                                |
+----------------------+--------------------------------------+
| ...                  | ...                                  |
| OS-EXT-SRV-ATTR:host | HostB                                |
| ...                  | ...                                  |
| addresses            | a.b.c.d                              |
| flavor               | m1.tiny                              |
| id                   | d1df1b5a-70c4-4fed-98b7-423362f2c47c |
| name                 | vm1                                  |
| status               | ACTIVE                               |
| ...                  | ...                                  |
+----------------------+--------------------------------------+
```

3. Select the compute node the instance will be migrated to. In this example, we will migrate the instance to `HostC`, because `nova-compute` is running on it:

```
$ openstack compute service list

+----+------------------+-------+----------+---------+-------+--------------------
↪--------+
| ID | Binary           | Host  | Zone     | Status  | State | Updated At         ␣
↪        |
+----+------------------+-------+----------+---------+-------+--------------------
↪--------+
|  3 | nova-conductor   | HostA | internal | enabled | up    | 2017-02-
↪18T09:42:29.000000 |
|  4 | nova-scheduler   | HostA | internal | enabled | up    | 2017-02-
↪18T09:42:26.000000 |
|  5 | nova-compute     | HostB | nova     | enabled | up    | 2017-02-
↪18T09:42:29.000000 |
|  6 | nova-compute     | HostC | nova     | enabled | up    | 2017-02-
↪18T09:42:29.000000 |
+----+------------------+-------+----------+---------+-------+--------------------
↪--------+
```

4. Check that `HostC` has enough resources for migration:

```
$ openstack host show HostC

+-------+------------+-----+-----------+---------+
| Host  | Project    | CPU | Memory MB | Disk GB |
+-------+------------+-----+-----------+---------+
| HostC | (total)    |  16 |     32232 |     878 |
| HostC | (used_now) |  22 |     21284 |     422 |
```

(continues on next page)

(continued from previous page)

```
| HostC | (used_max) |  22 |     21284 |     422 |
| HostC | p1         |  22 |     21284 |     422 |
| HostC | p2         |  22 |     21284 |     422 |
+-------+------------+-----+-----------+---------+
```

- `cpu`: Number of CPUs
- `memory_mb`: Total amount of memory, in MB
- `disk_gb`: Total amount of space for NOVA-INST-DIR/instances, in GB

In this table, the first row shows the total amount of resources available on the physical server. The second line shows the currently used resources. The third line shows the maximum used resources. The fourth line and below shows the resources available for each project.

5. Migrate the instance:

```
$ openstack server migrate d1df1b5a-70c4-4fed-98b7-423362f2c47c --live HostC
```

6. Confirm that the instance has been migrated successfully:

```
$ openstack server show d1df1b5a-70c4-4fed-98b7-423362f2c47c

+----------------------+--------------------------------------+
| Field                | Value                                |
+----------------------+--------------------------------------+
| ...                  | ...                                  |
| OS-EXT-SRV-ATTR:host | HostC                                |
| ...                  | ...                                  |
+----------------------+--------------------------------------+
```

If the instance is still running on `HostB`, the migration failed. The `nova-scheduler` and `nova-conductor` log files on the controller and the `nova-compute` log file on the source compute host can help pin-point the problem.

Automatic selection of the destination host

To leave the selection of the destination host to the Compute service, use the nova command-line client.

1. Obtain the instance ID as shown in step 1 of the section *Manual selection of the destination host*.
2. Leave out the host selection steps 2, 3, and 4.
3. Migrate the instance:

```
$ nova live-migration d1df1b5a-70c4-4fed-98b7-423362f2c47c
```

Monitoring the migration

1. Confirm that the instance is migrating:

```
$ openstack server show d1df1b5a-70c4-4fed-98b7-423362f2c47c

+----------------------+--------------------------------------+
| Field                | Value                                |
+----------------------+--------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| ...                  | ...                                  |
| status               | MIGRATING                            |
| ...                  | ...                                  |
+----------------------+--------------------------------------+
```

2. Check progress

   Use the nova command-line client for novas migration monitoring feature. First, obtain the migration ID:

```
$ nova server-migration-list d1df1b5a-70c4-4fed-98b7-423362f2c47c
+----+-------------+-----------  (...)
| Id | Source Node | Dest Node | (...)
+----+-------------+-----------+ (...)
| 2  | -           | -         | (...)
+----+-------------+-----------+ (...)
```

   For readability, most output columns were removed. Only the first column, **Id**, is relevant. In this example, the migration ID is 2. Use this to get the migration status.

```
$ nova server-migration-show d1df1b5a-70c4-4fed-98b7-423362f2c47c 2
+------------------------+--------------------------------------+
| Property               | Value                                |
+------------------------+--------------------------------------+
| created_at             | 2017-03-08T02:53:06.000000           |
| dest_compute           | controller                           |
| dest_host              | -                                    |
| dest_node              | -                                    |
| disk_processed_bytes   | 0                                    |
| disk_remaining_bytes   | 0                                    |
| disk_total_bytes       | 0                                    |
| id                     | 2                                    |
| memory_processed_bytes | 65502513                             |
| memory_remaining_bytes | 786427904                            |
| memory_total_bytes     | 1091379200                           |
| server_uuid            | d1df1b5a-70c4-4fed-98b7-423362f2c47c |
| source_compute         | compute2                             |
| source_node            | -                                    |
| status                 | running                              |
| updated_at             | 2017-03-08T02:53:47.000000           |
+------------------------+--------------------------------------+
```

   The output shows that the migration is running. Progress is measured by the number of memory bytes that remain to be copied. If this number is not decreasing over time, the migration may be unable to complete, and it may be aborted by the Compute service.

Note: The command reports that no disk bytes are processed, even in the event of block migration.

What to do when the migration times out

During the migration process, the instance may write to a memory page after that page has been copied to the destination. When that happens, the same page has to be copied again. The instance may write to memory pages faster than they can be copied, so that the migration cannot complete. There are two optional actions, controlled by *libvirt.live_migration_timeout_action*, which can be taken against a VM after *libvirt.live_migration_completion_timeout* is reached:

1. `abort` (default): The live migration operation will be cancelled after the completion timeout is reached. This is similar to using API `DELETE /servers/{server_id}/migrations/{migration_id}`.
2. `force_complete`: The compute service will either pause the VM or trigger post-copy depending on if post copy is enabled and available (*`libvirt.live_migration_permit_post_copy`* is set to *True*). This is similar to using API `POST /servers/{server_id}/migrations/{migration_id}/action (force_complete)`.

You can also read the *`libvirt.live_migration_timeout_action`* configuration option help for more details.

The following remarks assume the KVM/Libvirt hypervisor.

How to know that the migration timed out

To determine that the migration timed out, inspect the `nova-compute` log file on the source host. The following log entry shows that the migration timed out:

```
# grep WARNING.*d1df1b5a-70c4-4fed-98b7-423362f2c47c /var/log/nova/nova-compute.log
...
WARNING nova.virt.libvirt.migration [req-...] [instance: ...]
live migration not completed after 1800 sec
```

Addressing migration timeouts

To stop the migration from putting load on infrastructure resources like network and disks, you may opt to cancel it manually.

```
$ nova live-migration-abort INSTANCE_ID MIGRATION_ID
```

To make live-migration succeed, you have several options:

- **Manually force-complete the migration**

  ```
  $ nova live-migration-force-complete INSTANCE_ID MIGRATION_ID
  ```

  The instance is paused until memory copy completes.

  > **Caution:** Since the pause impacts time keeping on the instance and not all applications tolerate incorrect time settings, use this approach with caution.

- **Enable auto-convergence**

  Auto-convergence is a Libvirt feature. Libvirt detects that the migration is unlikely to complete and slows down its CPU until the memory copy process is faster than the instances memory writes.

  To enable auto-convergence, set `live_migration_permit_auto_converge=true` in `nova.conf` and restart `nova-compute`. Do this on all compute hosts.

  > **Caution:** One possible downside of auto-convergence is the slowing down of the instance.

- **Enable post-copy**

This is a Libvirt feature. Libvirt detects that the migration does not progress and responds by activating the virtual machine on the destination host before all its memory has been copied. Access to missing memory pages result in page faults that are satisfied from the source host.

To enable post-copy, set `live_migration_permit_post_copy=true` in `nova.conf` and restart `nova-compute`. Do this on all compute hosts.

When post-copy is enabled, manual force-completion does not pause the instance but switches to the post-copy process.

Caution: Possible downsides:

- When the network connection between source and destination is interrupted, page faults cannot be resolved anymore, and the virtual machine is rebooted.
- Post-copy may lead to an increased page fault rate during migration, which can slow the instance down.

If live migrations routinely timeout or fail during cleanup operations due to the user token timing out, consider configuring nova to use *service user tokens*.

4.3.14 Logging

Logging module

Logging behavior can be changed by creating a configuration file. To specify the configuration file, add this line to the `/etc/nova/nova.conf` file:

```
log_config_append=/etc/nova/logging.conf
```

To change the logging level, add `DEBUG`, `INFO`, `WARNING`, or `ERROR` as a parameter.

The logging configuration file is an INI-style configuration file, which must contain a section called `logger_nova`. This controls the behavior of the logging facility in the `nova-*` services. For example:

```
[logger_nova]
level = INFO
handlers = stderr
qualname = nova
```

This example sets the debugging level to `INFO` (which is less verbose than the default `DEBUG` setting).

For more about the logging configuration syntax, including the `handlers` and `qualname` variables, see the Python documentation on logging configuration files.

For an example of the `logging.conf` file with various defined handlers, see the Example Configuration File for nova.

Syslog

OpenStack Compute services can send logging information to syslog. This is useful if you want to use rsyslog to forward logs to a remote machine. Separately configure the Compute service (nova), the Identity service (keystone), the Image service (glance), and, if you are using it, the Block Storage service (cinder) to send log messages to syslog. Open these configuration files:

- `/etc/nova/nova.conf`

- /etc/keystone/keystone.conf
- /etc/glance/glance-api.conf
- /etc/glance/glance-registry.conf
- /etc/cinder/cinder.conf

In each configuration file, add these lines:

```
debug = False
use_syslog = True
syslog_log_facility = LOG_LOCAL0
```

In addition to enabling syslog, these settings also turn off debugging output from the log.

Note: Although this example uses the same local facility for each service (LOG_LOCAL0, which corresponds to syslog facility LOCAL0), we recommend that you configure a separate local facility for each service, as this provides better isolation and more flexibility. For example, you can capture logging information at different severity levels for different services. syslog allows you to define up to eight local facilities, LOCAL0, LOCAL1, ..., LOCAL7. For more information, see the syslog documentation.

Rsyslog

rsyslog is useful for setting up a centralized log server across multiple machines. This section briefly describe the configuration to set up an rsyslog server. A full treatment of rsyslog is beyond the scope of this book. This section assumes rsyslog has already been installed on your hosts (it is installed by default on most Linux distributions).

This example provides a minimal configuration for /etc/rsyslog.conf on the log server host, which receives the log files

```
# provides TCP syslog reception
$ModLoad imtcp
$InputTCPServerRun 1024
```

Add a filter rule to /etc/rsyslog.conf which looks for a host name. This example uses COMPUTE_01 as the compute host name:

```
:hostname, isequal, "COMPUTE_01" /mnt/rsyslog/logs/compute-01.log
```

On each compute host, create a file named /etc/rsyslog.d/60-nova.conf, with the following content:

```
# prevent debug from dnsmasq with the daemon.none parameter
*.*;auth,authpriv.none,daemon.none,local0.none -/var/log/syslog
# Specify a log level of ERROR
local0.error    @@172.20.1.43:1024
```

Once you have created the file, restart the rsyslog service. Error-level log messages on the compute hosts should now be sent to the log server.

Serial console

The serial console provides a way to examine kernel output and other system messages during troubleshooting if the instance lacks network connectivity.

Read-only access from server serial console is possible using the `os-GetSerialOutput` server action. Most cloud images enable this feature by default. For more information, see *Common errors and fixes for Compute*.

OpenStack Juno and later supports read-write access using the serial console using the `os-GetSerialConsole` server action. This feature also requires a websocket client to access the serial console.

Configuring read-write serial console access

1. On a compute node, edit the `/etc/nova/nova.conf` file:

   In the `[serial_console]` section, enable the serial console:

```
[serial_console]
# ...
enabled = true
```

2. In the `[serial_console]` section, configure the serial console proxy similar to graphical console proxies:

```
[serial_console]
# ...
base_url = ws://controller:6083/
listen = 0.0.0.0
proxyclient_address = MANAGEMENT_INTERFACE_IP_ADDRESS
```

   The `base_url` option specifies the base URL that clients receive from the API upon requesting a serial console. Typically, this refers to the host name of the controller node.

   The `listen` option specifies the network interface nova-compute should listen on for virtual console connections. Typically, 0.0.0.0 will enable listening on all interfaces.

   The `proxyclient_address` option specifies which network interface the proxy should connect to. Typically, this refers to the IP address of the management interface.

   When you enable read-write serial console access, Compute will add serial console information to the Libvirt XML file for the instance. For example:

```
<console type='tcp'>
  <source mode='bind' host='127.0.0.1' service='10000'/>
  <protocol type='raw'/>
  <target type='serial' port='0'/>
  <alias name='serial0'/>
</console>
```

Accessing the serial console on an instance

1. Use the **`nova get-serial-proxy`** command to retrieve the websocket URL for the serial console on the instance:

```
$ nova get-serial-proxy INSTANCE_NAME
```

Type	Url
serial	ws://127.0.0.1:6083/?token=18510769-71ad-4e5a-8348-4218b5613b3d

   Alternatively, use the API directly:

```
$ curl -i 'http://<controller>:8774/v2.1/<tenant_uuid>/servers/<instance_uuid>/
↪action' \
  -X POST \
  -H "Accept: application/json" \
  -H "Content-Type: application/json" \
  -H "X-Auth-Project-Id: <project_id>" \
  -H "X-Auth-Token: <auth_token>" \
  -d '{"os-getSerialConsole": {"type": "serial"}}'
```

2. Use Python websocket with the URL to generate `.send`, `.recv`, and `.fileno` methods for serial console access. For example:

```
import websocket
ws = websocket.create_connection(
    'ws://127.0.0.1:6083/?token=18510769-71ad-4e5a-8348-4218b5613b3d',
    subprotocols=['binary', 'base64'])
```

Alternatively, use a Python websocket client.

Note: When you enable the serial console, typical instance logging using the **`nova console-log`** command is disabled. Kernel output and other system messages will not be visible unless you are actively viewing the serial console.

4.3.15 Manage the cloud

Show usage statistics for hosts and instances

You can show basic statistics on resource usage for hosts and instances.

Note: For more sophisticated monitoring, see the Ceilometer project. You can also use tools, such as Ganglia or Graphite, to gather more detailed data.

Show host usage statistics

The following examples show the host usage statistics for a host called `devstack`.

- List the hosts and the nova-related services that run on them:

```
$ openstack host list
+-----------+-------------+----------+
| Host Name | Service     | Zone     |
+-----------+-------------+----------+
| devstack  | conductor   | internal |
| devstack  | compute     | nova     |
| devstack  | network     | internal |
| devstack  | scheduler   | internal |
+-----------+-------------+----------+
```

- Get a summary of resource usage of all of the instances running on the host:

```
$ openstack host show devstack
+----------+----------------------------------+-----+-----------+---------+
| Host     | Project                          | CPU | MEMORY MB | DISK GB |
+----------+----------------------------------+-----+-----------+---------+
| devstack | (total)                          | 2   | 4003      | 157     |
| devstack | (used_now)                       | 3   | 5120      | 40      |
| devstack | (used_max)                       | 3   | 4608      | 40      |
| devstack | b70d90d65e464582b6b2161cf3603ced | 1   | 512       | 0       |
| devstack | 66265572db174a7aa66eba661f58eb9e | 2   | 4096      | 40      |
+----------+----------------------------------+-----+-----------+---------+
```

The `CPU` column shows the sum of the virtual CPUs for instances running on the host.

The `MEMORY MB` column shows the sum of the memory (in MB) allocated to the instances that run on the host.

The `DISK GB` column shows the sum of the root and ephemeral disk sizes (in GB) of the instances that run on the host.

The row that has the value `used_now` in the `PROJECT` column shows the sum of the resources allocated to the instances that run on the host, plus the resources allocated to the host itself.

The row that has the value `used_max` in the `PROJECT` column shows the sum of the resources allocated to the instances that run on the host.

Note: These values are computed by using information about the flavors of the instances that run on the hosts. This command does not query the CPU usage, memory usage, or hard disk usage of the physical host.

Show instance usage statistics

- Get CPU, memory, I/O, and network statistics for an instance.

  1. List instances:

```
$ openstack server list
+----------+----------------------+--------+------------------+--------+------
↪----+
| ID       | Name                 | Status | Networks         | Image  |␣
↪Flavor   |
+----------+----------------------+--------+------------------+--------+------
↪----+
| 84c6e... | myCirrosServer       | ACTIVE | private=10.0.0.3 | cirros | m1.
↪tiny  |
| 8a995... | myInstanceFromVolume | ACTIVE | private=10.0.0.4 | ubuntu | m1.
↪small |
+----------+----------------------+--------+------------------+--------+------
↪----+
```

  2. Get diagnostic statistics:

Note: As of microversion v2.48, diagnostics information for all virt drivers will have a standard format as below. Before microversion 2.48, each hypervisor had its own format. For more details on diagnostics response message see server diagnostics api documentation.

```
$ nova diagnostics myCirrosServer
+----------------+------------------------------------------------------------
↪------------+
| Property       | Value                                                      ␣
↪            |
+----------------+------------------------------------------------------------
↪------------+
| config_drive   | False                                                      ␣
↪            |
| cpu_details    | []                                                         ␣
↪            |
| disk_details   | [{"read_requests": 887, "errors_count": -1, "read_bytes":␣
↪20273152,    |
|                | "write_requests": 89, "write_bytes": 303104}]              ␣
↪            |
| driver         | libvirt                                                    ␣
↪            |
| hypervisor     | qemu                                                       ␣
↪            |
| hypervisor_os  | linux                                                      ␣
↪            |
| memory_details | {"used": 0, "maximum": 0}                                  ␣
↪            |
| nic_details    | [{"rx_packets": 9, "rx_drop": 0, "tx_octets": 1464, "tx_
↪errors": 0,    |
|                | "mac_address": "fa:16:3e:fa:db:d3", "rx_octets": 958, "rx_
↪rate": null, |
|                | "rx_errors": 0, "tx_drop": 0, "tx_packets": 9, "tx_rate":␣
↪null}]       |
| num_cpus       | 0                                                          ␣
↪            |
| num_disks      | 1                                                          ␣
↪            |
| num_nics       | 1                                                          ␣
↪            |
| state          | running                                                    ␣
↪            |
| uptime         | 5528                                                       ␣
↪            |
+----------------+------------------------------------------------------------
↪------------+
```

`config_drive` indicates if the config drive is supported on the instance.

`cpu_details` contains a list of details per vCPU.

`disk_details` contains a list of details per disk.

`driver` indicates the current driver on which the VM is running.

`hypervisor` indicates the current hypervisor on which the VM is running.

`nic_details` contains a list of details per vNIC.

`uptime` is the amount of time in seconds that the VM has been running.

Diagnostics prior to v2.48:

```
$ nova diagnostics myCirrosServer
+----------------------------+--------+
| Property                   | Value  |
+----------------------------+--------+
| memory                     | 524288 |
| memory-actual              | 524288 |
| memory-rss                 | 6444   |
| tap1fec8fb8-7a_rx          | 22137  |
| tap1fec8fb8-7a_rx_drop     | 0      |
| tap1fec8fb8-7a_rx_errors   | 0      |
| tap1fec8fb8-7a_rx_packets  | 166    |
| tap1fec8fb8-7a_tx          | 18032  |
| tap1fec8fb8-7a_tx_drop     | 0      |
| tap1fec8fb8-7a_tx_errors   | 0      |
| tap1fec8fb8-7a_tx_packets  | 130    |
| vda_errors                 | -1     |
| vda_read                   | 2048   |
| vda_read_req               | 2      |
| vda_write                  | 182272 |
| vda_write_req              | 74     |
+----------------------------+--------+
```

- Get summary statistics for each project:

```
$ openstack usage list
Usage from 2013-06-25 to 2013-07-24:
+---------+---------+--------------+-----------+---------------+
| Project | Servers | RAM MB-Hours | CPU Hours | Disk GB-Hours |
+---------+---------+--------------+-----------+---------------+
| demo    | 1       | 344064.44    | 672.00    | 0.00          |
| stack   | 3       | 671626.76    | 327.94    | 6558.86       |
+---------+---------+--------------+-----------+---------------+
```

System administrators can use the **`openstack`** to manage their clouds.

The `openstack` client can be used by all users, though specific commands might be restricted by the Identity service.

Managing the cloud with the openstack client

1. The `python-openstackclient` package provides an `openstack` shell that enables Compute API interactions from the command line. Install the client, and provide your user name and password (which can be set as environment variables for convenience), for the ability to administer the cloud from the command line.

   For more information on `python-openstackclient`, refer to the documentation.

2. Confirm the installation was successful:

```
$ openstack help
usage: openstack [--version] [-v | -q] [--log-file LOG_FILE] [-h] [--debug]
           [--os-cloud <cloud-config-name>]
           [--os-region-name <auth-region-name>]
           [--os-cacert <ca-bundle-file>] [--verify | --insecure]
           [--os-default-domain <auth-domain>]
           ...
```

   Running **`openstack help`** returns a list of `openstack` commands and parameters. To get help for a subcommand, run:

```
$ openstack help SUBCOMMAND
```

For a complete list of `openstack` commands and parameters, refer to the OpenStack Command-Line Reference.

3. Set the required parameters as environment variables to make running commands easier. For example, you can add `--os-username` as an `openstack` option, or set it as an environment variable. To set the user name, password, and project as environment variables, use:

```
$ export OS_USERNAME=joecool
$ export OS_PASSWORD=coolword
$ export OS_TENANT_NAME=coolu
```

4. The Identity service gives you an authentication endpoint, which Compute recognizes as `OS_AUTH_URL`:

```
$ export OS_AUTH_URL=http://hostname:5000/v2.0
```

4.3.16 Manage Compute users

Access to the Euca2ools (ec2) API is controlled by an access key and a secret key. The users access key needs to be included in the request, and the request must be signed with the secret key. Upon receipt of API requests, Compute verifies the signature and runs commands on behalf of the user.

To begin using Compute, you must create a user with the Identity service.

4.3.17 Manage volumes

Depending on the setup of your cloud provider, they may give you an endpoint to use to manage volumes. You can use the `openstack` CLI to manage volumes.

For the purposes of the compute service, attaching, detaching and *creating a server from a volume* are of primary interest.

Refer to the CLI documentation for more information.

Volume multi-attach

Nova added support for multiattach volumes in the 17.0.0 Queens release.

This document covers the nova-specific aspects of this feature. Refer to the block storage admin guide for more details about creating multiattach-capable volumes.

Boot from volume and attaching a volume to a server that is not SHELVED_OFFLOADED is supported. Ultimately the ability to perform these actions depends on the compute host and hypervisor driver that is being used.

There is also a recorded overview and demo for volume multi-attach.

Requirements

- The minimum required compute API microversion for attaching a multiattach-capable volume to more than one server is *2.60*.
- Cinder 12.0.0 (Queens) or newer is required.

- The `nova-compute` service must be running at least Queens release level code (17.0.0) and the hypervisor driver must support attaching block storage devices to more than one guest. Refer to *Feature Support Matrix* for details on which compute drivers support volume multiattach.
- When using the libvirt compute driver, the following native package versions determine multiattach support:
  - libvirt must be greater than or equal to 3.10, or
  - qemu must be less than 2.10
- Swapping an *in-use* multiattach volume is not supported (this is actually controlled via the block storage volume retype API).

Known issues

- Creating multiple servers in a single request with a multiattach-capable volume as the root disk is not yet supported: https://bugs.launchpad.net/nova/+bug/1747985
- Subsequent attachments to the same volume are all attached in *read/write* mode by default in the block storage service. A future change either in nova or cinder may address this so that subsequent attachments are made in *read-only* mode, or such that the mode can be specified by the user when attaching the volume to the server.

Testing

Continuous integration testing of the volume multiattach feature is done via the `tempest-full` and `tempest-slow` jobs, which, along with the tests themselves, are defined in the tempest repository.

4.3.18 Metadata service

Note: This section provides deployment information about the metadata service. For end-user information about the metadata service and instance metadata in general, refer to the *user guide*.

Note: This section provides deployment information about the metadata service using neutron. This functions very differently when deployed with the deprecated **`nova-network`** service.

For information about deploying the metadata service with the **`nova-network`** service, refer to the *nova-network documentation*

The metadata service provides a way for instances to retrieve instance-specific data. Instances access the metadata service at `http://169.254.169.254`. The metadata service supports two sets of APIs - an OpenStack metadata API and an EC2-compatible API - and also exposes vendordata and user data. Both the OpenStack metadata and EC2-compatible APIs are versioned by date.

The metadata service can be run globally, as part of the **`nova-api`** application, or on a per-cell basis, as part of the standalone **`nova-api-metadata`** application. A detailed comparison is provided in the *cells V2 guide*.

Changed in version 19.0.0: The ability to run the nova metadata API service on a per-cell basis was added in Stein. For versions prior to this release, you should not use the standalone **`nova-api-metadata`** application for multiple cells.

Guests access the service at `169.254.169.254`. The networking service, neutron, is responsible for intercepting these requests and adding HTTP headers which uniquely identify the source of the request before forwarding it to

the metadata API server. For the Open vSwitch and Linux Bridge backends provided with neutron, the flow looks something like so:

1. Instance sends a HTTP request for metadata to `169.254.169.254`.
2. This request either hits the router or DHCP namespace depending on the route in the instance
3. The metadata proxy service in the namespace adds the following info to the request:
   - Instance IP (`X-Forwarded-For` header)
   - Router or Network-ID (`X-Neutron-Network-Id` or `X-Neutron-Router-Id` header)
4. The metadata proxy service sends this request to the metadata agent (outside the namespace) via a UNIX domain socket.
5. The **`neutron-metadata-agent`** application forwards the request to the nova metadata API service by adding some new headers (instance ID and Tenant ID) to the request.

This flow may vary if a different networking backend is used.

Neutron and nova must be configured to communicate together with a shared secret. Neutron uses this secret to sign the Instance-ID header of the metadata request to prevent spoofing. This secret is configured through the *`neutron.metadata_proxy_shared_secret`* config option in nova and the equivalent `metadata_proxy_shared_secret` config option in neutron.

Configuration

The **`nova-api`** application accepts the following metadata service-related options:

- *`enabled_apis`*
- *`enabled_ssl_apis`*
- *`neutron.service_metadata_proxy`*
- *`neutron.metadata_proxy_shared_secret`*
- *`api.metadata_cache_expiration`*
- *`api.use_forwarded_for`*
- *`api.local_metadata_per_cell`*
- *`api.dhcp_domain`*

Note: This list excludes configuration options related to the vendordata feature. Refer to *vendordata feature documentation* for information on configuring this.

For example, to configure the **`nova-api`** application to serve the metadata API, without SSL, using the `StaticJSON` vendordata provider, add the following to a `nova-api.conf` file:

```
[DEFAULT]
enabled_apis = osapi_compute,metadata
enabled_ssl_apis =
metadata_listen = 0.0.0.0
metadata_listen_port = 0
metadata_workers = 4

[neutron]
service_metadata_proxy = True
```

(continues on next page)

(continued from previous page)

```
[api]
dhcp_domain =
metadata_cache_expiration = 15
use_forwarded_for = False
local_metadata_per_cell = False
vendordata_providers = StaticJSON
vendordata_jsonfile_path = /etc/nova/vendor_data.json
```

Note: This does not include configuration options that are not metadata-specific but are nonetheless required, such as *api.auth_strategy*.

Configuring the application to use the `DynamicJSON` vendordata provider is more involved and is not covered here.

The **nova-api-metadata** application accepts almost the same options:

- *neutron.service_metadata_proxy*
- *neutron.metadata_proxy_shared_secret*
- *api.metadata_cache_expiration*
- *api.use_forwarded_for*
- *api.local_metadata_per_cell*
- *api.dhcp_domain*

Note: This list excludes configuration options related to the vendordata feature. Refer to *vendordata feature documentation* for information on configuring this.

For example, to configure the **nova-api-metadata** application to serve the metadata API, without SSL, add the following to a `nova-api.conf` file:

```
[DEFAULT]
metadata_listen = 0.0.0.0
metadata_listen_port = 0
metadata_workers = 4

[neutron]
service_metadata_proxy = True

[api]
dhcp_domain =
metadata_cache_expiration = 15
use_forwarded_for = False
local_metadata_per_cell = False
```

Note: This does not include configuration options that are not metadata-specific but are nonetheless required, such as *api.auth_strategy*.

For information about configuring the neutron side of the metadata service, refer to the neutron configuration guide

Config drives

Config drives are special drives that are attached to an instance when it boots. The instance can mount this drive and read files from it to get information that is normally available through the metadata service. For more information, refer to *Config drives* and the *user guide*.

Vendordata

Vendordata provides a way to pass vendor or deployment-specific information to instances. For more information, refer to *Vendordata* and the *user guide*.

User data

User data is a blob of data that the user can specify when they launch an instance. For more information, refer to *the user guide*.

4.3.19 Migrate instances

Note: This documentation is about cold-migration. For live-migration usage, see *Live-migrate instances*.

When you want to move an instance from one compute host to another, you can use the **`openstack server migrate`** command. The scheduler chooses the destination compute host based on its settings. This process does not assume that the instance has shared storage available on the target host. If you are using SSH tunneling, you must ensure that each node is configured with SSH key authentication so that the Compute service can use SSH to move disks to other nodes. For more information, see *Configure SSH between compute nodes*.

1. To list the VMs you want to migrate, run:

```
$ openstack server list
```

2. Use the **`openstack server migrate`** command.

```
$ openstack server migrate VM_INSTANCE
```

3. To migrate an instance and watch the status, use this example script:

```
#!/bin/bash

# Provide usage
usage() {
    echo "Usage: $0 VM_ID"
    exit 1
}

[[ $# -eq 0 ]] && usage
VM_ID=$1

# Show the details for the VM
echo "Instance details:"
openstack server show ${VM_ID}

# Migrate the VM to an alternate hypervisor
```

(continues on next page)

(continued from previous page)

```
echo -n "Migrating instance to alternate host "
openstack server migrate ${VM_ID}
while [[ "$(openstack server show ${VM_ID} -f value -c status)" != "VERIFY_RESIZE
↪" ]]; do
    echo -n "."
    sleep 2
done
openstack server resize --confirm ${VM_ID}
echo " instance migrated and resized."

# Show the details for the migrated VM
echo "Migrated instance details:"
openstack server show ${VM_ID}

# Pause to allow users to examine VM details
read -p "Pausing, press <enter> to exit."
```

Note: If you see the following error, it means you are either running the command with the wrong credentials, such as a non-admin user, or the `policy.json` file prevents migration for your user:

```
Policy doesn't allow os_compute_api:os-migrate-server:migrate to be performed. (HTTP␣
↪403)
```

Note: If you see the following error, similar to this message, SSH tunneling was not set up between the compute nodes:

```
ProcessExecutionError: Unexpected error while running command.
Stderr: u Host key verification failed.\r\n
```

The instance is booted from a new host, but preserves its configuration including instance ID, name, IP address, any metadata, and other properties.

4.3.20 Use snapshots to migrate instances

This guide can be used to migrate an instance between different clouds.

To use snapshots to migrate instances from OpenStack projects to clouds, complete these steps.

In the source project:

1. *Create a snapshot of the instance*
2. *Download the snapshot as an image*

In the destination project:

1. *Import the snapshot to the new environment*
2. *Boot a new instance from the snapshot*

Note: Some cloud providers allow only administrators to perform this task.

Create a snapshot of the instance

1. Shut down the source VM before you take the snapshot to ensure that all data is flushed to disk. If necessary, list the instances to view the instance name:

```
$ openstack server list
+--------------------------------------+------------+--------+------------------+-
↪------------------+-------------------------+
| ID                                   | Name       | Status | Networks         |␣
↪Image               | Flavor                  |
+--------------------------------------+------------+--------+------------------+-
↪------------------+-------------------------+
| d0d1b7d9-a6a5-41d3-96ab-07975aadd7fb | myInstance | ACTIVE | private=10.0.0.3 |␣
↪ubuntu-16.04-amd64 | general.micro.tmp.linux |
+--------------------------------------+------------+--------+------------------+-
↪------------------+-------------------------+
```

2. Use the **openstack server stop** command to shut down the instance:

```
$ openstack server stop myInstance
```

3. Use the **openstack server list** command to confirm that the instance shows a SHUTOFF status:

```
$ openstack server list
+--------------------------------------+------------+---------+------------------
↪+--------------------+-------------------------+
| ID                                   | Name       | Status  | Networks         ␣
↪| Image               | Flavor                  |
+--------------------------------------+------------+---------+------------------
↪+--------------------+-------------------------+
| d0d1b7d9-a6a5-41d3-96ab-07975aadd7fb | myInstance | SHUTOFF | private=10.0.0.3␣
↪| ubuntu-16.04-amd64 | general.micro.tmp.linux |
+--------------------------------------+------------+---------+------------------
↪+--------------------+-------------------------+
```

4. Use the **openstack server image create** command to take a snapshot:

```
$ openstack server image create --name myInstanceSnapshot myInstance
```

   If snapshot operations routinely fail because the user token times out while uploading a large disk image, consider configuring nova to use *service user tokens*.

5. Use the **openstack image list** command to check the status until the status is ACTIVE:

```
$ openstack image list
+--------------------------------------+---------------------------+--------+
| ID                                   | Name                      | Status |
+--------------------------------------+---------------------------+--------+
| ab567a44-b670-4d22-8ead-80050dfcd280 | myInstanceSnapshot        | active |
+--------------------------------------+---------------------------+--------+
```

Download the snapshot as an image

1. Get the image ID:

```
$ openstack image list
+--------------------------------------+---------------------------+--------+
| ID                                   | Name                      | Status |
+--------------------------------------+---------------------------+--------+
| ab567a44-b670-4d22-8ead-80050dfcd280 | myInstanceSnapshot        | active |
+--------------------------------------+---------------------------+--------+
```

2. Download the snapshot by using the image ID that was returned in the previous step:

```
$ openstack image save --file snapshot.raw ab567a44-b670-4d22-8ead-80050dfcd280
```

Note: The **`openstack image save`** command requires the image ID and cannot use the image name. Check there is sufficient space on the destination file system for the image file.

---

3. Make the image available to the new environment, either through HTTP or direct upload to a machine (`scp`).

Import the snapshot to the new environment

In the new project or cloud environment, import the snapshot:

```
$ openstack image create --container-format bare --disk-format qcow2 \
  --file snapshot.raw myInstanceSnapshot
```

Boot a new instance from the snapshot

In the new project or cloud environment, use the snapshot to create the new instance:

```
$ openstack server create --flavor m1.tiny --image myInstanceSnapshot myNewInstance
```

4.3.21 Networking with nova-network

Deprecated since version 14.0.0: `nova-network` was deprecated in the OpenStack Newton release. In Ocata and future releases, you can start `nova-network` only with a cells v1 configuration. This is not a recommended configuration for deployment.

Understanding the networking configuration options helps you design the best configuration for your Compute instances.

You can choose to either install and configure `nova-network` or use the OpenStack Networking service (neutron). This section contains a brief overview of `nova-network`. For more information about OpenStack Networking, refer to the documentation.

Networking concepts

Compute assigns a private IP address to each VM instance. Compute makes a distinction between fixed IPs and floating IP. Fixed IPs are IP addresses that are assigned to an instance on creation and stay the same until the instance is explicitly terminated. Floating IPs are addresses that can be dynamically associated with an instance. A floating IP address can be disassociated and associated with another instance at any time. A user can reserve a floating IP for their project.

Note: Currently, Compute with `nova-network` only supports Linux bridge networking that allows virtual interfaces to connect to the outside network through the physical interface.

The network controller with `nova-network` provides virtual networks to enable compute servers to interact with each other and with the public network. Compute with `nova-network` supports the following network modes, which are implemented as Network Manager types:

Flat Network Manager In this mode, a network administrator specifies a subnet. IP addresses for VM instances are assigned from the subnet, and then injected into the image on launch. Each instance receives a fixed IP address from the pool of available addresses. A system administrator must create the Linux networking bridge (typically named `br100`, although this is configurable) on the systems running the `nova-network` service. All instances of the system are attached to the same bridge, which is configured manually by the network administrator.

Note: Configuration injection currently only works on Linux-style systems that keep networking configuration in `/etc/network/interfaces`.

Flat DHCP Network Manager In this mode, OpenStack starts a DHCP server (dnsmasq) to allocate IP addresses to VM instances from the specified subnet, in addition to manually configuring the networking bridge. IP addresses for VM instances are assigned from a subnet specified by the network administrator.

Like flat mode, all instances are attached to a single bridge on the compute node. Additionally, a DHCP server configures instances depending on single-/multi-host mode, alongside each `nova-network`. In this mode, Compute does a bit more configuration. It attempts to bridge into an Ethernet device (`flat_interface`, eth0 by default). For every instance, Compute allocates a fixed IP address and configures dnsmasq with the MAC ID and IP address for the VM. Dnsmasq does not take part in the IP address allocation process, it only hands out IPs according to the mapping done by Compute. Instances receive their fixed IPs with the **`dhcpdiscover`** command. These IPs are not assigned to any of the hosts network interfaces, only to the guest-side interface for the VM.

In any setup with flat networking, the hosts providing the `nova-network` service are responsible for forwarding traffic from the private network. They also run and configure dnsmasq as a DHCP server listening on this bridge, usually on IP address 10.0.0.1 (see *DHCP server: dnsmasq*). Compute can determine the NAT entries for each network, although sometimes NAT is not used, such as when the network has been configured with all public IPs, or if a hardware router is used (which is a high availability option). In this case, hosts need to have `br100` configured and physically connected to any other nodes that are hosting VMs. You must set the `flat_network_bridge` option or create networks with the bridge parameter in order to avoid raising an error. Compute nodes have iptables or ebtables entries created for each project and instance to protect against MAC ID or IP address spoofing and ARP poisoning.

Note: In single-host Flat DHCP mode you will be able to ping VMs through their fixed IP from the `nova-network` node, but you cannot ping them from the compute nodes. This is expected behavior.

VLAN Network Manager This is the default mode for OpenStack Compute. In this mode, Compute creates a VLAN and bridge for each project. For multiple-machine installations, the VLAN Network Mode requires a switch that supports VLAN tagging (IEEE 802.1Q). The project gets a range of private IPs that are only accessible from inside the VLAN. In order for a user to access the instances in their project, a special VPN instance (code named `cloudpipe`) needs to be created. Compute generates a certificate and key for the user to access the VPN and starts the VPN automatically. It provides a private network segment for each projects instances that can be accessed through a dedicated VPN connection from the internet. In this mode, each project gets its own VLAN, Linux networking bridge, and subnet.

The subnets are specified by the network administrator, and are assigned dynamically to a project when required. A DHCP server is started for each VLAN to pass out IP addresses to VM instances from the subnet assigned to the project. All instances belonging to one project are bridged into the same VLAN for that project. OpenStack Compute creates the Linux networking bridges and VLANs when required.

These network managers can co-exist in a cloud system. However, because you cannot select the type of network for a given project, you cannot configure multiple network types in a single Compute installation.

All network managers configure the network using network drivers. For example, the Linux L3 driver (`l3.py` and `linux_net.py`), which makes use of `iptables`, `route` and other network management facilities, and the libvirt network filtering facilities. The driver is not tied to any particular network manager; all network managers use the same driver. The driver usually initializes only when the first VM lands on this host node.

All network managers operate in either single-host or multi-host mode. This choice greatly influences the network configuration. In single-host mode, a single `nova-network` service provides a default gateway for VMs and hosts a single DHCP server (dnsmasq). In multi-host mode, each compute node runs its own `nova-network` service. In both cases, all traffic between VMs and the internet flows through `nova-network`. Each mode has benefits and drawbacks.

All networking options require network connectivity to be already set up between OpenStack physical nodes. OpenStack does not configure any physical network interfaces. All network managers automatically create VM virtual interfaces. Some network managers can also create network bridges such as `br100`.

The internal network interface is used for communication with VMs. The interface should not have an IP address attached to it before OpenStack installation, it serves only as a fabric where the actual endpoints are VMs and dnsmasq. Additionally, the internal network interface must be in `promiscuous` mode, so that it can receive packets whose target MAC address is the guest VM, not the host.

All machines must have a public and internal network interface (controlled by these options: `public_interface` for the public interface, and `flat_interface` and `vlan_interface` for the internal interface with flat or VLAN managers). This guide refers to the public network as the external network and the private network as the internal or project network.

For flat and flat DHCP modes, use the **`nova network-create`** command to create a network:

```
$ nova network-create vmnet \
  --fixed-range-v4 10.0.0.0/16 --fixed-cidr 10.0.20.0/24 --bridge br100
```

This example uses the following parameters:

`--fixed-range-v4` Specifies the network subnet.

`--fixed-cidr` Specifies a range of fixed IP addresses to allocate, and can be a subset of the `--fixed-range-v4` argument.

`--bridge` Specifies the bridge device to which this network is connected on every compute node.

DHCP server: dnsmasq

The Compute service uses dnsmasq as the DHCP server when using either Flat DHCP Network Manager or VLAN Network Manager. For Compute to operate in IPv4/IPv6 dual-stack mode, use at least dnsmasq v2.63. The `nova-network` service is responsible for starting dnsmasq processes.

The behavior of dnsmasq can be customized by creating a dnsmasq configuration file. Specify the configuration file using the `dnsmasq_config_file` configuration option:

```
dnsmasq_config_file=/etc/dnsmasq-nova.conf
```

For more information about creating a dnsmasq configuration file, see the *Configuration Options*, and the dnsmasq documentation.

Dnsmasq also acts as a caching DNS server for instances. You can specify the DNS server that dnsmasq uses by setting the `dns_server` configuration option in `/etc/nova/nova.conf`. This example configures dnsmasq to use Googles public DNS server:

```
dns_server=8.8.8.8
```

Dnsmasq logs to syslog (typically `/var/log/syslog` or `/var/log/messages`, depending on Linux distribution). Logs can be useful for troubleshooting, especially in a situation where VM instances boot successfully but are not reachable over the network.

Administrators can specify the starting point IP address to reserve with the DHCP server (in the format n.n.n.n) with this command:

```
$ nova-manage fixed reserve --address IP_ADDRESS
```

This reservation only affects which IP address the VMs start at, not the fixed IP addresses that `nova-network` places on the bridges.

Configure Compute to use IPv6 addresses

If you are using OpenStack Compute with `nova-network`, you can put Compute into dual-stack mode, so that it uses both IPv4 and IPv6 addresses for communication. In dual-stack mode, instances can acquire their IPv6 global unicast addresses by using a stateless address auto-configuration mechanism [RFC 4862/2462]. IPv4/IPv6 dual-stack mode works with both `VlanManager` and `FlatDHCPManager` networking modes.

In `VlanManager` networking mode, each project uses a different 64-bit global routing prefix. In `FlatDHCPManager` mode, all instances use one 64-bit global routing prefix.

This configuration was tested with virtual machine images that have an IPv6 stateless address auto-configuration capability. This capability is required for any VM to run with an IPv6 address. You must use an EUI-64 address for stateless address auto-configuration. Each node that executes a `nova-*` service must have `python-netaddr` and `radvd` installed.

Switch into IPv4/IPv6 dual-stack mode

1. For every node running a `nova-*` service, install `python-netaddr`:

   ```
   # apt-get install python-netaddr
   ```

2. For every node running `nova-network`, install `radvd` and configure IPv6 networking:

   ```
   # apt-get install radvd
   # echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
   # echo 0 > /proc/sys/net/ipv6/conf/all/accept_ra
   ```

3. On all nodes, edit the `nova.conf` file and specify `use_ipv6 = True`.

4. Restart all `nova-*` services.

IPv6 configuration options

You can use the following options with the **`nova network-create`** command:

- Add a fixed range for IPv6 addresses to the **nova network-create** command. Specify public or private after the network-create parameter.

```
$ nova network-create public --fixed-range-v4 FIXED_RANGE_V4 \
  --vlan VLAN_ID --vpn VPN_START --fixed-range-v6 FIXED_RANGE_V6
```

- Set the IPv6 global routing prefix by using the --fixed_range_v6 parameter. The default value for the parameter is fd00::/48.

  When you use FlatDHCPManager, the command uses the original --fixed_range_v6 value. For example:

```
$ nova network-create public  --fixed-range-v4 10.0.2.0/24 \
  --fixed-range-v6 fd00:1::/48
```

- When you use VlanManager, the command increments the subnet ID to create subnet prefixes. Guest VMs use this prefix to generate their IPv6 global unicast addresses. For example:

```
$ nova network-create public --fixed-range-v4 10.0.1.0/24 --vlan 100 \
  --vpn 1000 --fixed-range-v6 fd00:1::/48
```

Table 7: Description of IPv6 configuration options

Configuration option = Default value	Description
[DEFAULT]	
fixed_range_v6 = fd00::/48	(StrOpt) Fixed IPv6 address block
gateway_v6 = None	(StrOpt) Default IPv6 gateway
ipv6_backend = rfc2462	(StrOpt) Backend to use for IPv6 generation
use_ipv6 = False	(BoolOpt) Use IPv6

Metadata service

Note: This section provides deployment information about the metadata service using the deprecated **nova-network** service.

For information about deploying the metadata service with neutron, refer to the *metadata service documentation*.

For end-user information about the metadata service and instance metadata in general, refer to the *user guide*.

The metadata service is implemented by either the nova-api service or the nova-api-metadata service. Note that the nova-api-metadata service is generally only used when running in multi-host mode, as it retrieves instance-specific metadata. If you are running the nova-api service, you must have metadata as one of the elements listed in the enabled_apis configuration option in /etc/nova/nova.conf. The default enabled_apis configuration setting includes the metadata service, so you do not need to modify it.

Hosts access the service at 169.254.169.254:80, and this is translated to metadata_host:metadata_port by an iptables rule established by the nova-network service. In multi-host mode, you can set metadata_host to 127.0.0.1.

For instances to reach the metadata service, the nova-network service must configure iptables to NAT port 80 of the 169.254.169.254 address to the IP address specified in metadata_host (this defaults to $my_ip, which is the IP address of the nova-network service) and port specified in metadata_port (which defaults to 8775) in /etc/nova/nova.conf.

Note: The `metadata_host` configuration option must be an IP address, not a host name.

The default Compute service settings assume that `nova-network` and `nova-api` are running on the same host. If this is not the case, in the `/etc/nova/nova.conf` file on the host running `nova-network`, set the `metadata_host` configuration option to the IP address of the host where `nova-api` is running.

Table 8: Description of metadata configuration options

Configuration option = Default value	Description
[DEFAULT]	
metadata_host = $my_ip	(StrOpt) The IP address for the metadata API server
metadata_listen = 0.0.0.0	(StrOpt) The IP address on which the metadata API will listen.
metadata_listen_port = 8775	(IntOpt) The port on which the metadata API will listen.
metadata_port = 8775	(IntOpt) The port for the metadata API port
metadata_workers = None	(IntOpt) Number of workers for metadata service. The default will be the number of CPUs available.
[api]	
metadata_cache_expiration = 15	(IntOpt) Time in seconds to cache metadata; 0 to disable metadata caching entirely (not recommended). Increasing this should improve response times of the metadata API when under heavy load. Higher values may increase memory usage and result in longer times for host metadata changes to take effect.
vendordata_providers = StaticJSON	(ListOpt) A list of vendordata providers. See *Vendordata* for more information.
vendordata_jsonfile_path = None	(StrOpt) File to load JSON formatted vendor data from

Enable ping and SSH on VMs

You need to enable `ping` and `ssh` on your VMs for network access. This can be done with the **`openstack`** command.

Note: Run these commands as root only if the credentials used to interact with `nova-api` are in `/root/.bashrc`.

Enable ping and SSH with **`openstack security group rule create`** commands:

```
$ openstack security group rule create --protocol icmp default
$ openstack security group rule create --protocol tcp --dst-port 22:22 default
```

If you have run these commands and still cannot ping or SSH your instances, check the number of running `dnsmasq` processes, there should be two. If not, kill the processes and restart the service with these commands:

```
# killall dnsmasq
# service nova-network restart
```

Configure public (floating) IP addresses

This section describes how to configure floating IP addresses with `nova-network`. For information about doing this with OpenStack Networking, refer to L3-routing-and-NAT.

Private and public IP addresses

In this section, the term floating IP address is used to refer to an IP address, usually public, that you can dynamically add to a running virtual instance.

Every virtual instance is automatically assigned a private IP address. You can choose to assign a public (or floating) IP address instead. OpenStack Compute uses network address translation (NAT) to assign floating IPs to virtual instances.

To be able to assign a floating IP address, edit the `/etc/nova/nova.conf` file to specify which interface the `nova-network` service should bind public IP addresses to:

```
public_interface=VLAN100
```

If you make changes to the `/etc/nova/nova.conf` file while the `nova-network` service is running, you will need to restart the service to pick up the changes.

Note: Floating IPs are implemented by using a source NAT (SNAT rule in iptables), so security groups can sometimes display inconsistent behavior if VMs use their floating IP to communicate with other VMs, particularly on the same physical host. Traffic from VM to VM across the fixed network does not have this issue, and so this is the recommended setup. To ensure that traffic does not get SNATed to the floating range, explicitly set:

```
dmz_cidr=x.x.x.x/y
```

The `x.x.x.x/y` value specifies the range of floating IPs for each pool of floating IPs that you define. This configuration is also required if the VMs in the source group have floating IPs.

Enable IP forwarding

IP forwarding is disabled by default on most Linux distributions. You will need to enable it in order to use floating IPs.

Note: IP forwarding only needs to be enabled on the nodes that run `nova-network`. However, you will need to enable it on all compute nodes if you use `multi_host` mode.

To check if IP forwarding is enabled, run:

```
$ cat /proc/sys/net/ipv4/ip_forward
0
```

Alternatively, run:

```
$ sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
```

In these examples, IP forwarding is disabled.

To enable IP forwarding dynamically, run:

```
# sysctl -w net.ipv4.ip_forward=1
```

Alternatively, run:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

To make the changes permanent, edit the `/etc/sysctl.conf` file and update the IP forwarding setting:

```
net.ipv4.ip_forward = 1
```

Save the file and run this command to apply the changes:

```
# sysctl -p
```

You can also apply the changes by restarting the network service:

- on Ubuntu, Debian:

  ```
  # /etc/init.d/networking restart
  ```

- on RHEL, Fedora, CentOS, openSUSE and SLES:

  ```
  # service network restart
  ```

Create a list of available floating IP addresses

Compute maintains a list of floating IP addresses that are available for assigning to instances. Use the **`nova-manage floating`** commands to perform floating IP operations:

- Add entries to the list:

  ```
  # nova-manage floating create --pool nova --ip_range 68.99.26.170/31
  ```

- List the floating IP addresses in the pool:

  ```
  # openstack floating ip list
  ```

- Create specific floating IPs for either a single address or a subnet:

  ```
  # nova-manage floating create --pool POOL_NAME --ip_range CIDR
  ```

- Remove floating IP addresses using the same parameters as the create command:

  ```
  # openstack floating ip delete CIDR
  ```

For more information about how administrators can associate floating IPs with instances, see ip floating in the *python-openstackclient* User Documentation.

Automatically add floating IPs

You can configure `nova-network` to automatically allocate and assign a floating IP address to virtual instances when they are launched. Add this line to the `/etc/nova/nova.conf` file:

```
auto_assign_floating_ip=True
```

Save the file, and restart `nova-network`

Note: If this option is enabled, but all floating IP addresses have already been allocated, the **`openstack server create`** command will fail.

Remove a network from a project

You cannot delete a network that has been associated to a project. This section describes the procedure for dissociating it so that it can be deleted.

In order to disassociate the network, you will need the ID of the project it has been associated to. To get the project ID, you will need to be an administrator.

Disassociate the network from the project using the **`nova-manage project scrub`** command, with the project ID as the final parameter:

```
# nova-manage project scrub --project ID
```

Multiple interfaces for instances (multinic)

The multinic feature allows you to use more than one interface with your instances. This is useful in several scenarios:

- SSL Configurations (VIPs)
- Services failover/HA
- Bandwidth Allocation
- Administrative/Public access to your instances

Each VIP represents a separate network with its own IP block. Every network mode has its own set of changes regarding multinic usage:

Using multinic

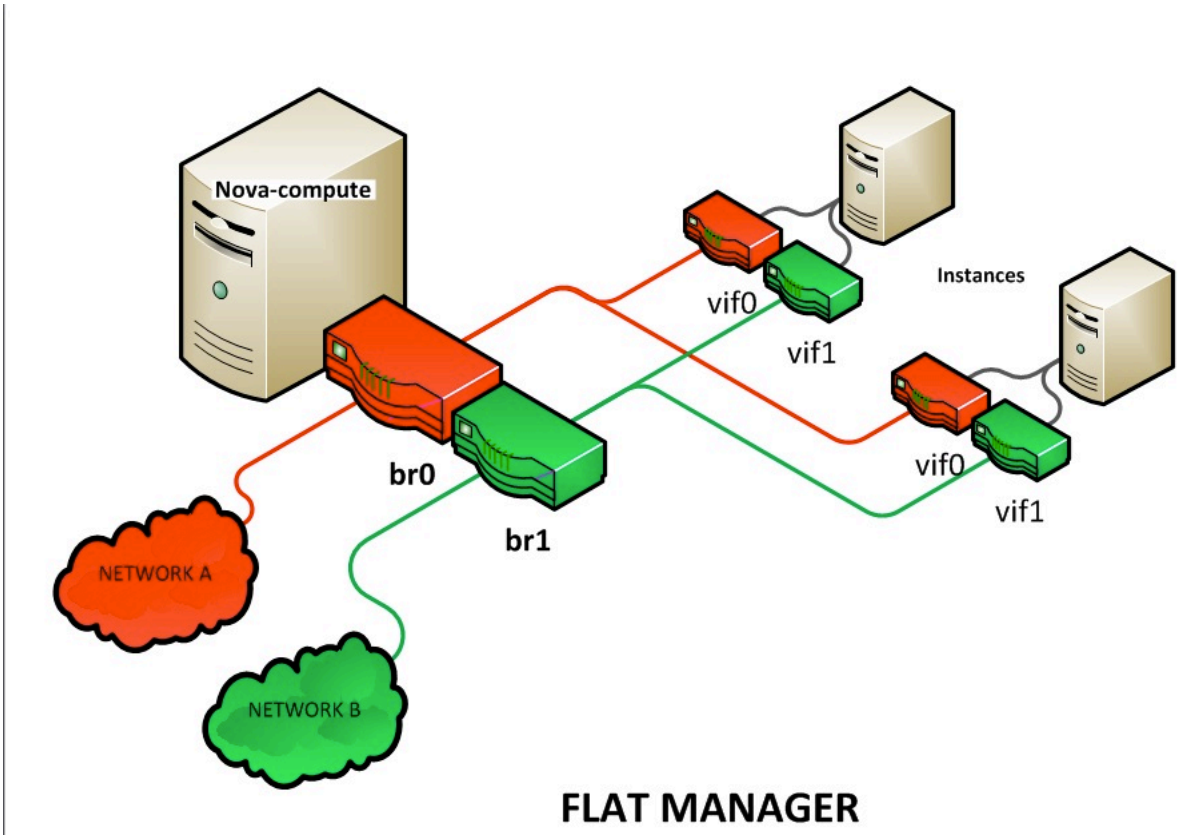
In order to use multinic, create two networks, and attach them to the project (named `project` on the command line):

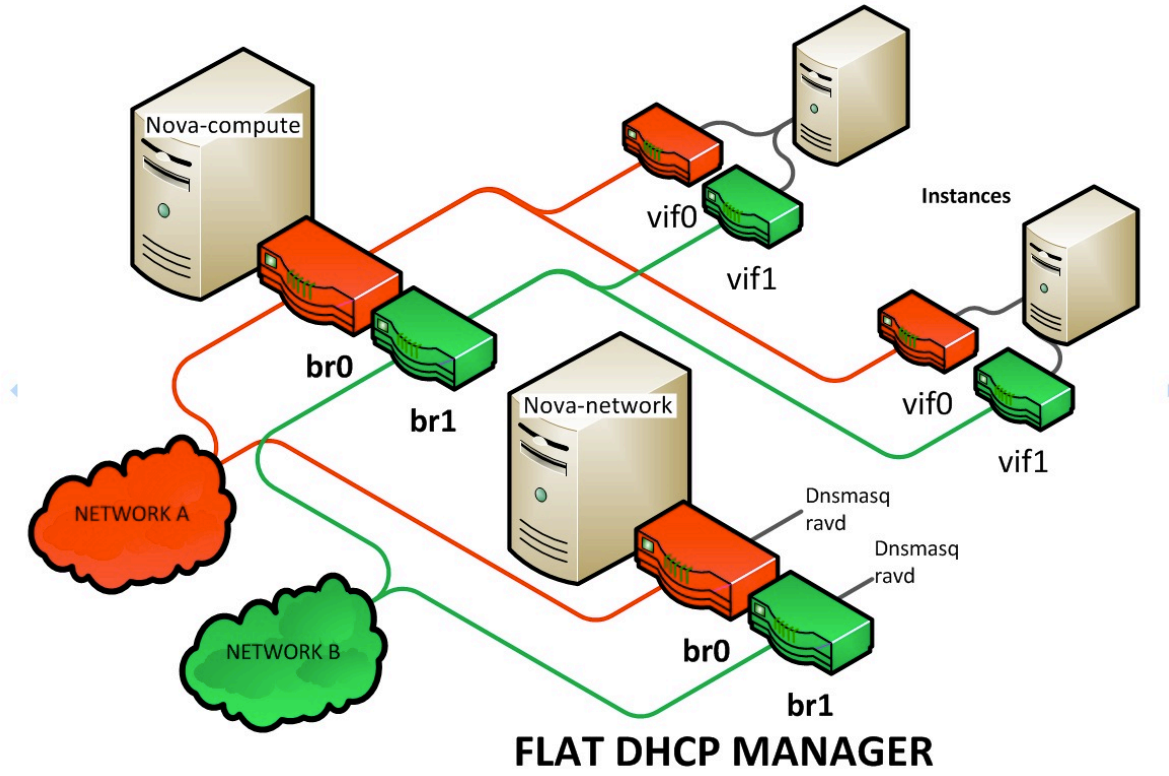
```
$ nova network-create first-net --fixed-range-v4 20.20.0.0/24 --project-id $your-
↪project
$ nova network-create second-net --fixed-range-v4 20.20.10.0/24 --project-id $your-
↪project
```

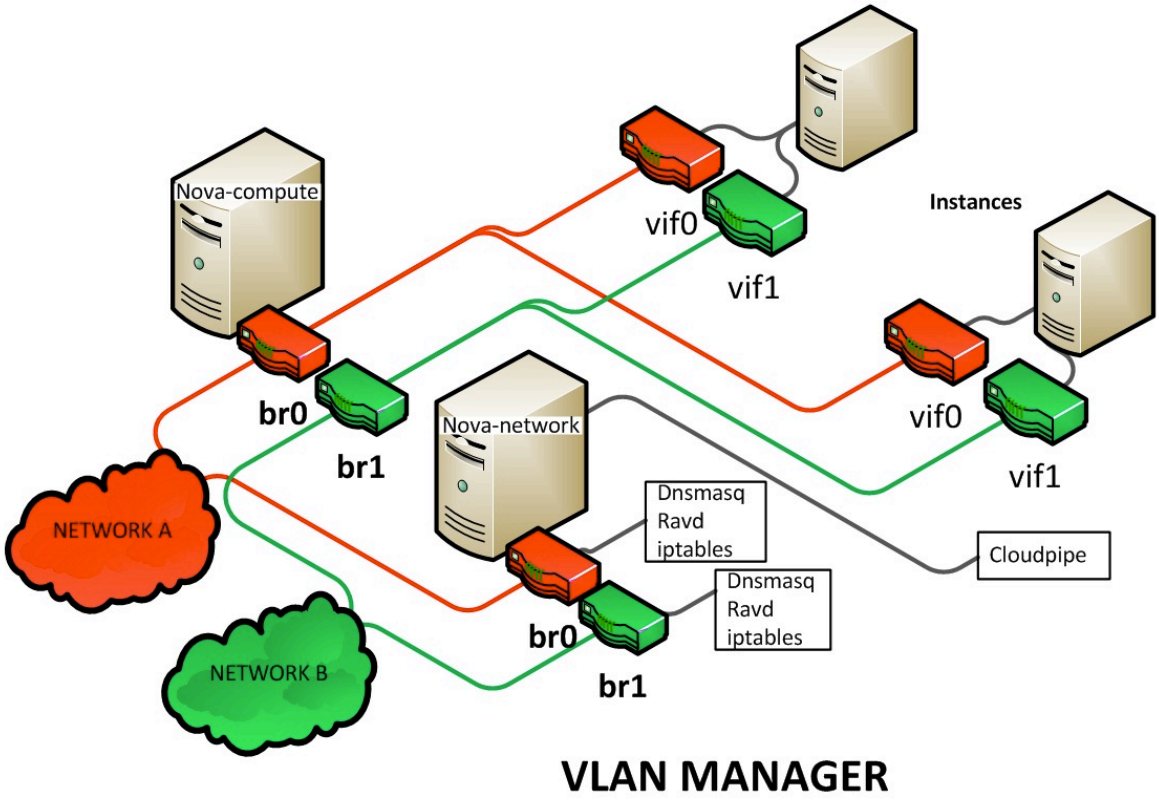
Each new instance will now receive two IP addresses from their respective DHCP servers:

```
$ openstack server list
+---------+----------+--------+-----------------------------------------+------------+
|ID       | Name     | Status | Networks                                | Image Name |
+---------+----------+--------+-----------------------------------------+------------+
| 1234... | MyServer | ACTIVE | network2=20.20.0.3; private=20.20.10.14 | cirros     |
+---------+----------+--------+-----------------------------------------+------------+
```

Note: Make sure you start the second interface on the instance, or it wont be reachable through the second IP.


Nova-compute
vif0
vif1
Instances
br0
br1
vif0
vif1
NETWORK A
NETWORK B
FLAT MANAGER


Nova-compute
vif0
vif1
Instances
br0
br1
Nova-network
vif0
vif1
NETWORK A
Dnsmasq
ravd
Dnsmasq
ravd
NETWORK B
br0
br1
FLAT DHCP MANAGER



This example demonstrates how to set up the interfaces within the instance. This is the configuration that needs to be applied inside the image.

Edit the `/etc/network/interfaces` file:

```
# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet dhcp
```

If the Virtual Network Service Neutron is installed, you can specify the networks to attach to the interfaces by using the `--nic` flag with the **`openstack server create`** command:

```
$ openstack server create --image ed8b2a37-5535-4a5f-a615-443513036d71 \
  --flavor 1 --nic net-id=NETWORK1_ID --nic net-id=NETWORK2_ID test-vm1
```

Troubleshooting Networking

Cannot reach floating IPs

Problem

You cannot reach your instances through the floating IP address.

Solution

- Check that the default security group allows ICMP (ping) and SSH (port 22), so that you can reach the instances:

```
$ openstack security group rule list default
+--------------------------------------+-------------+-----------+----------------
↪-+----------------------+
| ID                                   | IP Protocol | IP Range  | Port Range
↪ | Remote Security Group |
+--------------------------------------+-------------+-----------+----------------
↪-+----------------------+
| 63536865-e5b6-4df1-bac5-ca6d97d8f54d | tcp         | 0.0.0.0/0 | 22:22
↪ | None                 |
| e9d3200f-647a-4293-a9fc-e65ceee189ae | icmp        | 0.0.0.0/0 | type=1:code=-1
↪ | None                 |
+--------------------------------------+-------------+-----------+----------------
↪-+----------------------+
```

- Check the NAT rules have been added to iptables on the node that is running `nova-network`:

```
# iptables -L -nv -t nat \
    -A nova-network-PREROUTING -d 68.99.26.170/32 -j DNAT --to-destination 10.0.0.
↪3 \
    -A nova-network-floating-snat -s 10.0.0.3/32 -j SNAT --to-source 68.99.26.170
```

- Check that the public address (`68.99.26.170` in this example), has been added to your public interface. You should see the address in the listing when you use the **`ip addr`** command:

```
$ ip addr
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
link/ether xx:xx:xx:17:4b:c2 brd ff:ff:ff:ff:ff:ff
inet 13.22.194.80/24 brd 13.22.194.255 scope global eth0
inet 68.99.26.170/32 scope global eth0
inet6 fe80::82b:2bf:fe1:4b2/64 scope link
valid_lft forever preferred_lft forever
```

Note: You cannot use `SSH` to access an instance with a public IP from within the same server because the routing configuration does not allow it.

- Use `tcpdump` to identify if packets are being routed to the inbound interface on the compute host. If the packets are reaching the compute hosts but the connection is failing, the issue may be that the packet is being dropped by reverse path filtering. Try disabling reverse-path filtering on the inbound interface. For example, if the inbound interface is `eth2`, run:

```
# sysctl -w net.ipv4.conf.ETH2.rp_filter=0
```

If this solves the problem, add the following line to `/etc/sysctl.conf` so that the reverse-path filter is persistent:

```
net.ipv4.conf.rp_filter=0
```

Temporarily disable firewall

Problem

Networking issues prevent administrators accessing or reaching VMs through various pathways.

Solution

You can disable the firewall by setting this option in `/etc/nova/nova.conf`:

```
firewall_driver=nova.virt.firewall.NoopFirewallDriver
```

Warning: We strongly recommend you remove this line to re-enable the firewall once your networking issues have been resolved.

Packet loss from instances to nova-network server (VLANManager mode)

Problem

If you can access your instances with `SSH` but the network to your instance is slow, or if you find that running certain operations are slower than they should be (for example, `sudo`), packet loss could be occurring on the connection to the instance.

Packet loss can be caused by Linux networking configuration settings related to bridges. Certain settings can cause packets to be dropped between the VLAN interface (for example, `vlan100`) and the associated bridge interface (for example, `br100`) on the host running `nova-network`.

Solution

One way to check whether this is the problem is to open three terminals and run the following commands:

1. In the first terminal, on the host running `nova-network`, use `tcpdump` on the VLAN interface to monitor DNS-related traffic (UDP, port 53). As root, run:

   ```
   # tcpdump -K -p -i vlan100 -v -vv udp port 53
   ```

2. In the second terminal, also on the host running `nova-network`, use `tcpdump` to monitor DNS-related traffic on the bridge interface. As root, run:

   ```
   # tcpdump -K -p -i br100 -v -vv udp port 53
   ```

3. In the third terminal, use `SSH` to access the instance and generate DNS requests by using the **`nslookup`** command:

   ```
   $ nslookup www.google.com
   ```

The symptoms may be intermittent, so try running **nslookup** multiple times. If the network configuration is correct, the command should return immediately each time. If it is not correct, the command hangs for several seconds before returning.

4. If the **nslookup** command sometimes hangs, and there are packets that appear in the first terminal but not the second, then the problem may be due to filtering done on the bridges. Try disabling filtering, and running these commands as root:

```
# sysctl -w net.bridge.bridge-nf-call-arptables=0
# sysctl -w net.bridge.bridge-nf-call-iptables=0
# sysctl -w net.bridge.bridge-nf-call-ip6tables=0
```

If this solves your issue, add the following line to `/etc/sysctl.conf` so that the changes are persistent:

```
net.bridge.bridge-nf-call-arptables=0
net.bridge.bridge-nf-call-iptables=0
net.bridge.bridge-nf-call-ip6tables=0
```

KVM: Network connectivity works initially, then fails

Problem

With KVM hypervisors, instances running Ubuntu 12.04 sometimes lose network connectivity after functioning properly for a period of time.

Solution

Try loading the `vhost_net` kernel module as a workaround for this issue (see bug #997978) . This kernel module may also improve network performance on KVM. To load the kernel module:

```
# modprobe vhost_net
```

Note: Loading the module has no effect on running instances.

4.3.22 Networking with neutron

While nova uses the OpenStack Networking service (neutron) to provide network connectivity for instances, nova itself provides some additional features not possible with neutron alone. These are described below.

SR-IOV

Changed in version 2014.2: The feature described below was first introduced in the Juno release.

The SR-IOV specification defines a standardized mechanism to virtualize PCIe devices. This mechanism can virtualize a single PCIe Ethernet controller to appear as multiple PCIe devices. Each device can be directly assigned to an instance, bypassing the hypervisor and virtual switch layer. As a result, users are able to achieve low latency and near-line wire speed.

A full guide on configuring and using SR-IOV is provided in the OpenStack Networking service documentation

NUMA Affinity

New in version 18.0.0: The feature described below was first introduced in the Rocky release.

Important: The functionality described below is currently only supported by the libvirt/KVM driver.

As described in *CPU topologies*, NUMA is a computer architecture where memory accesses to certain regions of system memory can have higher latencies than other regions, depending on the CPU(s) your process is running on. This effect extends to devices connected to the PCIe bus, a concept known as NUMA I/O. Many Network Interface Cards (NICs) connect using the PCIe interface, meaning they are susceptible to the ill-effects of poor NUMA affinitization. As a result, NUMA locality must be considered when creating an instance where high dataplane performance is a requirement.

Fortunately, nova provides functionality to ensure NUMA affinitization is provided for instances using neutron. How this works depends on the type of port you are trying to use.

Todo: Add documentation for PCI NUMA affinity and PCI policies and link to it from here.

For SR-IOV ports, virtual functions, which are PCI devices, are attached to the instance. This means the instance can benefit from the NUMA affinity guarantees provided for PCI devices. This happens automatically.

For all other types of ports, some manual configuration is required.

1. Identify the type of network(s) you wish to provide NUMA affinity for.

   - If a network is an L2-type network (`provider:network_type` of `flat` or `vlan`), affinity of the network to given NUMA node(s) can vary depending on value of the `provider:physical_network` attribute of the network, commonly referred to as the *physnet* of the network. This is because most neutron drivers map each *physnet* to a different bridge, to which multiple NICs are attached, or to a different (logical) NIC.
   - If a network is an L3-type networks (`provider:network_type` of `vxlan`, `gre` or `geneve`), all traffic will use the device to which the *endpoint IP* is assigned. This means all L3 networks on a given host will have affinity to the same NUMA node(s). Refer to the neutron documentation for more information.

2. Determine the NUMA affinity of the NICs attached to the given network(s).

   How this should be achieved varies depending on the switching solution used and whether the network is a L2-type network or an L3-type networks.

   Consider an L2-type network using the Linux Bridge mechanism driver. As noted in the neutron documentation, *physnets* are mapped to interfaces using the `[linux_bridge] physical_interface_mappings` configuration option. For example:

   ```
   [linux_bridge]
   physical_interface_mappings = provider:PROVIDER_INTERFACE
   ```

   Once you have the device name, you can query *sysfs* to retrieve the NUMA affinity for this device. For example:

   ```
   $ cat /sys/class/net/PROVIDER_INTERFACE/device/numa_node
   ```

   For an L3-type network using the Linux Bridge mechanism driver, the device used will be configured using protocol-specific endpoint IP configuration option. For VXLAN, this is the `[vxlan] local_ip` option. For example:

```
[vxlan]
local_ip = OVERLAY_INTERFACE_IP_ADDRESS
```

Once you have the IP address in question, you can use **ip** to identify the device that has been assigned this IP address and from there can query the NUMA affinity using *sysfs* as above.

Note: The example provided above is merely that: an example. How one should identify this information can vary massively depending on the driver used, whether bonding is used, the type of network used, etc.

---

3. Configure NUMA affinity in `nova.conf`.

   Once you have identified the NUMA affinity of the devices used for your networks, you need to configure this in `nova.conf`. As before, how this should be achieved varies depending on the type of network.

   For L2-type networks, NUMA affinity is defined based on the `provider:physical_network` attribute of the network. There are two configuration options that must be set:

   `[neutron] physnets` This should be set to the list of physnets for which you wish to provide NUMA affinity. Refer to the *`documentation`* for more information.

   `[neutron_physnet_{physnet}] numa_nodes` This should be set to the list of NUMA node(s) that networks with the given `{physnet}` should be affinitized to.

   For L3-type networks, NUMA affinity is defined globally for all tunneled networks on a given host. There is only one configuration option that must be set:

   `[neutron_tunneled] numa_nodes` This should be set to a list of one or NUMA nodes to which instances using tunneled networks will be affinitized.

4. Configure a NUMA topology for instance flavor(s)

   For network NUMA affinity to have any effect, the instance must have a NUMA topology itself. This can be configured explicitly, using the `hw:numa_nodes` extra spec, or implicitly through the use of CPU pinning (`hw:cpu_policy=dedicated`) or PCI devices. For more information, refer to *CPU topologies*.

Examples

Take an example for deployment using L2-type networks first.

```
[neutron]
physnets = foo,bar

[neutron_physnet_foo]
numa_nodes = 0

[neutron_physnet_bar]
numa_nodes = 2, 3
```

This configuration will ensure instances using one or more L2-type networks with `provider:physical_network=foo` must be scheduled on host cores from NUMA nodes 0, while instances using one or more networks with `provider:physical_network=bar` must be scheduled on host cores from both NUMA nodes 2 and 3. For the latter case, it will be necessary to split the guest across two or more host NUMA nodes using the `hw:numa_nodes` *flavor extra spec*.

Now, take an example for a deployment using L3 networks.

```
[neutron_tunneled]
numa_nodes = 0
```

This is much simpler as all tunneled traffic uses the same logical interface. As with the L2-type networks, this configuration will ensure instances using one or more L3-type networks must be scheduled on host cores from NUMA node 0. It is also possible to define more than one NUMA node, in which case the instance must be split across these nodes.

4.3.23 Recover from a failed compute node

If you deploy Compute with a shared file system, you can use several methods to quickly recover from a node failure. This section discusses manual recovery.

Evacuate instances

If a hardware malfunction or other error causes the cloud compute node to fail, you can use the **nova evacuate** command to evacuate instances. See *evacuate instances* for more information on using the command.

Manual recovery

To manually recover a failed compute node:

1. Identify the VMs on the affected hosts by using a combination of the **openstack server list** and **openstack server show** commands.
2. Query the Compute database for the status of the host. This example converts an EC2 API instance ID to an OpenStack ID. If you use the **nova** commands, you can substitute the ID directly. This example output is truncated:

```
mysql> SELECT * FROM instances WHERE id = CONV('15b9', 16, 10) \G;
*************************** 1. row ***************************
created_at: 2012-06-19 00:48:11
updated_at: 2012-07-03 00:35:11
deleted_at: NULL
...
id: 5561
...
power_state: 5
vm_state: shutoff
...
hostname: at3-ui02
host: np-rcc54
...
uuid: 3f57699a-e773-4650-a443-b4b37eed5a06
...
task_state: NULL
...
```

Note: Find the credentials for your database in `/etc/nova.conf` file.

3. Decide to which compute host to move the affected VM. Run this database command to move the VM to that host:

```
mysql> UPDATE instances SET host = 'np-rcc46' WHERE uuid = '3f57699a-e773-4650-
↪a443-b4b37eed5a06';
```

4. If you use a hypervisor that relies on libvirt, such as KVM, update the `libvirt.xml` file in `/var/lib/nova/instances/[instance ID]` with these changes:
   - Change the `DHCPSERVER` value to the host IP address of the new compute host.
   - Update the VNC IP to `0.0.0.0`.
5. Reboot the VM:

```
$ openstack server reboot 3f57699a-e773-4650-a443-b4b37eed5a06
```

Typically, the database update and **`openstack server reboot`** command recover a VM from a failed host. However, if problems persist, try one of these actions:

- Use **`virsh`** to recreate the network filter configuration.
- Restart Compute services.
- Update the `vm_state` and `power_state` fields in the Compute database.

Recover from a UID/GID mismatch

Sometimes when you run Compute with a shared file system or an automated configuration tool, files on your compute node might use the wrong UID or GID. This UID or GID mismatch can prevent you from running live migrations or starting virtual machines.

This procedure runs on `nova-compute` hosts, based on the KVM hypervisor:

1. Set the nova UID to the same number in `/etc/passwd` on all hosts. For example, set the UID to `112`.

   Note: Choose UIDs or GIDs that are not in use for other users or groups.

2. Set the `libvirt-qemu` UID to the same number in the `/etc/passwd` file on all hosts. For example, set the UID to `119`.
3. Set the `nova` group to the same number in the `/etc/group` file on all hosts. For example, set the group to `120`.
4. Set the `libvirtd` group to the same number in the `/etc/group` file on all hosts. For example, set the group to `119`.
5. Stop the services on the compute node.
6. Change all files that the nova user or group owns. For example:

```
# find / -uid 108 -exec chown nova {} \;
# note the 108 here is the old nova UID before the change
# find / -gid 120 -exec chgrp nova {} \;
```

7. Repeat all steps for the `libvirt-qemu` files, if required.
8. Restart the services.
9. To verify that all files use the correct IDs, run the **`find`** command.

Recover cloud after disaster

This section describes how to manage your cloud after a disaster and back up persistent storage volumes. Backups are mandatory, even outside of disaster scenarios.

For a definition of a disaster recovery plan (DRP), see https://en.wikipedia.org/wiki/Disaster_Recovery_Plan.

A disk crash, network loss, or power failure can affect several components in your cloud architecture. The worst disaster for a cloud is a power loss. A power loss affects these components:

- A cloud controller (`nova-api`, `nova-objectstore`, `nova-network`)
- A compute node (`nova-compute`)
- A storage area network (SAN) used by OpenStack Block Storage (`cinder-volumes`)

Before a power loss:

- Create an active iSCSI session from the SAN to the cloud controller (used for the `cinder-volumes` LVMs VG).
- Create an active iSCSI session from the cloud controller to the compute node (managed by `cinder-volume`).
- Create an iSCSI session for every volume (so 14 EBS volumes requires 14 iSCSI sessions).
- Create `iptables` or `ebtables` rules from the cloud controller to the compute node. This allows access from the cloud controller to the running instance.
- Save the current state of the database, the current state of the running instances, and the attached volumes (mount point, volume ID, volume status, etc), at least from the cloud controller to the compute node.

After power resumes and all hardware components restart:

- The iSCSI session from the SAN to the cloud no longer exists.
- The iSCSI session from the cloud controller to the compute node no longer exists.
- nova-network reapplies configurations on boot and, as a result, recreates the iptables and ebtables from the cloud controller to the compute node.
- Instances stop running.

  Instances are not lost because neither `destroy` nor `terminate` ran. The files for the instances remain on the compute node.
- The database does not update.

Begin recovery

> **Warning:** Do not add any steps or change the order of steps in this procedure.

1. Check the current relationship between the volume and its instance, so that you can recreate the attachment.

   Use the **`openstack volume list`** command to get this information. Note that the **`openstack`** client can get volume information from OpenStack Block Storage.
2. Update the database to clean the stalled state. Do this for every volume by using these queries:

```
mysql> use cinder;
mysql> update volumes set mountpoint=NULL;
mysql> update volumes set status="available" where status <>"error_deleting";
mysql> update volumes set attach_status="detached";
mysql> update volumes set instance_id=0;
```

Use **openstack volume list** command to list all volumes.

3. Restart the instances by using the **openstack server reboot INSTANCE** command.

Important: Some instances completely reboot and become reachable, while some might stop at the plymouth stage. This is expected behavior. DO NOT reboot a second time.

Instance state at this stage depends on whether you added an */etc/fstab* entry for that volume. Images built with the cloud-init package remain in a `pending` state, while others skip the missing volume and start. You perform this step to ask Compute to reboot every instance so that the stored state is preserved. It does not matter if not all instances come up successfully. For more information about cloud-init, see help.ubuntu.com/community/CloudInit/.

---

4. If required, run the **openstack server add volume** command to reattach the volumes to their respective instances. This example uses a file of listed volumes to reattach them:

```
#!/bin/bash

while read line; do
    volume=`echo $line | $CUT -f 1 -d " "`
    instance=`echo $line | $CUT -f 2 -d " "`
    mount_point=`echo $line | $CUT -f 3 -d " "`
        echo "ATTACHING VOLUME FOR INSTANCE - $instance"
    openstack server add volume $instance $volume $mount_point
    sleep 2
done < $volumes_tmp_file
```

Instances that were stopped at the plymouth stage now automatically continue booting and start normally. Instances that previously started successfully can now see the volume.

5. Log in to the instances with SSH and reboot them.

If some services depend on the volume or if a volume has an entry in fstab, you can now restart the instance. Restart directly from the instance itself and not through **nova**:

```
# shutdown -r now
```

When you plan for and complete a disaster recovery, follow these tips:

- Use the `errors=remount` option in the `fstab` file to prevent data corruption.

  In the event of an I/O error, this option prevents writes to the disk. Add this configuration option into the cinder-volume server that performs the iSCSI connection to the SAN and into the instances `fstab` files.

- Do not add the entry for the SANs disks to the cinder-volumes `fstab` file.

  Some systems hang on that step, which means you could lose access to your cloud-controller. To re-run the session manually, run this command before performing the mount:

```
# iscsiadm -m discovery -t st -p $SAN_IP $ iscsiadm -m node --target-name
↪$IQN -p $SAN_IP -l
```

- On your instances, if you have the whole `/home/` directory on the disk, leave a users directory with the users bash files and the `authorized_keys` file instead of emptying the `/home/` directory and mapping the disk on it.

  This action enables you to connect to the instance without the volume attached, if you allow only connections through public keys.

To reproduce the power loss, connect to the compute node that runs that instance and close the iSCSI session. Do not detach the volume by using the **`openstack server remove volume`** command. You must manually close the iSCSI session. This example closes an iSCSI session with the number `15`:

```
# iscsiadm -m session -u -r 15
```

Do not forget the `-r` option. Otherwise, all sessions close.

Warning: There is potential for data loss while running instances during this procedure. If you are using Liberty or earlier, ensure you have the correct patch and set the options appropriately.

4.3.24 Manage quotas

Todo: Merge this into quotas.rst

To prevent system capacities from being exhausted without notification, you can set up quotas. Quotas are operational limits. For example, the number of gigabytes allowed for each project can be controlled so that cloud resources are optimized. Quotas can be enforced at both the project and the project-user level.

Using the command-line interface, you can manage quotas for the OpenStack Compute service, the OpenStack Block Storage service, and the OpenStack Networking service.

The cloud operator typically changes default values because a project requires more than ten volumes or 1 TB on a compute node.

Note: To view all projects, run:

```
$ openstack project list
+----------------------------------+----------+
| ID                               | Name     |
+----------------------------------+----------+
| e66d97ac1b704897853412fc8450f7b9 | admin    |
| bf4a37b885fe46bd86e999e50adad1d3 | services |
| 21bd1c7c95234fd28f589b60903606fa | tenant01 |
| f599c5cd1cba4125ae3d7caed08e288c | tenant02 |
+----------------------------------+----------+
```

To display all current users for a project, run:

```
$ openstack user list --project PROJECT_NAME
+----------------------------------+--------+
| ID                               | Name   |
+----------------------------------+--------+
| ea30aa434ab24a139b0e85125ec8a217 | demo00 |
| 4f8113c1d838467cad0c2f337b3dfded | demo01 |
+----------------------------------+--------+
```

Use `openstack quota show PROJECT_NAME` to list all quotas for a project.

Use `openstack quota set PROJECT_NAME -parameters` to set quota values.

4.3.25 Manage Compute service quotas

As an administrative user, you can use the **`nova quota-*`** commands, which are provided by the `python-novaclient` package, to update the Compute service quotas for a specific project or project user, as well as update the quota defaults for a new project.

Compute quota descriptions

Quota name	Description
cores	Number of instance cores (VCPUs) allowed per project.
instances	Number of instances allowed per project.
key_pairs	Number of key pairs allowed per user.
metadata_items	Number of metadata items allowed per instance.
ram	Megabytes of instance ram allowed per project.
server_groups	Number of server groups per project.
server_group_members	Number of servers per server group.

The following quotas were previously available but were removed in microversion 2.36 as they proxied information available from the networking service.

Quota name	Description
fixed_ips	Number of fixed IP addresses allowed per project. This number must be equal to or greater than the number of allowed instances.
floating_ips	Number of floating IP addresses allowed per project.
networks	Number of networks allowed per project (nova-network only).
security_groups	Number of security groups per project.
security_group_rules	Number of security group rules per project.

Similarly, the following quotas were previously available but were removed in microversion 2.57 as the personality files feature was deprecated.

Quota name	Description
injected_files	Number of injected files allowed per project.
injected_file_content_bytes	Number of content bytes allowed per injected file.
injected_file_path_bytes	Length of injected file path.

View and update Compute quotas for a project

To view and update default quota values

1. List all default quotas for all projects:

```
$ openstack quota show --default
+----------------------+----------+
| Field                | Value    |
+----------------------+----------+
| backup-gigabytes     | 1000     |
| backups              | 10       |
| cores                | 20       |
| fixed-ips            | -1       |
| floating-ips         | 50       |
| gigabytes            | 1000     |
| health_monitors      | None     |
| injected-file-size   | 10240    |
| injected-files       | 5        |
| injected-path-size   | 255      |
| instances            | 10       |
| key-pairs            | 100      |
| l7_policies          | None     |
| listeners            | None     |
| load_balancers       | None     |
| location             | None     |
| name                 | None     |
| networks             | 10       |
| per-volume-gigabytes | -1       |
| pools                | None     |
| ports                | 50       |
| project              | None     |
| project_name         | project  |
| properties           | 128      |
| ram                  | 51200    |
| rbac_policies        | 10       |
| routers              | 10       |
| secgroup-rules       | 100      |
| secgroups            | 10       |
| server-group-members | 10       |
| server-groups        | 10       |
| snapshots            | 10       |
| subnet_pools         | -1       |
| subnets              | 10       |
| volumes              | 10       |
+----------------------+----------+
```

Note: This lists default quotas for all services and not just nova.

2. Update a default value for a new project, for example:

```
$ openstack quota set --instances 15 --class default
```

To view quota values for an existing project

1. List the currently set quota values for a project:

```
$ openstack quota show PROJECT_NAME
+----------------------+----------------------------------+
| Field                | Value                            |
```

(continues on next page)

(continued from previous page)

```
+----------------------+----------------------------------+
| backup-gigabytes     | 1000                             |
| backups              | 10                               |
| cores                | 32                               |
| fixed-ips            | -1                               |
| floating-ips         | 10                               |
| gigabytes            | 1000                             |
| health_monitors      | None                             |
| injected-file-size   | 10240                            |
| injected-files       | 5                                |
| injected-path-size   | 255                              |
| instances            | 10                               |
| key-pairs            | 100                              |
| l7_policies          | None                             |
| listeners            | None                             |
| load_balancers       | None                             |
| location             | None                             |
| name                 | None                             |
| networks             | 20                               |
| per-volume-gigabytes | -1                               |
| pools                | None                             |
| ports                | 60                               |
| project              | c8156b55ec3b486193e73d2974196993 |
| project_name         | project                          |
| properties           | 128                              |
| ram                  | 65536                            |
| rbac_policies        | 10                               |
| routers              | 10                               |
| secgroup-rules       | 50                               |
| secgroups            | 50                               |
| server-group-members | 10                               |
| server-groups        | 10                               |
| snapshots            | 10                               |
| subnet_pools         | -1                               |
| subnets              | 20                               |
| volumes              | 10                               |
+----------------------+----------------------------------+
```

Note: This lists quotas for all services and not just nova.

To update quota values for an existing project

1. Obtain the project ID.

```
$ project=$(openstack project show -f value -c id PROJECT_NAME)
```

2. Update a particular quota value.

   To update quotas for a project:

```
$ openstack quota set --QUOTA_NAME QUOTA_VALUE PROJECT_NAME
```

   To update quotas for a class:

```
$ openstack quota set --class --QUOTA_NAME QUOTA_VALUE CLASS_NAME
```

Note: Only the `default` class is supported by nova.

For example:

```
$ openstack quota set --instances 50 PROJECT_NAME
$ openstack quota show PROJECT_NAME
+----------------------+----------------------------------+
| Field                | Value                            |
+----------------------+----------------------------------+
| ...                  | ...                              |
| instances            | 50                               |
| ...                  | ...                              |
+----------------------+----------------------------------+
```

Note: To view a list of options for the **`openstack quota set`** command, run:

```
$ openstack help quota set
```

View and update Compute quotas for a project user

To view quota values for a project user

1. Place the user ID in a usable variable.

```
$ projectUser=$(openstack user show -f value -c id USER_NAME)
```

2. Place the users project ID in a usable variable, as follows:

```
$ project=$(openstack project show -f value -c id PROJECT_NAME)
```

3. List the currently set quota values for a project user.

```
$ nova quota-show --user $projectUser --tenant $project
```

For example:

```
$ nova quota-show --user $projectUser --tenant $project
+-----------------------------+-------+
| Quota                       | Limit |
+-----------------------------+-------+
| instances                   | 10    |
| cores                       | 20    |
| ram                         | 51200 |
| floating_ips                | 20    |
| fixed_ips                   | -1    |
| metadata_items              | 128   |
| injected_files              | 5     |
| injected_file_content_bytes | 10240 |
| injected_file_path_bytes    | 255   |
```

(continues on next page)

(continued from previous page)

```
| key_pairs                    | 100   |
| security_groups              | 10    |
| security_group_rules         | 20    |
| server_groups                | 10    |
| server_group_members         | 10    |
+------------------------------+-------+
```

To update quota values for a project user

1. Place the user ID in a usable variable.

```
$ projectUser=$(openstack user show -f value -c id USER_NAME)
```

2. Place the users project ID in a usable variable, as follows:

```
$ project=$(openstack project show -f value -c id PROJECT_NAME)
```

3. Update a particular quota value, as follows:

```
$ nova quota-update  --user $projectUser --QUOTA_NAME QUOTA_VALUE $project
```

For example:

```
$ nova quota-update --user $projectUser --floating-ips 12 $project
$ nova quota-show --user $projectUser --tenant $project
+------------------------------+-------+
| Quota                        | Limit |
+------------------------------+-------+
| instances                    | 10    |
| cores                        | 20    |
| ram                          | 51200 |
| floating_ips                 | 12    |
| fixed_ips                    | -1    |
| metadata_items               | 128   |
| injected_files               | 5     |
| injected_file_content_bytes  | 10240 |
| injected_file_path_bytes     | 255   |
| key_pairs                    | 100   |
| security_groups              | 10    |
| security_group_rules         | 20    |
| server_groups                | 10    |
| server_group_members         | 10    |
+------------------------------+-------+
```

Note: To view a list of options for the **`nova quota-update`** command, run:

```
$ nova help quota-update
```

To display the current quota usage for a project user

Use **`nova limits`** to get a list of the current quota values and the current quota usage:

```
$ nova limits --tenant PROJECT_NAME

+------+-----+-------+--------+------+----------------+
| Verb | URI | Value | Remain | Unit | Next_Available |
+------+-----+-------+--------+------+----------------+
+------+-----+-------+--------+------+----------------+

+--------------------+------+-------+
| Name               | Used | Max   |
+--------------------+------+-------+
| Cores              | 0    | 20    |
| Instances          | 0    | 10    |
| Keypairs           | -    | 100   |
| Personality        | -    | 5     |
| Personality Size   | -    | 10240 |
| RAM                | 0    | 51200 |
| Server Meta        | -    | 128   |
| ServerGroupMembers | -    | 10    |
| ServerGroups       | 0    | 10    |
+--------------------+------+-------+
```

Note: The **`nova limits`** command generates an empty table as a result of the Compute API, which prints an empty list for backward compatibility purposes.

4.3.26 Configure remote console access

OpenStack provides a number of different methods to interact with your guests: VNC, SPICE, Serial, RDP or MKS. If configured, these can be accessed by users through the OpenStack dashboard or the command line. This document outlines how these different technologies can be configured.

Overview

It is considered best practice to deploy only one of the consoles types and not all console types are supported by all compute drivers. Regardless of what option is chosen, a console proxy service is required. These proxy services are responsible for the following:

- Provide a bridge between the public network where the clients live and the private network where the servers with consoles live.
- Mediate token authentication.
- Transparently handle hypervisor-specific connection details to provide a uniform client experience.

For some combinations of compute driver and console driver, these proxy services are provided by the hypervisor or another service. For all others, nova provides services to handle this proxying. Consider a noVNC-based VNC console connection for example:

1. A user connects to the API and gets an `access_url` such as, `http://ip:port/?path=%3Ftoken%3Dxyz`.
2. The user pastes the URL in a browser or uses it as a client parameter.
3. The browser or client connects to the proxy.

4. The proxy authorizes the token for the user, and maps the token to the *private* host and port of the VNC server for an instance.

   The compute host specifies the address that the proxy should use to connect through the *vnc.server_proxyclient_address* option. In this way, the VNC proxy works as a bridge between the public network and private host network.

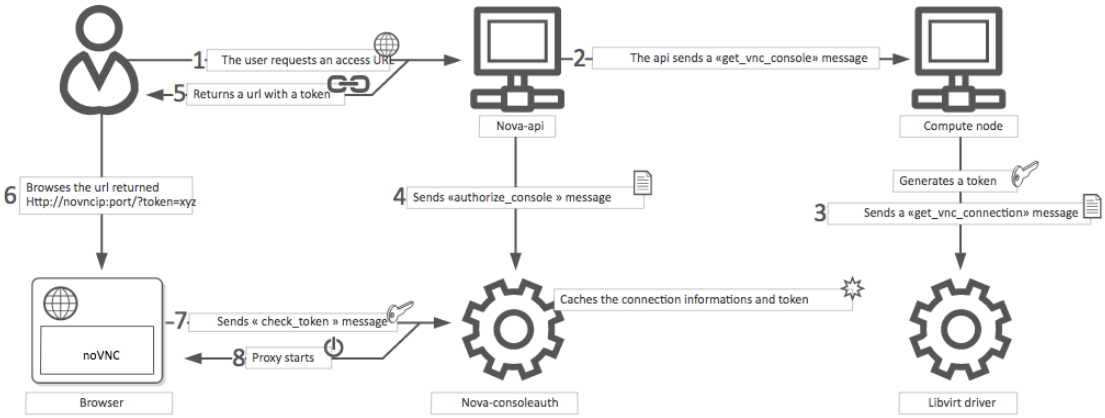
5. The proxy initiates the connection to VNC server and continues to proxy until the session ends.

This means a typical deployment with noVNC-based VNC consoles will have the following components:

- One or more **nova-novncproxy** service. Supports browser-based noVNC clients. For simple deployments, this service typically runs on the same machine as **nova-api** because it operates as a proxy between the public network and the private compute host network.
- One or more **nova-compute** services. Hosts the instances for which consoles are provided.

Todo: The below diagram references **nova-consoleauth** and needs to be updated.

This particular example is illustrated below.



noVNC-based VNC console

VNC is a graphical console with wide support among many hypervisors and clients. noVNC provides VNC support through a web browser.

Note: It has been reported that versions of noVNC older than 0.6 do not work with the **nova-novncproxy** service.

If using non-US key mappings, you need at least noVNC 1.0.0 for a fix.

If using VMware ESX/ESXi hypervisors, you need at least noVNC 1.1.0 for a fix.

Configuration

To enable the noVNC VNC console service, you must configure both the **nova-novncproxy** service and the **nova-compute** service. Most options are defined in the *vnc* group.

The **nova-novncproxy** service accepts the following options:

- *daemon*
- *ssl_only*
- *source_is_ipv6*
- *cert*
- *key*
- *web*
- *vnc.novncproxy_host*
- *vnc.novncproxy_port*

If using the libvirt compute driver and enabling *VNC proxy security*, the following additional options are supported:

- *vnc.auth_schemes*
- *vnc.vencrypt_client_key*
- *vnc.vencrypt_client_cert*
- *vnc.vencrypt_ca_certs*

For example, to configure this via a `nova-novncproxy.conf` file:

```
[vnc]
novncproxy_host = 0.0.0.0
novncproxy_port = 6082
```

Note: This doesnt show configuration with security. For information on how to configure this, refer to *VNC proxy security* below.

The **`nova-compute`** service requires the following options to configure noVNC-based VNC console support:

- *vnc.enabled*
- *vnc.novncproxy_base_url*
- *vnc.server_listen*
- *vnc.server_proxyclient_address*
- *vnc.keymap*

If using the VMware compute driver, the following additional options are supported:

- *vmware.vnc_port*
- *vmware.vnc_port_total*

For example, to configure this via a `nova.conf` file:

```
[vnc]
enabled = True
novncproxy_base_url = http://IP_ADDRESS:6082/vnc_auto.html
server_listen = 127.0.0.1
server_proxyclient_address = 127.0.0.1
keymap = en-us
```

Replace `IP_ADDRESS` with the IP address from which the proxy is accessible by the outside world. For example, this may be the management interface IP address of the controller or the VIP.

VNC proxy security

Deploy the public-facing interface of the VNC proxy with HTTPS to prevent attacks from malicious parties on the network between the tenant user and proxy server. When using HTTPS, the TLS encryption only applies to data between the tenant user and proxy server. The data between the proxy server and Compute node instance will still be unencrypted. To provide protection for the latter, it is necessary to enable the VeNCrypt authentication scheme for VNC in both the Compute nodes and noVNC proxy server hosts.

QEMU/KVM Compute node configuration

Ensure each Compute node running QEMU/KVM with libvirt has a set of certificates issued to it. The following is a list of the required certificates:

- `/etc/pki/libvirt-vnc/server-cert.pem`

  An x509 certificate to be presented **by the VNC server**. The `CommonName` should match the **primary hostname of the compute node**. Use of `subjectAltName` is also permitted if there is a need to use multiple hostnames or IP addresses to access the same Compute node.

- `/etc/pki/libvirt-vnc/server-key.pem`

  The private key used to generate the `server-cert.pem` file.

- `/etc/pki/libvirt-vnc/ca-cert.pem`

  The authority certificate used to sign `server-cert.pem` and sign the VNC proxy server certificates.

The certificates must have v3 basic constraints[2] present to indicate the permitted key use and purpose data.

We recommend using a dedicated certificate authority solely for the VNC service. This authority may be a child of the master certificate authority used for the OpenStack deployment. This is because libvirt does not currently have a mechanism to restrict what certificates can be presented by the proxy server.

For further details on certificate creation, consult the QEMU manual page documentation on VNC server certificate setup[1].

Configure libvirt to enable the VeNCrypt authentication scheme for the VNC server. In `/etc/libvirt/qemu.conf`, uncomment the following settings:

- `vnc_tls=1`

  This instructs libvirt to enable the VeNCrypt authentication scheme when launching QEMU, passing it the certificates shown above.

- `vnc_tls_x509_verify=1`

  This instructs QEMU to require that all VNC clients present a valid x509 certificate. Assuming a dedicated certificate authority is used for the VNC service, this ensures that only approved VNC proxy servers can connect to the Compute nodes.

After editing `qemu.conf`, the `libvirtd` service must be restarted:

```
$ systemctl restart libvirtd.service
```

Changes will not apply to any existing running guests on the Compute node, so this configuration should be done before launching any instances.

[2] https://tools.ietf.org/html/rfc3280#section-4.2.1.10
[1] https://qemu.weilnetz.de/doc/qemu-doc.html#vnc_005fsec_005fcertificate_005fverify

noVNC proxy server configuration

The noVNC proxy server initially only supports the `none` authentication scheme, which does no checking. Therefore, it is necessary to enable the `vencrypt` authentication scheme by editing the `nova.conf` file to set.

```
[vnc]
auth_schemes=vencrypt,none
```

The *`vnc.auth_schemes`* values should be listed in order of preference. If enabling VeNCrypt on an existing deployment which already has instances running, the noVNC proxy server must initially be allowed to use `vencrypt` and `none`. Once it is confirmed that all Compute nodes have VeNCrypt enabled for VNC, it is possible to remove the `none` option from the list of the *`vnc.auth_schemes`* values.

At that point, the noVNC proxy will refuse to connect to any Compute node that does not offer VeNCrypt.

As well as enabling the authentication scheme, it is necessary to provide certificates to the noVNC proxy.

- `/etc/pki/nova-novncproxy/client-cert.pem`

  An x509 certificate to be presented **to the VNC server**. While libvirt/QEMU will not currently do any validation of the `CommonName` field, future versions will allow for setting up access controls based on the `CommonName`. The `CommonName` field should match the **primary hostname of the controller node**. If using a HA deployment, the `Organization` field can also be configured to a value that is common across all console proxy instances in the deployment. This avoids the need to modify each compute nodes whitelist every time a console proxy instance is added or removed.

- `/etc/pki/nova-novncproxy/client-key.pem`

  The private key used to generate the `client-cert.pem` file.

- `/etc/pki/nova-novncproxy/ca-cert.pem`

  The certificate authority cert used to sign `client-cert.pem` and sign the compute node VNC server certificates.

The certificates must have v3 basic constraints[2] present to indicate the permitted key use and purpose data.

Once the certificates have been created, the noVNC console proxy service must be told where to find them. This requires editing `nova.conf` to set.

```
[vnc]
vencrypt_client_key=/etc/pki/nova-novncproxy/client-key.pem
vencrypt_client_cert=/etc/pki/nova-novncproxy/client-cert.pem
vencrypt_ca_certs=/etc/pki/nova-novncproxy/ca-cert.pem
```

SPICE console

The VNC protocol is fairly limited, lacking support for multiple monitors, bi-directional audio, reliable cut-and-paste, video streaming and more. SPICE is a new protocol that aims to address the limitations in VNC and provide good remote desktop support.

SPICE support in OpenStack Compute shares a similar architecture to the VNC implementation. The OpenStack dashboard uses a SPICE-HTML5 widget in its console tab that communicates with the **`nova-spicehtml5proxy`** service by using SPICE-over-websockets. The **`nova-spicehtml5proxy`** service communicates directly with the hypervisor process by using SPICE.

Configuration

Important: VNC must be explicitly disabled to get access to the SPICE console. Set the *vnc.enabled* option to `False` to disable the VNC console.

To enable the SPICE console service, you must configure both the **nova-spicehtml5proxy** service and the **nova-compute** service. Most options are defined in the *spice* group.

The **nova-spicehtml5proxy** service accepts the following options.

- *daemon*
- *ssl_only*
- *source_is_ipv6*
- *cert*
- *key*
- *web*
- *spice.html5proxy_host*
- *spice.html5proxy_port*

For example, to configure this via a `nova-spicehtml5proxy.conf` file:

```
[spice]
html5proxy_host = 0.0.0.0
html5proxy_port = 6082
```

The **nova-compute** service requires the following options to configure SPICE console support.

- *spice.enabled*
- *spice.agent_enabled*
- *spice.html5proxy_base_url*
- *spice.server_listen*
- *spice.server_proxyclient_address*
- *spice.keymap*

For example, to configure this via a `nova.conf` file:

```
[spice]
agent_enabled = False
enabled = True
html5proxy_base_url = http://IP_ADDRESS:6082/spice_auto.html
server_listen = 127.0.0.1
server_proxyclient_address = 127.0.0.1
keymap = en-us
```

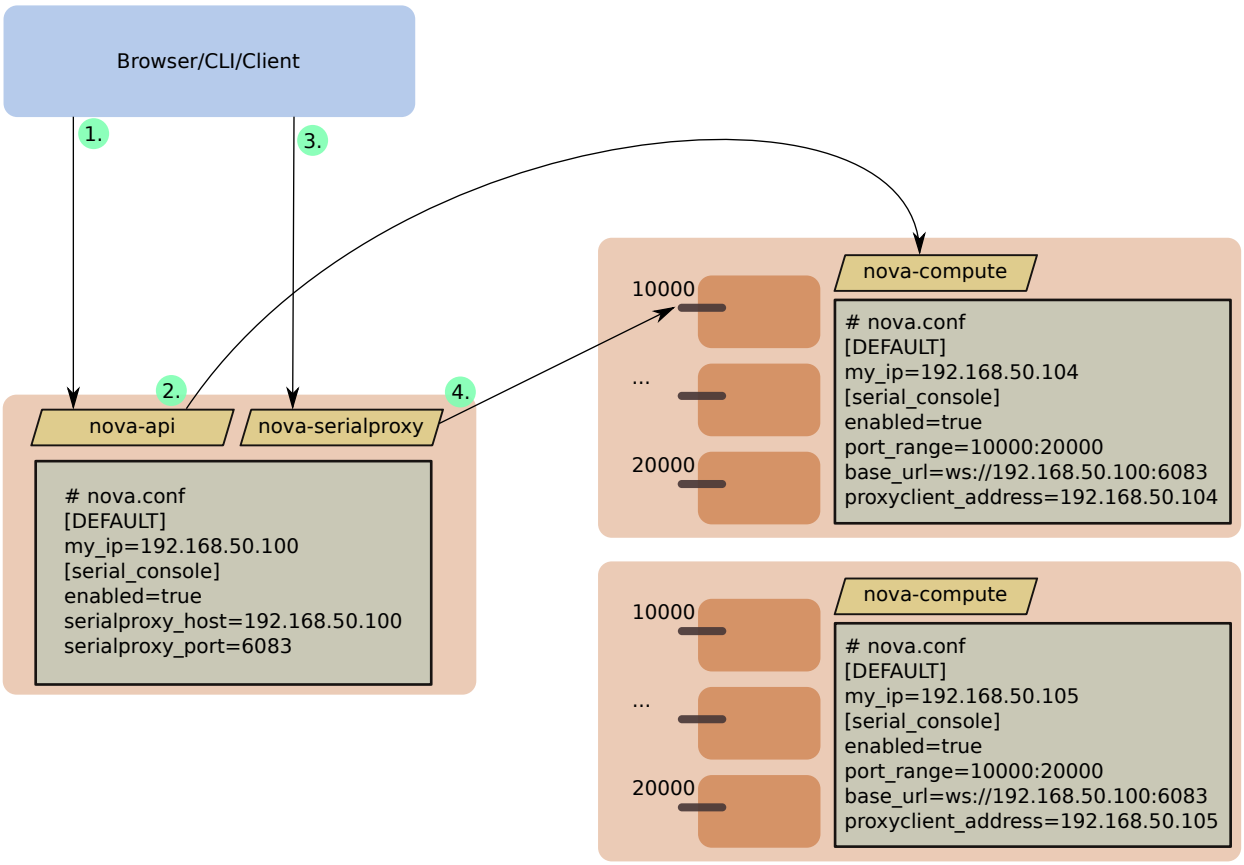
Replace `IP_ADDRESS` with the IP address from which the proxy is accessible by the outside world. For example, this may be the management interface IP address of the controller or the VIP.

Serial

Serial consoles provide an alternative to graphical consoles like VNC or SPICE. They work a little differently to graphical consoles so an example is beneficial. The example below uses these nodes:

- controller node with IP `192.168.50.100`
- compute node 1 with IP `192.168.50.104`
- compute node 2 with IP `192.168.50.105`

Heres the general flow of actions:



1. The user requests a serial console connection string for an instance from the REST API.
2. The **`nova-api`** service asks the **`nova-compute`** service, which manages that instance, to fulfill that request.
3. That connection string gets used by the user to connect to the **`nova-serialproxy`** service.
4. The **`nova-serialproxy`** service then proxies the console interaction to the port of the compute node where the instance is running. That port gets forwarded by the hypervisor (or ironic conductor, for ironic) to the guest.

Configuration

To enable the serial console service, you must configure both the **`nova-serialproxy`** service and the **`nova-compute`** service. Most options are defined in the *`serial_console`* group.

The **`nova-serialproxy`** service accepts the following options.

- *daemon*
- *ssl_only*
- *source_is_ipv6*
- *cert*
- *key*
- *web*
- *serial_console.serialproxy_host*
- *serial_console.serialproxy_port*

For example, to configure this via a `nova-serialproxy.conf` file:

```
[serial_console]
serialproxy_host = 0.0.0.0
serialproxy_port = 6083
```

The **nova-compute** service requires the following options to configure serial console support.

- *serial_console.enabled*
- *serial_console.base_url*
- *serial_console.proxyclient_address*
- *serial_console.port_range*

For example, to configure this via a `nova.conf` file:

```
[serial_console]
enabled = True
base_url = ws://IP_ADDRESS:6083/
proxyclient_address = 127.0.0.1
port_range = 10000:20000
```

Replace `IP_ADDRESS` with the IP address from which the proxy is accessible by the outside world. For example, this may be the management interface IP address of the controller or the VIP.

There are some things to keep in mind when configuring these options:

- *serial_console.serialproxy_host* is the address the **nova-serialproxy** service listens to for incoming connections.
- *serial_console.serialproxy_port* must be the same value as the port in the URI of *serial_console.base_url*.
- The URL defined in *serial_console.base_url* will form part of the response the user will get when asking for a serial console connection string. This means it needs to be an URL the user can connect to.
- *serial_console.proxyclient_address* will be used by the **nova-serialproxy** service to determine where to connect to for proxying the console interaction.

RDP

RDP is a graphical console primarily used with Hyper-V. Nova does not provide a console proxy service for RDP - instead, an external proxy service, such as the **wsgate** application provided by FreeRDP-WebConnect, should be used.

Configuration

To enable the RDP console service, you must configure both a console proxy service like **wsgate** and the **nova-compute** service. All options for the latter service are defined in the *rdp* group.

Information on configuring an RDP console proxy service, such as **wsgate**, is not provided here. However, more information can be found at cloudbase.it.

The **nova-compute** service requires the following options to configure RDP console support.

- *rdp.enabled*
- *rdp.html5_proxy_base_url*

For example, to configure this via a `nova.conf` file:

```
[rdp]
enabled = True
html5_proxy_base_url = https://IP_ADDRESS:6083/
```

Replace `IP_ADDRESS` with the IP address from which the proxy is accessible by the outside world. For example, this may be the management interface IP address of the controller or the VIP.

MKS

MKS is the protocol used for accessing the console of a virtual machine running on VMware vSphere. It is very similar to VNC. Due to the architecture of the VMware vSphere hypervisor, it is not necessary to run a console proxy service.

Configuration

To enable the MKS console service, only the **nova-compute** service must be configured. All options are defined in the *mks* group.

The **nova-compute** service requires the following options to configure MKS console support.

- *mks.enabled*
- *mks.mksproxy_base_url*

For example, to configure this via a `nova.conf` file:

```
[mks]
enabled = True
mksproxy_base_url = https://127.0.0.1:6090/
```

XVP-based VNC console

VNC is a graphical console with wide support among many hypervisors and clients. Xen VNC Proxy (XVP) provides VNC support via a simple Java client.

Deprecated since version 19.0.0: **nova-xvpvnxproxy** is deprecated since 19.0.0 (Stein) and will be removed in an upcoming release.

Configuration

To enable the XVP VNC console service, you must configure both the **nova-xvpvncproxy** service and the **nova-compute** service. Most options are defined in the *vnc* group.

The **nova-xvpvncproxy** service accepts the following options.

- *daemon*
- *ssl_only*
- *source_is_ipv6*
- *cert*
- *key*
- *web*
- *vnc.xvpvncproxy_host*
- *vnc.xvpvncproxy_port*

For example, to configure this via a `nova-xvpvncproxy.conf` file:

```
[vnc]
xvpvncproxy_host = 0.0.0.0
xvpvncproxy_port = 6081
```

The **nova-compute** service requires the following options to configure XVP-based VNC support.

- *vnc.enabled*
- *vnc.xvpvncproxy_base_url*
- *vnc.server_listen*
- *vnc.server_proxyclient_address*
- *vnc.keymap*

For example, to configure this via a `nova.conf` file:

```
[vnc]
enabled = True
xvpvncproxy_base_url = http://IP_ADDRESS:6081/console
server_listen = 127.0.0.1
server_proxyclient_address = 127.0.0.1
keymap = en-us
```

Replace `IP_ADDRESS` with the IP address from which the proxy is accessible by the outside world. For example, this may be the management interface IP address of the controller or the VIP.

About nova-consoleauth

The now-removed **nova-consoleauth** service was previously used to provide a shared service to manage token authentication that the client proxies outlined below could leverage. Token authentication was moved to the database in 18.0.0 (Rocky) and the service was removed in 20.0.0 (Train).

Frequently Asked Questions

- **Q: What is the difference between ‘‘nova-xvpvncproxy‘‘ and ‘‘nova-novncproxy‘‘?**

  A: `nova-xvpvncproxy`, which ships with OpenStack Compute, is a proxy that supports a simple Java client. `nova-novncproxy` uses noVNC to provide VNC support through a web browser.

- **Q: I want VNC support in the OpenStack dashboard. What services do I need?**

  A: You need `nova-novncproxy` and correctly configured compute hosts.

- **Q: My VNC proxy worked fine during my all-in-one test, but now it doesnt work on multi host. Why?**

  A: The default options work for an all-in-one install, but changes must be made on your compute hosts once you start to build a cluster. As an example, suppose you have two servers:

```
PROXYSERVER (public_ip=172.24.1.1, management_ip=192.168.1.1)
COMPUTESERVER (management_ip=192.168.1.2)
```

  Your `nova-compute` configuration file must set the following values:

```
[vnc]
# These flags help construct a connection data structure
server_proxyclient_address=192.168.1.2
novncproxy_base_url=http://172.24.1.1:6080/vnc_auto.html
xvpvncproxy_base_url=http://172.24.1.1:6081/console

# This is the address where the underlying vncserver (not the proxy)
# will listen for connections.
server_listen=192.168.1.2
```

  Note: `novncproxy_base_url` and `xvpvncproxy_base_url` use a public IP; this is the URL that is ultimately returned to clients, which generally do not have access to your private network. Your PROXYSERVER must be able to reach `server_proxyclient_address`, because that is the address over which the VNC connection is proxied.

- **Q: My noVNC does not work with recent versions of web browsers. Why?**

  A: Make sure you have installed `python-numpy`, which is required to support a newer version of the WebSocket protocol (HyBi-07+).

- **Q: How do I adjust the dimensions of the VNC window image in the OpenStack dashboard?**

  A: These values are hard-coded in a Django HTML template. To alter them, edit the `_detail_vnc.html` template file. The location of this file varies based on Linux distribution. On Ubuntu 14.04, the file is at `/usr/share/pyshared/horizon/dashboards/nova/instances/templates/instances/_detail_vnc.html`.

  Modify the `width` and `height` options, as follows:

```
<iframe src="{{ vnc_url }}" width="720" height="430"></iframe>
```

- **Q: My noVNC connections failed with ValidationError: Origin header protocol does not match. Why?**

  A: Make sure the `base_url` match your TLS setting. If you are using https console connections, make sure that the value of `novncproxy_base_url` is set explicitly where the `nova-novncproxy` service is running.

References

4.3.27 Secure with rootwrap

Rootwrap allows unprivileged users to safely run Compute actions as the root user. Compute previously used **sudo** for this purpose, but this was difficult to maintain, and did not allow advanced filters. The **rootwrap** command replaces **sudo** for Compute.

To use rootwrap, prefix the Compute command with **nova-rootwrap**. For example:

```
$ sudo nova-rootwrap /etc/nova/rootwrap.conf command
```

A generic `sudoers` entry lets the Compute user run **nova-rootwrap** as root. The **nova-rootwrap** code looks for filter definition directories in its configuration file, and loads command filters from them. It then checks if the command requested by Compute matches one of those filters and, if so, executes the command (as root). If no filter matches, it denies the request.

Note: Be aware of issues with using NFS and root-owned files. The NFS share must be configured with the `no_root_squash` option enabled, in order for rootwrap to work correctly.

Rootwrap is fully controlled by the root user. The root user owns the sudoers entry which allows Compute to run a specific rootwrap executable as root, and only with a specific configuration file (which should also be owned by root). The **nova-rootwrap** command imports the Python modules it needs from a cleaned, system-default PYTHONPATH. The root-owned configuration file points to root-owned filter definition directories, which contain root-owned filters definition files. This chain ensures that the Compute user itself is not in control of the configuration or modules used by the **nova-rootwrap** executable.

Configure rootwrap

Configure rootwrap in the `rootwrap.conf` file. Because it is in the trusted security path, it must be owned and writable by only the root user. The `rootwrap_config=entry` parameter specifies the files location in the sudoers entry and in the `nova.conf` configuration file.

The `rootwrap.conf` file uses an INI file format with these sections and parameters:

Table 9: **rootwrap.conf configuration options**

Configuration option=Default value	(Type) Description
[DEFAULT] filters_path=/etc/nova/rootwrap.d,/usr/share/nova/rootwrap	(ListOpt) Comma-separated list of directories containing filter definition files. Defines where rootwrap filters are stored. Directories defined on this line should all exist, and be owned and writable only by the root user.

If the root wrapper is not performing correctly, you can add a workaround option into the `nova.conf` configuration file. This workaround re-configures the root wrapper configuration to fall back to running commands as `sudo`, and is a Kilo release feature.

Including this workaround in your configuration file safeguards your environment from issues that can impair root wrapper performance. Tool changes that have impacted Python Build Reasonableness (PBR) for example, are a known issue that affects root wrapper performance.

To set up this workaround, configure the `disable_rootwrap` option in the `[workaround]` section of the `nova.conf` configuration file.

The filters definition files contain lists of filters that rootwrap will use to allow or deny a specific command. They are generally suffixed by `.filters` . Since they are in the trusted security path, they need to be owned and writable only by the root user. Their location is specified in the `rootwrap.conf` file.

Filter definition files use an INI file format with a `[Filters]` section and several lines, each with a unique parameter name, which should be different for each filter you define:

Table 10: **Filters configuration options**

Configuration option=Default value	(Type) Description
[Filters] filter_name=kpartx: CommandFilter, /sbin/kpartx, root	(ListOpt) Comma-separated list containing the filter class to use, followed by the Filter arguments (which vary depending on the Filter class selected).

Configure the rootwrap daemon

Administrators can use rootwrap daemon support instead of running rootwrap with **`sudo`**. The rootwrap daemon reduces the overhead and performance loss that results from running `oslo.rootwrap` with **`sudo`**. Each call that needs rootwrap privileges requires a new instance of rootwrap. The daemon prevents overhead from the repeated calls. The daemon does not support long running processes, however.

To enable the rootwrap daemon, set `use_rootwrap_daemon` to `True` in the Compute service configuration file.

4.3.28 Manage project security

Security groups are sets of IP filter rules that are applied to all project instances, which define networking access to the instance. Group rules are project specific; project members can edit the default rules for their group and add new rule sets.

All projects have a `default` security group which is applied to any instance that has no other defined security group. Unless you change the default, this security group denies all incoming traffic and allows only outgoing traffic to your instance.

By default, security groups (and their quota) are managed by the Neutron networking service.

Working with security groups

From the command-line you can get a list of security groups for the project, using the **`openstack`** commands.

List and view current security groups

1. Ensure your system variables are set for the user and project for which you are checking security group rules. For example:

```
export OS_USERNAME=demo00
export OS_TENANT_NAME=tenant01
```

2. Output security groups, as follows:

```
$ openstack security group list
+--------------------------------------+---------+-------------+
| Id                                   | Name    | Description |
+--------------------------------------+---------+-------------+
| 73580272-d8fa-4927-bd55-c85e43bc4877 | default | default     |
| 6777138a-deb7-4f10-8236-6400e7aff5b0 | open    | all ports   |
+--------------------------------------+---------+-------------+
```

3. View the details of a group, as follows:

```
$ openstack security group rule list GROUPNAME
```

For example:

```
$ openstack security group rule list open
+--------------------------------------+-------------+-----------+-----------------
↪-+-----------------------+
| ID                                   | IP Protocol | IP Range  | Port Range
↪ | Remote Security Group |
+--------------------------------------+-------------+-----------+-----------------
↪-+-----------------------+
| 353d0611-3f67-4848-8222-a92adbdb5d3a | udp         | 0.0.0.0/0 | 1:65535
↪ | None                  |
| 63536865-e5b6-4df1-bac5-ca6d97d8f54d | tcp         | 0.0.0.0/0 | 1:65535
↪ | None                  |
+--------------------------------------+-------------+-----------+-----------------
↪-+-----------------------+
```

These rules are allow type rules as the default is deny. The first column is the IP protocol (one of ICMP, TCP, or UDP). The second and third columns specify the affected port range. The third column specifies the IP range in CIDR format. This example shows the full port range for all protocols allowed from all IPs.

Create a security group

When adding a new security group, you should pick a descriptive but brief name. This name shows up in brief descriptions of the instances that use it where the longer description field often does not. For example, seeing that an instance is using security group http is much easier to understand than bobs_group or secgrp1.

1. Ensure your system variables are set for the user and project for which you are creating security group rules.
2. Add the new security group, as follows:

```
$ openstack security group create GroupName --description Description
```

For example:

```
$ openstack security group create global_http --description "Allows Web traffic
↪anywhere on the Internet."
+-----------------+-----------------------------------------------------------------
↪-----------------------------------------------------+
| Field           | Value
↪                                                    |
+-----------------+-----------------------------------------------------------------
↪-----------------------------------------------------+
| created_at      | 2016-11-03T13:50:53Z
↪                                                    |
```

(continues on next page)

(continued from previous page)

```
| description     | Allows Web traffic anywhere on the Internet.                 ␣
↪                                                                 |
| headers         |                                                              ␣
↪                                                                 |
| id              | c0b92b20-4575-432a-b4a9-eaf2ad53f696                         ␣
↪                                                                 |
| name            | global_http                                                  ␣
↪                                                                 |
| project_id      | 5669caad86a04256994cdf755df4d3c1                             ␣
↪                                                                 |
| project_id      | 5669caad86a04256994cdf755df4d3c1                             ␣
↪                                                                 |
| revision_number | 1                                                            ␣
↪                                                                 |
| rules           | created_at='2016-11-03T13:50:53Z', direction='egress',␣
↪ethertype='IPv4', id='4d8cec94-e0ee-4c20-9f56-8fb67c21e4df',      |
|                 | project_id='5669caad86a04256994cdf755df4d3c1', revision_
↪number='1', updated_at='2016-11-03T13:50:53Z'                     |
|                 | created_at='2016-11-03T13:50:53Z', direction='egress',␣
↪ethertype='IPv6', id='31be2ad1-be14-4aef-9492-ecebede2cf12',      |
|                 | project_id='5669caad86a04256994cdf755df4d3c1', revision_
↪number='1', updated_at='2016-11-03T13:50:53Z'                     |
| updated_at      | 2016-11-03T13:50:53Z                                         ␣
↪                                                                 |
+-----------------+-----------------------------------------------------------------
↪-----------------------------------------------------------+
```

3. Add a new group rule, as follows:

```
$ openstack security group rule create SEC_GROUP_NAME \
    --protocol PROTOCOL --dst-port FROM_PORT:TO_PORT --remote-ip CIDR
```

The arguments are positional, and the `from-port` and `to-port` arguments specify the local port range connections are allowed to access, not the source and destination ports of the connection. For example:

```
$ openstack security group rule create global_http \
    --protocol tcp --dst-port 80:80 --remote-ip 0.0.0.0/0
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| created_at        | 2016-11-06T14:02:00Z                 |
| description       |                                      |
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| headers           |                                      |
| id                | 2ba06233-d5c8-43eb-93a9-8eaa94bc9eb5 |
| port_range_max    | 80                                   |
| port_range_min    | 80                                   |
| project_id        | 5669caad86a04256994cdf755df4d3c1     |
| project_id        | 5669caad86a04256994cdf755df4d3c1     |
| protocol          | tcp                                  |
| remote_group_id   | None                                 |
| remote_ip_prefix  | 0.0.0.0/0                            |
| revision_number   | 1                                    |
| security_group_id | c0b92b20-4575-432a-b4a9-eaf2ad53f696 |
| updated_at        | 2016-11-06T14:02:00Z                 |
+-------------------+--------------------------------------+
```

You can create complex rule sets by creating additional rules. For example, if you want to pass both HTTP and HTTPS traffic, run:

```
$ openstack security group rule create global_http \
    --protocol tcp --dst-port 443:443 --remote-ip 0.0.0.0/0
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| created_at        | 2016-11-06T14:09:20Z                 |
| description       |                                      |
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| headers           |                                      |
| id                | 821c3ef6-9b21-426b-be5b-c8a94c2a839c |
| port_range_max    | 443                                  |
| port_range_min    | 443                                  |
| project_id        | 5669caad86a04256994cdf755df4d3c1     |
| project_id        | 5669caad86a04256994cdf755df4d3c1     |
| protocol          | tcp                                  |
| remote_group_id   | None                                 |
| remote_ip_prefix  | 0.0.0.0/0                            |
| revision_number   | 1                                    |
| security_group_id | c0b92b20-4575-432a-b4a9-eaf2ad53f696 |
| updated_at        | 2016-11-06T14:09:20Z                 |
+-------------------+--------------------------------------+
```

Despite only outputting the newly added rule, this operation is additive (both rules are created and enforced).

4. View all rules for the new security group, as follows:

```
$ openstack security group rule list global_http
+--------------------------------------+-------------+-----------+-----------------
↪-+----------------------+
| ID                                   | IP Protocol | IP Range  | Port Range     ␣
↪ | Remote Security Group |
+--------------------------------------+-------------+-----------+-----------------
↪-+----------------------+
| 353d0611-3f67-4848-8222-a92adbdb5d3a | tcp         | 0.0.0.0/0 | 80:80          ␣
↪ | None                 |
| 63536865-e5b6-4df1-bac5-ca6d97d8f54d | tcp         | 0.0.0.0/0 | 443:443        ␣
↪ | None                 |
+--------------------------------------+-------------+-----------+-----------------
↪-+----------------------+
```

Delete a security group

1. Ensure your system variables are set for the user and project for which you are deleting a security group.
2. Delete the new security group, as follows:

```
$ openstack security group delete GROUPNAME
```

For example:

```
$ openstack security group delete global_http
```

Create security group rules for a cluster of instances

Source Groups are a special, dynamic way of defining the CIDR of allowed sources. The user specifies a Source Group (Security Group name), and all the users other Instances using the specified Source Group are selected dynamically. This alleviates the need for individual rules to allow each new member of the cluster.

1. Make sure to set the system variables for the user and project for which you are creating a security group rule.
2. Add a source group, as follows:

```
$ openstack security group rule create secGroupName \
    --remote-group source-group --protocol ip-protocol \
    --dst-port from-port:to-port
```

For example:

```
$ openstack security group rule create cluster \
    --remote-group global_http --protocol tcp --dst-port 22:22
```

The `cluster` rule allows SSH access from any other instance that uses the `global_http` group.

nova-network configuration (deprecated)

You can use the *`allow_same_net_traffic`* option in the `/etc/nova/nova.conf` file to globally control whether the rules apply to hosts which share a network. There are two possible values:

`True` (default) Hosts on the same subnet are not filtered and are allowed to pass all types of traffic between them. On a flat network, this allows all instances from all projects unfiltered communication. With VLAN networking, this allows access between instances within the same project. You can also simulate this setting by configuring the default security group to allow all traffic from the subnet.

`False` Security groups are enforced for all connections.

Additionally, the number of maximum rules per security group is controlled by the `security_group_rules` and the number of allowed security groups per project is controlled by the `security_groups` quota (see *Manage quotas*).

4.3.29 Security hardening

OpenStack Compute can be integrated with various third-party technologies to increase security. For more information, see the OpenStack Security Guide.

Encrypt Compute metadata traffic

Enabling SSL encryption

OpenStack supports encrypting Compute metadata traffic with HTTPS. Enable SSL encryption in the `metadata_agent.ini` file.

1. Enable the HTTPS protocol.

```
nova_metadata_protocol = https
```

2. Determine whether insecure SSL connections are accepted for Compute metadata server requests. The default value is `False`.

```
nova_metadata_insecure = False
```

3. Specify the path to the client certificate.

```
nova_client_cert = PATH_TO_CERT
```

4. Specify the path to the private key.

```
nova_client_priv_key = PATH_TO_KEY
```

Securing live migration streams with QEMU-native TLS

It is strongly recommended to secure all the different live migration streams of a nova instancei.e. guest RAM, device state, and disks (via NBD) when using non-shared storage. For further details on how to set this up, refer to the *Secure live migration with QEMU-native TLS* document.

Mitigation for MDS (Microarchitectural Data Sampling) security flaws

It is strongly recommended to patch all compute nodes and nova instances against the processor-related security flaws, such as MDS (and other previous vulnerabilities). For details on applying mitigation for the MDS flaws, refer to the *Mitigation for MDS (Microarchitectural Data Sampling) Security Flaws* document.

4.3.30 Configure Compute service groups

The Compute service must know the status of each compute node to effectively manage and use them. This can include events like a user launching a new VM, the scheduler sending a request to a live node, or a query to the ServiceGroup API to determine if a node is live.

When a compute worker running the nova-compute daemon starts, it calls the join API to join the compute group. Any service (such as the scheduler) can query the groups membership and the status of its nodes. Internally, the ServiceGroup client driver automatically updates the compute worker status.

Database ServiceGroup driver

By default, Compute uses the database driver to track if a node is live. In a compute worker, this driver periodically sends a `db update` command to the database, saying Im OK with a timestamp. Compute uses a pre-defined timeout (`service_down_time`) to determine if a node is dead.

The driver has limitations, which can be problematic depending on your environment. If a lot of compute worker nodes need to be checked, the database can be put under heavy load, which can cause the timeout to trigger, and a live node could incorrectly be considered dead. By default, the timeout is 60 seconds. Reducing the timeout value can help in this situation, but you must also make the database update more frequently, which again increases the database workload.

The database contains data that is both transient (such as whether the node is alive) and persistent (such as entries for VM owners). With the ServiceGroup abstraction, Compute can treat each type separately.

Memcache ServiceGroup driver

The memcache ServiceGroup driver uses memcached, a distributed memory object caching system that is used to increase site performance. For more details, see memcached.org.

To use the memcache driver, you must install memcached. You might already have it installed, as the same driver is also used for the OpenStack Object Storage and OpenStack dashboard. To install memcached, see the *Environment -> Memcached* section in the Installation Tutorials and Guides depending on your distribution.

These values in the `/etc/nova/nova.conf` file are required on every node for the memcache driver:

```
# Driver for the ServiceGroup service
servicegroup_driver = "mc"

# Memcached servers. Use either a list of memcached servers to use for caching (list␣
↪value),
# or "<None>" for in-process caching (default).
memcached_servers = <None>

# Timeout; maximum time since last check-in for up service (integer value).
# Helps to define whether a node is dead
service_down_time = 60
```

4.3.31 Manage Compute services

You can enable and disable Compute services. The following examples disable and enable the `nova-compute` service.

1. List the Compute services:

```
$ openstack compute service list
+----+----------------+------------+----------+---------+-------+----------------
↪-----------+
| ID | Binary         | Host       | Zone     | Status  | State | Updated At     ␣
↪           |
+----+----------------+------------+----------+---------+-------+----------------
↪-----------+
|  4 | nova-scheduler | controller | internal | enabled | up    | 2016-12-
↪20T00:44:48.000000 |
|  5 | nova-conductor | controller | internal | enabled | up    | 2016-12-
↪20T00:44:54.000000 |
|  8 | nova-compute   | compute    | nova     | enabled | up    | 2016-10-
↪21T02:35:03.000000 |
+----+----------------+------------+----------+---------+-------+----------------
↪-----------+
```

2. Disable a nova service:

```
$ openstack compute service set --disable --disable-reason "trial log" compute␣
↪nova-compute
+----------+--------------+----------+--------------------+
| Host     | Binary       | Status   | Disabled Reason    |
+----------+--------------+----------+--------------------+
| compute  | nova-compute | disabled | trial log          |
+----------+--------------+----------+--------------------+
```

3. Check the service list:

```
$ openstack compute service list
+----+----------------+------------+----------+---------+-------+----------------
↪-----------+
| ID | Binary         | Host       | Zone     | Status  | State | Updated At     ␣
↪           |
```

(continues on next page)

(continued from previous page)

```
+----+----------------+------------+----------+---------+-------+-----------------
↪-----------+
|  5 | nova-scheduler | controller | internal | enabled | up    | 2016-12-
↪20T00:44:48.000000 |
|  6 | nova-conductor | controller | internal | enabled | up    | 2016-12-
↪20T00:44:54.000000 |
|  9 | nova-compute   | compute    | nova     | disabled| up    | 2016-10-
↪21T02:35:03.000000 |
+----+----------------+------------+----------+---------+-------+-----------------
↪-----------+
```

4. Enable the service:

```
$ openstack compute service set --enable compute nova-compute
+----------+--------------+---------+
| Host     | Binary       | Status  |
+----------+--------------+---------+
| compute  | nova-compute | enabled |
+----------+--------------+---------+
```

4.3.32 Configure SSH between compute nodes

Todo: Consider merging this into a larger migration document or to the installation guide

If you are resizing or migrating an instance between hypervisors, you might encounter an SSH (Permission denied) error. Ensure that each node is configured with SSH key authentication so that the Compute service can use SSH to move disks to other nodes.

Note: It is not necessary that all the compute nodes share the same key pair. However for the ease of the configuration, this document only utilizes a single key pair for communication between compute nodes.

To share a key pair between compute nodes, complete the following steps:

1. On the first node, obtain a key pair (public key and private key). Use the root key that is in the `/root/.ssh/id_rsa` and `/root/.ssh/id_rsa.pub` directories or generate a new key pair.
2. Run **`setenforce 0`** to put SELinux into permissive mode.
3. Enable login abilities for the nova user:

```
# usermod -s /bin/bash nova
```

   Ensure you can switch to the nova account:

```
# su - nova
```

4. As root, create the folder that is needed by SSH and place the private key that you obtained in step 1 into this folder, and add the pub key to the authorized_keys file:

```
mkdir -p /var/lib/nova/.ssh
cp <private key>  /var/lib/nova/.ssh/id_rsa
echo 'StrictHostKeyChecking no' >> /var/lib/nova/.ssh/config
```

(continues on next page)

(continued from previous page)

```
chmod 600 /var/lib/nova/.ssh/id_rsa /var/lib/nova/.ssh/authorized_keys
echo <pub key> >> /var/lib/nova/.ssh/authorized_keys
```

5. Copy the whole folder created in step 4 to the rest of the nodes:

```
# scp -r /var/lib/nova/.ssh remote-host:/var/lib/nova/
```

6. Ensure that the nova user can now log in to each node without using a password:

```
# su - nova
$ ssh *computeNodeAddress*
$ exit
```

7. As root on each node, restart both libvirt and the Compute services:

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

4.3.33 Troubleshoot Compute

Common problems for Compute typically involve misconfigured networking or credentials that are not sourced properly in the environment. Also, most flat networking configurations do not enable **`ping`** or **`ssh`** from a compute node to the instances that run on that node. Another common problem is trying to run 32-bit images on a 64-bit compute node. This section shows you how to troubleshoot Compute.

Compute service logging

Compute stores a log file for each service in `/var/log/nova`. For example, `nova-compute.log` is the log for the `nova-compute` service. You can set the following options to format log strings for the `nova.log` module in the `nova.conf` file:

- `logging_context_format_string`
- `logging_default_format_string`

If the log level is set to `debug`, you can also specify `logging_debug_format_suffix` to append extra formatting. For information about what variables are available for the formatter, see Formatter Objects.

You have two logging options for OpenStack Compute based on configuration settings. In `nova.conf`, include the `logfile` option to enable logging. Alternatively you can set `use_syslog = 1` so that the nova daemon logs to syslog.

Guru Meditation reports

A Guru Meditation report is sent by the Compute service upon receipt of the `SIGUSR2` signal (`SIGUSR1` before Mitaka). This report is a general-purpose error report that includes details about the current state of the service. The error report is sent to `stderr`.

For example, if you redirect error output to `nova-api-err.log` using **`nova-api 2>/var/log/nova/nova-api-err.log`**, resulting in the process ID 8675, you can then run:

```
# kill -USR2 8675
```

This command triggers the Guru Meditation report to be printed to `/var/log/nova/nova-api-err.log`.

The report has the following sections:

- Package: Displays information about the package to which the process belongs, including version information.
- Threads: Displays stack traces and thread IDs for each of the threads within the process.
- Green Threads: Displays stack traces for each of the green threads within the process (green threads do not have thread IDs).
- Configuration: Lists all configuration options currently accessible through the CONF object for the current process.

For more information, see *Guru Meditation Reports*.

Common errors and fixes for Compute

The ask.openstack.org site offers a place to ask and answer questions, and you can also mark questions as frequently asked questions. This section describes some errors people have posted previously. Bugs are constantly being fixed, so online resources are a great way to get the most up-to-date errors and fixes.

Credential errors, 401, and 403 forbidden errors

Problem

Missing credentials cause a `403 forbidden` error.

Solution

To resolve this issue, use one of these methods:

1. Manual method

   Gets the `novarc` file from the project ZIP file, saves existing credentials in case of override, and manually sources the `novarc` file.

2. Script method

   Generates `novarc` from the project ZIP file and sources it for you.

When you run `nova-api` the first time, it generates the certificate authority information, including `openssl.cnf`. If you start the CA services before this, you might not be able to create your ZIP file. Restart the services. When your CA information is available, create your ZIP file.

Also, check your HTTP proxy settings to see whether they cause problems with `novarc` creation.

Live migration permission issues

Problem

When live migrating an instance, you may see errors like the below:

```
libvirtError: operation failed: Failed to connect to remote libvirt URI
qemu+ssh://stack@cld6b16/system: Cannot recv data: Host key verification
failed.: Connection reset by peer
```

Solution

Ensure you have completed all the steps outlined in *Configure SSH between compute nodes*. In particular, its important to note that the `libvirt` process runs as `root` even though it may be connecting to a different user (`stack` in the above example). You can ensure everything is correctly configured by attempting to connect to the remote host via the `root` user. Using the above example once again:

```
$ su - -c 'ssh stack@cld6b16'
```

Instance errors

Problem

Sometimes a particular instance shows `pending` or you cannot SSH to it. Sometimes the image itself is the problem. For example, when you use flat manager networking, you do not have a DHCP server and certain images do not support interface injection; you cannot connect to them.

Solution

To fix instance errors use an image that does support this method, such as Ubuntu, which obtains an IP address correctly with FlatManager network settings.

To troubleshoot other possible problems with an instance, such as an instance that stays in a spawning state, check the directory for the particular instance under `/var/lib/nova/instances` on the `nova-compute` host and make sure that these files are present:

- `libvirt.xml`
- `disk`
- `disk-raw`
- `kernel`
- `ramdisk`
- `console.log`, after the instance starts.

If any files are missing, empty, or very small, the `nova-compute` service did not successfully download the images from the Image service.

Also check `nova-compute.log` for exceptions. Sometimes they do not appear in the console output.

Next, check the log file for the instance in the `/var/log/libvirt/qemu` directory to see if it exists and has any useful error messages in it.

Finally, from the `/var/lib/nova/instances` directory for the instance, see if this command returns an error:

```
# virsh create libvirt.xml
```

Empty log output for Linux instances

Problem

You can view the log output of running instances from either the *Log* tab of the dashboard or the output of **nova console-log**. In some cases, the log output of a running Linux instance will be empty or only display a single character (for example, the *?* character).

This occurs when the Compute service attempts to retrieve the log output of the instance via a serial console while the instance itself is not configured to send output to the console.

Solution

To rectify this, append the following parameters to kernel arguments specified in the instances boot loader:

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

Upon rebooting, the instance will be configured to send output to the Compute service.

Reset the state of an instance

Problem

Instances can remain in an intermediate state, such as `deleting`.

Solution

You can use the **nova reset-state** command to manually reset the state of an instance to an error state. You can then delete the instance. For example:

```
$ nova reset-state c6bbbf26-b40a-47e7-8d5c-eb17bf65c485
$ openstack server delete c6bbbf26-b40a-47e7-8d5c-eb17bf65c485
```

You can also use the `--active` parameter to force the instance back to an active state instead of an error state. For example:

```
$ nova reset-state --active c6bbbf26-b40a-47e7-8d5c-eb17bf65c485
```

Injection problems

Problem

Instances may boot slowly, or do not boot. File injection can cause this problem.

Solution

To disable injection in libvirt, set the following in `nova.conf`:

```
[libvirt]
inject_partition = -2
```

Note: If you have not enabled the config drive and you want to make user-specified files available from the metadata server for to improve performance and avoid boot failure if injection fails, you must disable injection.

Cannot find suitable emulator for x86_64

Problem

When you attempt to create a VM, the error shows the VM is in the `BUILD` then `ERROR` state.

Solution

On the KVM host, run **`cat /proc/cpuinfo`**. Make sure the `vmx` or `svm` flags are set.

Follow the instructions in the *Enable KVM* section in the Nova Configuration Reference to enable hardware virtualization support in your BIOS.

Failed to attach volume after detaching

Problem

Failed to attach a volume after detaching the same volume.

Solution

You must change the device name on the **`nova-attach`** command. The VM might not clean up after a **`nova-detach`** command runs. This example shows how the **`nova-attach`** command fails when you use the `vdb`, `vdc`, or `vdd` device names:

```
# ls -al /dev/disk/by-path/
total 0
drwxr-xr-x 2 root root 200 2012-08-29 17:33 .
drwxr-xr-x 5 root root 100 2012-08-29 17:33 ..
lrwxrwxrwx 1 root root 9 2012-08-29 17:33 pci-0000:00:04.0-virtio-pci-virtio0 -> ../..
↪/vda
lrwxrwxrwx 1 root root 10 2012-08-29 17:33 pci-0000:00:04.0-virtio-pci-virtio0-part1 -
↪> ../../vda1
lrwxrwxrwx 1 root root 10 2012-08-29 17:33 pci-0000:00:04.0-virtio-pci-virtio0-part2 -
↪> ../../vda2
lrwxrwxrwx 1 root root 10 2012-08-29 17:33 pci-0000:00:04.0-virtio-pci-virtio0-part5 -
↪> ../../vda5
lrwxrwxrwx 1 root root 9 2012-08-29 17:33 pci-0000:00:06.0-virtio-pci-virtio2 -> ../..
↪/vdb
lrwxrwxrwx 1 root root 9 2012-08-29 17:33 pci-0000:00:08.0-virtio-pci-virtio3 -> ../..
↪/vdc
lrwxrwxrwx 1 root root 9 2012-08-29 17:33 pci-0000:00:09.0-virtio-pci-virtio4 -> ../..
↪/vdd
lrwxrwxrwx 1 root root 10 2012-08-29 17:33 pci-0000:00:09.0-virtio-pci-virtio4-part1 -
↪> ../../vdd1
```

You might also have this problem after attaching and detaching the same volume from the same VM with the same mount point multiple times. In this case, restart the KVM host.

Failed to attach volume, systool is not installed

Problem

This warning and error occurs if you do not have the required `sysfsutils` package installed on the compute node:

```
WARNING nova.virt.libvirt.utils [req-1200f887-c82b-4e7c-a891-fac2e3735dbb\
admin admin|req-1200f887-c82b-4e7c-a891-fac2e3735dbb admin admin] systool\
is not installed
ERROR nova.compute.manager [req-1200f887-c82b-4e7c-a891-fac2e3735dbb admin\
admin|req-1200f887-c82b-4e7c-a891-fac2e3735dbb admin admin]
[instance: df834b5a-8c3f-477a-be9b-47c97626555c|instance: df834b5a-8c3f-47\
7a-be9b-47c97626555c]
Failed to attach volume 13d5c633-903a-4764-a5a0-3336945b1db1 at /dev/vdk.
```

Solution

Install the `sysfsutils` package on the compute node. For example:

```
# apt-get install sysfsutils
```

Failed to connect volume in FC SAN

Problem

The compute node failed to connect to a volume in a Fibre Channel (FC) SAN configuration. The WWN may not be zoned correctly in your FC SAN that links the compute host to the storage array:

```
ERROR nova.compute.manager [req-2ddd5297-e405-44ab-aed3-152cd2cfb8c2 admin\
demo|req-2ddd5297-e405-44ab-aed3-152cd2cfb8c2 admin demo] [instance: 60ebd\
6c7-c1e3-4bf0-8ef0-f07aa4c3d5f3|instance: 60ebd6c7-c1e3-4bf0-8ef0-f07aa4c3\
d5f3]
Failed to connect to volume 6f6a6a9c-dfcf-4c8d-b1a8-4445ff883200 while\
attaching at /dev/vdjTRACE nova.compute.manager [instance: 60ebd6c7-c1e3-4\
bf0-8ef0-f07aa4c3d5f3|instance: 60ebd6c7-c1e3-4bf0-8ef0-f07aa4c3d5f3]
Traceback (most recent call last):...f07aa4c3d5f3\] ClientException: The\
server has either erred or is incapable of performing the requested\
operation.(HTTP 500)(Request-ID: req-71e5132b-21aa-46ee-b3cc-19b5b4ab2f00)
```

Solution

The network administrator must configure the FC SAN fabric by correctly zoning the WWN (port names) from your compute node HBAs.

Multipath call failed exit

Problem

Multipath call failed exit. This warning occurs in the Compute log if you do not have the optional `multipath-tools` package installed on the compute node. This is an optional package and the volume attachment

does work without the multipath tools installed. If the `multipath-tools` package is installed on the compute node, it is used to perform the volume attachment. The IDs in your message are unique to your system.

```
WARNING nova.storage.linuxscsi [req-cac861e3-8b29-4143-8f1b-705d0084e571 \
admin admin|req-cac861e3-8b29-4143-8f1b-705d0084e571 admin admin] \
Multipath call failed exit (96)
```

Solution

Install the `multipath-tools` package on the compute node. For example:

```
# apt-get install multipath-tools
```

Failed to Attach Volume, Missing sg_scan

Problem

Failed to attach volume to an instance, `sg_scan` file not found. This error occurs when the sg3-utils package is not installed on the compute node. The IDs in your message are unique to your system:

```
ERROR nova.compute.manager [req-cf2679fd-dd9e-4909-807f-48fe9bda3642 admin admin|req-
↪cf2679fd-dd9e-4909-807f-48fe9bda3642 admin admin]
[instance: 7d7c92e0-49fa-4a8e-87c7-73f22a9585d5|instance:  7d7c92e0-49fa-4a8e-87c7-
↪73f22a9585d5]
Failed to attach volume  4cc104c4-ac92-4bd6-9b95-c6686746414a at /dev/vdcTRACE nova.
↪compute.manager
[instance:  7d7c92e0-49fa-4a8e-87c7-73f22a9585d5|instance: 7d7c92e0-49fa-4a8e-87c7-
↪73f22a9585d5]
Stdout: '/usr/local/bin/nova-rootwrap: Executable not found: /usr/bin/sg_scan'
```

Solution

Install the `sg3-utils` package on the compute node. For example:

```
# apt-get install sg3-utils
```

Requested microversions are ignored

Problem

When making a request with a microversion beyond 2.1, for example:

```
$ openstack --os-compute-api-version 2.15 server group create \
  --policy soft-anti-affinity my-soft-anti-group
```

It fails saying that soft-anti-affinity is not a valid policy, even thought it is allowed with the 2.15 microversion.

Solution

Ensure the `compute` endpoint in the identity service catalog is pointing at `/v2.1` instead of `/v2`. The former route supports microversions, while the latter route is considered the legacy v2.0 compatibility-mode route which renders all requests as if they were made on the legacy v2.0 API.

User token times out during long-running operations

Problem

Long-running operations such as live migration or snapshot can sometimes overrun the expiry of the user token. In such cases, post operations such as cleaning up after a live migration can fail when the nova-compute service needs to cleanup resources in other services, such as in the block-storage (cinder) or networking (neutron) services.

For example:

```
2018-12-17 13:47:29.591 16987 WARNING nova.virt.libvirt.migration [req-7bc758de-b2e4-
↪461b-a971-f79be6cd4703 313d1247d7b845da9c731eec53e50a26␣
↪2f693c782fa748c2baece8db95b4ba5b - default default] [instance: ead8ecc3-f473-4672-
↪a67b-c44534c6042d] Live migration not completed after 2400 sec
2018-12-17 13:47:30.097 16987 WARNING nova.virt.libvirt.driver [req-7bc758de-b2e4-
↪461b-a971-f79be6cd4703 313d1247d7b845da9c731eec53e50a26␣
↪2f693c782fa748c2baece8db95b4ba5b - default default] [instance: ead8ecc3-f473-4672-
↪a67b-c44534c6042d] Migration operation was cancelled
2018-12-17 13:47:30.299 16987 ERROR nova.virt.libvirt.driver [req-7bc758de-b2e4-461b-
↪a971-f79be6cd4703 313d1247d7b845da9c731eec53e50a26 2f693c782fa748c2baece8db95b4ba5b␣
↪- default default] [instance: ead8ecc3-f473-4672-a67b-c44534c6042d] Live Migration␣
↪failure: operation aborted: migration job: canceled by client: libvirtError:␣
↪operation aborted: migration job: canceled by client
2018-12-17 13:47:30.685 16987 INFO nova.compute.manager [req-7bc758de-b2e4-461b-a971-
↪f79be6cd4703 313d1247d7b845da9c731eec53e50a26 2f693c782fa748c2baece8db95b4ba5b -␣
↪default default] [instance: ead8ecc3-f473-4672-a67b-c44534c6042d] Swapping old␣
↪allocation on 3e32d595-bd1f-4136-a7f4-c6703d2fbe18 held by migration 17bec61d-544d-
↪47e0-a1c1-37f9d7385286 for instance
2018-12-17 13:47:32.450 16987 ERROR nova.volume.cinder [req-7bc758de-b2e4-461b-a971-
↪f79be6cd4703 313d1247d7b845da9c731eec53e50a26 2f693c782fa748c2baece8db95b4ba5b -␣
↪default default] Delete attachment failed for attachment 58997d5b-24f0-4073-819e-
↪97916fb1ee19. Error: The request you have made requires authentication. (HTTP 401)␣
↪Code: 401: Unauthorized: The request you have made requires authentication. (HTTP␣
↪401)
```

Solution

Configure nova to use service user tokens to supplement the regular user token used to initiate the operation. The identity service (keystone) will then authenticate a request using the service user token if the user token has already expired.

To use, create a service user in the identity service similar as you would when creating the `nova` service user.

Then configure the *`service_user`* section of the nova configuration file, for example:

```
[service_user]
send_service_user_token = True
auth_type = password
```

(continues on next page)

(continued from previous page)

```
project_domain_name = Default
project_name = service
user_domain_name = Default
password = secretservice
username = nova
auth_url = https://104.130.216.102/identity
...
```

And configure the other identity options as necessary for the service user, much like you would configure nova to work with the image service (glance) or networking service.

Note: Please note that the role of the *service_user* you configure needs to be a superset of *keystone_authtoken.service_token_roles* (The option *keystone_authtoken.service_token_roles* is configured in cinder, glance and neutron).

4.3.34 System administration

To effectively administer compute, you must understand how the different installed nodes interact with each other. Compute can be installed in many different ways using multiple servers, but generally multiple compute nodes control the virtual servers and a cloud controller node contains the remaining Compute services.

The Compute cloud works using a series of daemon processes named `nova-*` that exist persistently on the host machine. These binaries can all run on the same machine or be spread out on multiple boxes in a large deployment. The responsibilities of services and drivers are:

Services

nova-api Receives XML requests and sends them to the rest of the system. A WSGI app routes and authenticates requests. Supports the OpenStack Compute APIs. A `nova.conf` configuration file is created when Compute is installed.

nova-compute Manages virtual machines. Loads a Service object, and exposes the public methods on ComputeManager through a Remote Procedure Call (RPC).

nova-conductor Provides database-access support for compute nodes (thereby reducing security risks).

nova-objectstore A simple file-based storage system for images that replicates most of the S3 API. It can be replaced with OpenStack Image service and either a simple image manager or OpenStack Object Storage as the virtual machine image storage facility. It must exist on the same node as `nova-compute`.

nova-network Manages floating and fixed IPs, DHCP, bridging and VLANs. Loads a Service object which exposes the public methods on one of the subclasses of NetworkManager. Different networking strategies are available by changing the `network_manager` configuration option to `FlatManager`, `FlatDHCPManager`, or `VLANManager` (defaults to `VLANManager` if nothing is specified).

Deprecated since version 14.0.0: `nova-network` was deprecated in the OpenStack Newton release.

nova-scheduler Dispatches requests for new virtual machines to the correct node.

nova-novncproxy Provides a VNC proxy for browsers, allowing VNC consoles to access virtual machines.

Note: Some services have drivers that change how the service implements its core functionality. For example, the `nova-compute` service supports drivers that let you choose which hypervisor type it can use. `nova-network` and `nova-scheduler` also have drivers.

4.3.35 Secure live migration with QEMU-native TLS

Context

The encryption offered by novas *libvirt.live_migration_tunnelled* does not secure all the different migration streams of a nova instance, namely: guest RAM, device state, and disks (via NBD) when using non-shared storage. Further, the tunnelling via libvirtd has inherent limitations: (a) it cannot handle live migration of disks in a non-shared storage setup (a.k.a. block migration); and (b) has a huge performance overhead and latency, because it burns more CPU and memory bandwidth due to increased number of data copies on both source and destination hosts.

To solve this existing limitation, QEMU and libvirt have gained (refer *below* for version details) support for native TLS, i.e. TLS built into QEMU. This will secure all data transports, including disks that are not on shared storage, without incurring the limitations of the tunnelled via libvirtd transport.

To take advantage of the native TLS support in QEMU and libvirt, nova has introduced new configuration attribute *libvirt.live_migration_with_native_tls*.

Prerequisites

(1) Version requirement: This feature needs at least libvirt 4.4.0 and QEMU 2.11.

(2) A pre-configured TLS environmenti.e. CA, server, and client certificates, their file permissions, et almust be correctly configured (typically by an installer tool) on all relevant compute nodes. To simplify your PKI (Public Key Infrastructure) setup, use deployment tools that take care of handling all the certificate lifecycle management. For example, refer to the TLS everywhere guide from the TripleO project.

(3) Password-less SSH setup for all relevant compute nodes.

(4) On all relevant compute nodes, ensure the TLS-related config attributes in `/etc/libvirt/qemu.conf` are in place:

```
default_tls_x509_cert_dir = "/etc/pki/qemu"
default_tls_x509_verify = 1
```

If it is not already configured, modify `/etc/sysconfig/libvirtd` on both (ComputeNode1 & ComputeNode2) to listen for TCP/IP connections:

```
LIBVIRTD_ARGS="--listen"
```

Then, restart the libvirt daemon (also on both nodes):

```
$ systemctl restart libvirtd
```

Refer to the *Related information* section on a note about the other TLS-related configuration attributes in `/etc/libvirt/qemu.conf`.

Validating your TLS environment on compute nodes

Assuming you have two compute hosts (`ComputeNode1`, and `ComputeNode2`) run the **`virt-pki-validate`** tool (comes with the `libvirt-client` package on your Linux distribution) on both the nodes to ensure all the necessary PKI files are configured are configured:

```
[ComputeNode1]$ virt-pki-validate
Found /usr/bin/certtool
Found CA certificate /etc/pki/CA/cacert.pem for TLS Migration Test
```

(continues on next page)

(continued from previous page)

```
Found client certificate /etc/pki/libvirt/clientcert.pem for ComputeNode1
Found client private key /etc/pki/libvirt/private/clientkey.pem
Found server certificate /etc/pki/libvirt/servercert.pem for ComputeNode1
Found server private key /etc/pki/libvirt/private/serverkey.pem
Make sure /etc/sysconfig/libvirtd is setup to listen to
TCP/IP connections and restart the libvirtd service

[ComputeNode2]$ virt-pki-validate
Found /usr/bin/certtool
Found CA certificate /etc/pki/CA/cacert.pem for TLS Migration Test
Found client certificate /etc/pki/libvirt/clientcert.pem for ComputeNode2
Found client private key /etc/pki/libvirt/private/clientkey.pem
Found server certificate /etc/pki/libvirt/servercert.pem for ComputeNode2
Found server private key /etc/pki/libvirt/private/serverkey.pem
Make sure /etc/sysconfig/libvirtd is setup to listen to
TCP/IP connections and restart the libvirtd service
```

Other TLS environment related checks on compute nodes

IMPORTANT: Ensure that the permissions of certificate files and keys in `/etc/pki/qemu/*` directory on both source *and* destination compute nodes to be the following `0640` with `root:qemu` as the group/user. For example, on a Fedora-based system:

```
$ ls -lasrtZ /etc/pki/qemu
total 32
0 drwxr-xr-x. 10 root root system_u:object_r:cert_t:s0      110 Dec 10 10:39 ..
4 -rw-r-----.  1 root qemu unconfined_u:object_r:cert_t:s0 1464 Dec 10 11:08 ca-cert.
↪pem
4 -rw-r-----.  1 root qemu unconfined_u:object_r:cert_t:s0 1558 Dec 10 11:08 server-
↪cert.pem
4 -rw-r-----.  1 root qemu unconfined_u:object_r:cert_t:s0 1619 Dec 10 11:09 client-
↪cert.pem
8 -rw-r-----.  1 root qemu unconfined_u:object_r:cert_t:s0 8180 Dec 10 11:09 client-
↪key.pem
8 -rw-r-----.  1 root qemu unconfined_u:object_r:cert_t:s0 8177 Dec 11 05:35 server-
↪key.pem
0 drwxr-xr-x.  2 root root unconfined_u:object_r:cert_t:s0  146 Dec 11 06:01 .
```

Performing the migration

(1) On all relevant compute nodes, enable the *libvirt.live_migration_with_native_tls* configuration attribute and set the *libvirt.live_migration_scheme* configuration attribute to tls:

```
[libvirt]
live_migration_with_native_tls = true
live_migration_scheme = tls
```

Note: Setting both *libvirt.live_migration_with_native_tls* and *libvirt.live_migration_tunnelled* at the same time is invalid (and disallowed).

Note: Not setting *libvirt.live_migration_scheme* to `tls` will result in libvirt using the unencrypted

TCP connection without displaying any error or a warning in the logs.

And restart the `nova-compute` service:

```
$ systemctl restart openstack-nova-compute
```

(2) Now that all TLS-related configuration is in place, migrate guests (with or without shared storage) from `ComputeNode1` to `ComputeNode2`. Refer to the *Live-migrate instances* document on details about live migration.

Related information

- If you have the relevant libvirt and QEMU versions (mentioned in the *Prerequisites* section earlier), then using the *`libvirt.live_migration_with_native_tls`* is strongly recommended over the more limited *`libvirt.live_migration_tunnelled`* option, which is intended to be deprecated in future.
- There are in total *nine* TLS-related config options in `/etc/libvirt/qemu.conf`:

```
default_tls_x509_cert_dir
default_tls_x509_verify
nbd_tls
nbd_tls_x509_cert_dir
migrate_tls_x509_cert_dir

vnc_tls_x509_cert_dir
spice_tls_x509_cert_dir
vxhs_tls_x509_cert_dir
chardev_tls_x509_cert_dir
```

If you set both `default_tls_x509_cert_dir` and `default_tls_x509_verify` parameters for all certificates, there is no need to specify any of the other `*_tls*` config options.

The intention (of libvirt) is that you can just use the `default_tls_x509_*` config attributes so that you dont need to set any other `*_tls*` parameters, _unless_ you need different certificates for some services. The rationale for that is that some services (e.g. migration / NBD) are only exposed to internal infrastructure; while some sevices (VNC, Spice) might be exposed publically, so might need different certificates. For OpenStack this does not matter, though, we will stick with the defaults.

- If they are not already open, ensure you open up these TCP ports on your firewall: `16514` (where the authenticated and encrypted TCP/IP socket will be listening on) and `49152-49215` (for regular migration) on all relevant compute nodes. (Otherwise you get `error: internal error: unable to execute QEMU command 'drive-mirror': Failed to connect socket: No route to host`).

4.3.36 Mitigation for MDS (Microarchitectural Data Sampling) Security Flaws

Issue

In May 2019, four new microprocessor flaws, known as MDS , have been discovered. These flaws affect unpatched Nova compute nodes and instances running on Intel x86_64 CPUs. (The said MDS security flaws are also referred to as RIDL and Fallout or ZombieLoad).

Resolution

To get mitigation for the said MDS security flaws, a new CPU flag, *md-clear*, needs to be exposed to the Nova instances. It can be done as follows.

(1) Update the following components to the versions from your Linux distribution that have fixes for the MDS flaws, on all compute nodes with Intel x86_64 CPUs:

- microcode_ctl
- kernel
- qemu-system-x86
- libvirt

(2) When using the libvirt driver, ensure that the CPU flag `md-clear` is exposed to the Nova instances. It can be done so in one of the three following ways, given that Nova supports three distinct CPU modes:

a. *`libvirt.cpu_mode`*=host-model

When using `host-model` CPU mode, the `md-clear` CPU flag will be passed through to the Nova guests automatically.

This mode is the default, when *`libvirt.virt_type`*=kvm|qemu is set in `/etc/nova/nova-cpu.conf` on compute nodes.

b. *`libvirt.cpu_mode`*=host-passthrough

When using `host-passthrough` CPU mode, the `md-clear` CPU flag will be passed through to the Nova guests automatically.

c. Specific custom CPU models this can be enabled using the Nova config attributes *`libvirt.cpu_mode`*=custom plus particular named CPU models, e.g. *`libvirt.cpu_models`*=IvyBridge.

(The list of all valid named CPU models that are supported by your host, QEMU, and libvirt can be found by running the command `virsh domcapabilities`.)

When using a custom CPU mode, you must *explicitly* enable the CPU flag `md-clear` to the Nova instances, in addition to the flags required for previous vulnerabilities, using the *`libvirt.cpu_model_extra_flags`*. E.g.:

```
[libvirt]
cpu_mode = custom
cpu_models = IvyBridge
cpu_model_extra_flags = spec-ctrl,ssbd,md-clear
```

(3) Reboot the compute node for the fixes to take effect. (To minimize workload downtime, you may wish to live migrate all guests to another compute node first.)

Once the above steps have been taken on every vulnerable compute node in the deployment, each running guest in the cluster must be fully powered down, and cold-booted (i.e. an explicit stop followed by a start), in order to activate the new CPU models. This can be done by the guest administrators at a time of their choosing.

Validate that the fixes are in effect

After applying relevant updates, administrators can check the kernels sysfs interface to see what mitigation is in place, by running the following command (on the host):

```
# cat /sys/devices/system/cpu/vulnerabilities/mds
Mitigation: Clear CPU buffers; SMT vulnerable
```

To unpack the message Mitigation: Clear CPU buffers; SMT vulnerable:

- The `Mitigation: Clear CPU buffers` bit means, you have the CPU buffer clearing mitigation enabled (which is mechanism to invoke a flush of various exploitable CPU buffers by invoking a CPU instruction called VERW).
- The `SMT vulnerable` bit means, depending on your workload, you may still be vulnerable to SMT-related problems. You need to evaluate whether your workloads need SMT (also called Hyper-Threading) to be disabled or not. Refer to the guidance from your Linux distribution and processor vendor.

To see the other possible values for the sysfs file, `/sys/devices/system/cpu/vulnerabilities/mds`, refer to the MDS system information section in Linux kernels documentation for MDS.

On the host, validate that KVM is capable of exposing the `md-clear` flag to guests:

```
# virsh domcapabilities kvm | grep md-clear
    <feature policy='require' name='md-clear'/>
```

Also, refer to the Diagnosis tab in this security notice document here

Performance Impact

Refer to this section titled Performance Impact and Disabling MDS from the security notice document here, under the Resolve tab. (Note that although the article referred to is from Red Hat, the findings and recommendations about performance impact apply for other distributions as well.)

4.3.37 Vendordata

Note: This section provides deployment information about the vendordata feature. For end-user information about the vendordata feature and instance metadata in general, refer to the *user guide*.

The *vendordata* feature provides a way to pass vendor or deployment-specific information to instances. This can be accessed by users using *the metadata service* or with *config drives*.

There are two vendordata modules provided with nova: `StaticJSON` and `DynamicJSON`.

`StaticJSON`

The `StaticJSON` module includes the contents of a static JSON file loaded from disk. This can be used for things which dont change between instances, such as the location of the corporate puppet server. It is the default provider.

Configuration

The service you must configure to enable the `StaticJSON` vendordata module depends on how guests are accessing vendordata. If using the metadata service, configuration applies to either **`nova-api`** or **`nova-api-metadata`**, depending on the deployment, while if using config drives, configuration applies to **`nova-compute`**. However, configuration is otherwise the same and the following options apply:

- *`api.vendordata_providers`*
- *`api.vendordata_jsonfile_path`*

Refer to the *metadata service* and *config drive* documentation for more information on how to configure the required services.

DynamicJSON

The `DynamicJSON` module can make a request to an external REST service to determine what metadata to add to an instance. This is how we recommend you generate things like Active Directory tokens which change per instance.

When used, the `DynamicJSON` module will make a request to any REST services listed in the *api.vendordata_dynamic_targets* configuration option. There can be more than one of these but note that they will be queried once per metadata request from the instance which can mean a lot of traffic depending on your configuration and the configuration of the instance.

The following data is passed to your REST service as a JSON encoded POST:

Key	Description
`project-id`	The ID of the project that owns this instance.
`instance-id`	The UUID of this instance.
`image-id`	The ID of the image used to boot this instance.
`user-data`	As specified by the user at boot time.
`hostname`	The hostname of the instance.
`metadata`	As specified by the user at boot time.

Metadata fetched from the REST service will appear in the metadata service at a new file called `vendordata2.json`, with a path (either in the metadata service URL or in the config drive) like this:

```
openstack/latest/vendor_data2.json
```

For each dynamic target, there will be an entry in the JSON file named after that target. For example:

```
{
    "testing": {
        "value1": 1,
        "value2": 2,
        "value3": "three"
    }
}
```

The novajoin project provides a dynamic vendordata service to manage host instantiation in an IPA server.

Deployment considerations

Nova provides authentication to external metadata services in order to provide some level of certainty that the request came from nova. This is done by providing a service token with the request – you can then just deploy your metadata service with the keystone authentication WSGI middleware. This is configured using the keystone authentication parameters in the *vendordata_dynamic_auth* configuration group.

Configuration

As with `StaticJSON`, the service you must configure to enable the `DynamicJSON` vendordata module depends on how guests are accessing vendordata. If using the metadata service, configuration applies to either **`nova-api`** or **`nova-api-metadata`**, depending on the deployment, while if using config drives, configuration applies to **`nova-compute`**. However, configuration is otherwise the same and the following options apply:

- *api.vendordata_providers*
- *api.vendordata_dynamic_ssl_certfile*

- *api.vendordata_dynamic_connect_timeout*
- *api.vendordata_dynamic_read_timeout*
- *api.vendordata_dynamic_failure_fatal*
- *api.vendordata_dynamic_targets*

Refer to the *metadata service* and *config drive* documentation for more information on how to configure the required services.

In addition, there are also many options related to authentication. These are provided by keystone but are listed below for completeness:

- *vendordata_dynamic_auth.cafile*
- *vendordata_dynamic_auth.certfile*
- *vendordata_dynamic_auth.keyfile*
- *vendordata_dynamic_auth.insecure*
- *vendordata_dynamic_auth.timeout*
- *vendordata_dynamic_auth.collect_timing*
- *vendordata_dynamic_auth.split_loggers*
- *vendordata_dynamic_auth.auth_type*
- *vendordata_dynamic_auth.auth_section*
- *vendordata_dynamic_auth.auth_url*
- *vendordata_dynamic_auth.system_scope*
- *vendordata_dynamic_auth.domain_id*
- *vendordata_dynamic_auth.domain_name*
- *vendordata_dynamic_auth.project_id*
- *vendordata_dynamic_auth.project_name*
- *vendordata_dynamic_auth.project_domain_id*
- *vendordata_dynamic_auth.project_domain_name*
- *vendordata_dynamic_auth.trust_id*
- *vendordata_dynamic_auth.default_domain_id*
- *vendordata_dynamic_auth.default_domain_name*
- *vendordata_dynamic_auth.user_id*
- *vendordata_dynamic_auth.username*
- *vendordata_dynamic_auth.user_domain_id*
- *vendordata_dynamic_auth.user_domain_name*
- *vendordata_dynamic_auth.password*
- *vendordata_dynamic_auth.tenant_id*
- *vendordata_dynamic_auth.tenant_name*

Refer to the keystone documentation for information on configuring these.

References

- Michael Stills talk from the Queens summit in Sydney, Metadata, User Data, Vendor Data, oh my!
- Michaels blog post on deploying a simple vendordata service which provides more details and sample code to supplement the documentation above.

4.4 Command-line Utilities

In this section you will find information on Novas command line utilities.

4.4.1 Nova Management Commands

These commands are used to manage existing installations. They are designed to be run by operators in an environment where they have direct access to the nova database.

nova-manage

Control and manage cloud computer instances

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-manage <category> <action> [<args>]
```

Description

nova-manage controls cloud computing instances by managing various admin-only aspects of Nova.

Options

The standard pattern for executing a nova-manage command is:

```
nova-manage <category> <command> [<args>]
```

Run without arguments to see a list of available command categories:

```
nova-manage
```

You can also run with a category argument such as `db` to see a list of all commands in that category:

```
nova-manage db
```

These sections describe the available categories and arguments for nova-manage.

Nova Database

nova-manage db version Print the current main database version.

nova-manage db sync [--version <version>] [--local_cell] Upgrade the main database schema up to the most recent version or `--version` if specified. By default, this command will also attempt to upgrade the schema for the cell0 database if it is mapped (see the `map_cell0` or `simple_cell_setup` commands for more details on mapping the cell0 database). If `--local_cell` is specified, then only the main database in the current cell is upgraded. The local database connection is determined by *database.connection* in the configuration file passed to nova-manage. This command should be run after `nova-manage api_db sync`.

Returns exit code 0 if the database schema was synced successfully, or 1 if cell0 cannot be accessed.

nova-manage db archive_deleted_rows [--max_rows <number>] [--verbose] [--until-complete] [-
Move deleted rows from production tables to shadow tables. Note that the corresponding rows in the `instance_mappings`, `request_specs` and `instance_group_member` tables of the API database are purged when instance records are archived and thus, *api_database.connection* is required in the config file. Specifying `--verbose` will print the results of the archive operation for any tables that were changed. Specifying `--until-complete` will make the command run continuously until all deleted rows are archived. Use the `--max_rows` option, which defaults to 1000, as a batch size for each iteration (note that the purged API database table records are not included in this batch size). Specifying `--before` will archive only instances that were deleted before the *date* provided, and records in other tables related to those instances. Specifying `--purge` will cause a *full* DB purge to be completed after archival. If a date range is desired for the purge, then run `nova-manage db purge --before <date>` manually after archiving is complete. Specifying `--all-cells` will cause the process to run against all cell databases.

Return Codes

Return code	Description
0	Nothing was archived.
1	Some number of rows were archived.
2	Invalid value for `--max_rows`.
3	No connection to the API database could be established using *api_database.connection*.
4	Invalid value for `--before`.
255	An unexpected error occurred.

If automating, this should be run continuously while the result is 1, stopping at 0, or use the `--until-complete` option.

nova-manage db purge [--all] [--before <date>] [--verbose] [--all-cells] Delete rows from shadow tables. Specifying `--all` will delete all data from all shadow tables. Specifying `--before` will delete data from all shadow tables that is older than the *date* provided. Specifying `--verbose` will cause information to be printed about purged records. Specifying `--all-cells` will cause the purge to be applied against all cell databases. For `--all-cells` to work, the api database connection information must be configured. Returns exit code 0 if rows were deleted, 1 if required arguments are not provided, 2 if an invalid date is provided, 3 if no data was deleted, 4 if the list of cells cannot be obtained.

nova-manage db null_instance_uuid_scan [--delete] Lists and optionally deletes database records where instance_uuid is NULL.

nova-manage db online_data_migrations [--max-count] Perform data migration to update all live data.

`--max-count` controls the maximum number of objects to migrate in a given call. If not specified, migration will occur in batches of 50 until fully complete.

Returns exit code 0 if no (further) updates are possible, 1 if the `--max-count` option was used and some updates were completed successfully (even if others generated errors), 2 if some updates generated errors and no other migrations were able to take effect in the last batch attempted, or 127 if invalid input is provided (e.g. non-numeric max-count).

This command should be called after upgrading database schema and nova services on all controller nodes. If it exits with partial updates (exit status 1) it should be called again, even if some updates initially generated errors, because some updates may depend on others having completed. If it exits with status 2, intervention is required to resolve the issue causing remaining updates to fail. It should be considered successfully completed only when the exit status is 0.

For example:

```
$ nova-manage db online_data_migrations
Running batches of 50 until complete
2 rows matched query migrate_instances_add_request_spec, 0 migrated
2 rows matched query populate_queued_for_delete, 2 migrated
+-------------------------------------------+--------------+-----------+
|                 Migration                 | Total Needed | Completed |
+-------------------------------------------+--------------+-----------+
|         create_incomplete_consumers       |      0       |     0     |
|      migrate_instances_add_request_spec   |      2       |     0     |
|       migrate_quota_classes_to_api_db     |      0       |     0     |
|        migrate_quota_limits_to_api_db     |      0       |     0     |
|          migration_migrate_to_uuid        |      0       |     0     |
|     populate_missing_availability_zones   |      0       |     0     |
|          populate_queued_for_delete       |      2       |     2     |
|               populate_uuids              |      0       |     0     |
|     service_uuids_online_data_migration   |      0       |     0     |
+-------------------------------------------+--------------+-----------+
```

In the above example, the `migrate_instances_add_request_spec` migration found two candidate records but did not need to perform any kind of data migration for either of them. In the case of the `populate_queued_for_delete` migration, two candidate records were found which did require a data migration. Since `--max-count` defaults to 50 and only two records were migrated with no more candidates remaining, the command completed successfully with exit code 0.

nova-manage db ironic_flavor_migration [--all] [--host] [--node] [--resource_class] Perform the ironic flavor migration process against the database while services are offline. This is *not recommended* for most people. The ironic compute driver will do this online and as necessary if run normally. This routine is provided only for advanced users that may be skipping the 16.0.0 Pike release, never able to run services normally at the Pike level. Since this utility is for use when all services (including ironic) are down, you must pass the resource class set on your node(s) with the `--resource_class` parameter.

To migrate a specific host and node, provide the hostname and node uuid with `--host $hostname --node $uuid`. To migrate all instances on nodes managed by a single host, provide only `--host`. To iterate over all nodes in the system in a single pass, use `--all`. Note that this process is not lightweight, so it should not be run frequently without cause, although it is not harmful to do so. If you have multiple cellsv2 cells, you should run this once per cell with the corresponding cell config for each (i.e. this does not iterate cells automatically).

Note that this is not recommended unless you need to run this specific data migration offline, and it should be used with care as the work done is non-trivial. Running smaller and more targeted batches (such as specific nodes) is recommended.

--before <date> The date argument accepted by the `--before` option can be in any of several formats, including `YYYY-MM-DD [HH:mm[:ss]]` and the default format produced by the `date` command, e.g. `Fri May 24 09:20:11 CDT 2019`. Date strings containing spaces must be quoted appropriately. Some examples:

```
# Purge shadow table rows older than a specific date
nova-manage db purge --before 2015-10-21
# or
nova-manage db purge --before "Oct 21 2015"
# Times are also accepted
nova-manage db purge --before "2015-10-21 12:00"
```

Note that relative dates (such as `yesterday`) are not supported natively. The `date` command can be helpful here:

```
# Archive deleted rows more than one month old
nova-manage db archive_deleted_rows --before "$(date -d 'now - 1 month')"
```

Nova API Database

nova-manage api_db version Print the current API database version.

nova-manage api_db sync [VERSION] Upgrade the API database schema up to the most recent version or `[VERSION]` if specified. This command does not create the API database, it runs schema migration scripts. The API database connection is determined by *api_database.connection* in the configuration file passed to nova-manage.

In the 18.0.0 Rocky or 19.0.0 Stein release, this command will also upgrade the optional placement database if `[placement_database]/connection` is configured.

Returns exit code 0 if the database schema was synced successfully. This command should be run before `nova-manage db sync`.

Nova Cells v2

nova-manage cell_v2 simple_cell_setup [--transport-url <transport_url>] Setup a fresh cells v2 environment. If a `transport_url` is not specified, it will use the one defined by *transport_url* in the configuration file. Returns 0 if setup is completed (or has already been done), 1 if no hosts are reporting (and cannot be mapped) and 1 if the transport url is missing or invalid.

nova-manage cell_v2 map_cell0 [--database_connection <database_connection>] Create a cell mapping to the database connection for the cell0 database. If a database_connection is not specified, it will use the one defined by *database.connection* in the configuration file passed to nova-manage. The cell0 database is used for instances that have not been scheduled to any cell. This generally applies to instances that have encountered an error before they have been scheduled. Returns 0 if cell0 is created successfully or already setup.

nova-manage cell_v2 map_instances --cell_uuid <cell_uuid> [--max-count <max_count>] [--rese Map instances to the provided cell. Instances in the nova database will be queried from oldest to newest and mapped to the provided cell. A `--max-count` can be set on the number of instance to map in a single run. Repeated runs of the command will start from where the last run finished so it is not necessary to increase `--max-count` to finish. A `--reset` option can be passed which will reset the marker, thus making the command start from the beginning as opposed to the default behavior of starting from where the last run finished.

If `--max-count` is not specified, all instances in the cell will be mapped in batches of 50. If you have a large number of instances, consider specifying a custom value and run the command until it exits with 0.

Return Codes

Return code	Description
0	All instances have been mapped.
1	There are still instances to be mapped.
127	Invalid value for `--max-count`.
255	An unexpected error occurred.

nova-manage cell_v2 map_cell_and_hosts [--name <cell_name>] [--transport-url <transport_url>]
Create a cell mapping to the database connection and message queue transport url, and map hosts to that cell. The database connection comes from the *database.connection* defined in the configuration file passed to nova-manage. If a transport_url is not specified, it will use the one defined by *transport_url* in the configuration file. This command is idempotent (can be run multiple times), and the verbose option will print out the resulting cell mapping uuid. Returns 0 on successful completion, and 1 if the transport url is missing or invalid.

nova-manage cell_v2 verify_instance --uuid <instance_uuid> [--quiet] Verify instance mapping to a cell. This command is useful to determine if the cells v2 environment is properly setup, specifically in terms of the cell, host, and instance mapping records required. Returns 0 when the instance is successfully mapped to a cell, 1 if the instance is not mapped to a cell (see the `map_instances` command), 2 if the cell mapping is missing (see the `map_cell_and_hosts` command if you are upgrading from a cells v1 environment, and the `simple_cell_setup` if you are upgrading from a non-cells v1 environment), 3 if it is a deleted instance which has instance mapping, and 4 if it is an archived instance which still has an instance mapping.

nova-manage cell_v2 create_cell [--name <cell_name>] [--transport-url <transport_url>] [--
Create a cell mapping to the database connection and message queue transport url. If a database_connection is not specified, it will use the one defined by *database.connection* in the configuration file passed to nova-manage. If a transport_url is not specified, it will use the one defined by *transport_url* in the configuration file. The verbose option will print out the resulting cell mapping uuid. All the cells created are by default enabled. However passing the `--disabled` option can create a pre-disabled cell, meaning no scheduling will happen to this cell. The meaning of the various exit codes returned by this command are explained below:

- Returns 0 if the cell mapping was successfully created.
- Returns 1 if the transport url or database connection was missing or invalid.
- Returns 2 if another cell is already using that transport url and/or database connection combination.

nova-manage cell_v2 discover_hosts [--cell_uuid <cell_uuid>] [--verbose] [--strict] [--by-s
Searches cells, or a single cell, and maps found hosts. This command will check the database for each cell (or a single one if passed in) and map any hosts which are not currently mapped. If a host is already mapped, nothing will be done. You need to re-run this command each time you add a batch of compute hosts to a cell (otherwise the scheduler will never place instances there and the API will not list the new hosts). If `--strict` is specified, the command will only return 0 if an unmapped host was discovered and mapped successfully. If `--by-service` is specified, this command will look in the appropriate cell(s) for any nova-compute services and ensure there are host mappings for them. This is less efficient and is only necessary when using compute drivers that may manage zero or more actual compute nodes at any given time (currently only ironic).

This command should be run once after all compute hosts have been deployed and should not be run in parallel. When run in parallel, the commands will collide with each other trying to map the same hosts in the database at the same time.

The meaning of the various exit codes returned by this command are explained below:

- Returns 0 if hosts were successfully mapped or no hosts needed to be mapped. If `--strict` is specified, returns 0 only if an unmapped host was discovered and mapped.

- Returns 1 if `--strict` is specified and no unmapped hosts were found. Also returns 1 if an exception was raised while running.
- Returns 2 if the command aborted because of a duplicate host mapping found. This means the command collided with another running discover_hosts command or scheduler periodic task and is safe to retry.

nova-manage cell_v2 list_cells [--verbose] By default the cell name, uuid, disabled state, masked transport URL and database connection details are shown. Use the `--verbose` option to see transport URL and database connection with their sensitive details.

nova-manage cell_v2 delete_cell [--force] --cell_uuid <cell_uuid> Delete a cell by the given uuid. Returns 0 if the empty cell is found and deleted successfully or the cell that has hosts is found and the cell, hosts and the instance_mappings are deleted successfully with `--force` option (this happens if there are no living instances), 1 if a cell with that uuid could not be found, 2 if host mappings were found for the cell (cell not empty) without `--force` option, 3 if there are instances mapped to the cell (cell not empty) irrespective of the `--force` option, and 4 if there are instance mappings to the cell but all instances have been deleted in the cell, again without the `--force` option.

nova-manage cell_v2 list_hosts [--cell_uuid <cell_uuid>] Lists the hosts in one or all v2 cells. By default hosts in all v2 cells are listed. Use the `--cell_uuid` option to list hosts in a specific cell. If the cell is not found by uuid, this command will return an exit code of 1. Otherwise, the exit code will be 0.

nova-manage cell_v2 update_cell --cell_uuid <cell_uuid> [--name <cell_name>] [--transport-u
Updates the properties of a cell by the given uuid. If a database_connection is not specified, it will attempt to use the one defined by *database.connection* in the configuration file. If a transport_url is not specified, it will attempt to use the one defined by *transport_url* in the configuration file. The meaning of the various exit codes returned by this command are explained below:

- If successful, it will return 0.
- If the cell is not found by the provided uuid, it will return 1.
- If the properties cannot be set, it will return 2.
- If the provided transport_url or/and database_connection is/are same as another cell, it will return 3.
- If an attempt is made to disable and enable a cell at the same time, it will return 4.
- If an attempt is made to disable or enable cell0 it will return 5.

Note: Updating the `transport_url` or `database_connection` fields on a running system will NOT result in all nodes immediately using the new values. Use caution when changing these values.

The scheduler will not notice that a cell has been enabled/disabled until it is restarted or sent the SIGHUP signal.

---

nova-manage cell_v2 delete_host --cell_uuid <cell_uuid> --host <host> Delete a host by the given host name and the given cell uuid. Returns 0 if the empty host is found and deleted successfully, 1 if a cell with that uuid could not be found, 2 if a host with that name could not be found, 3 if a host with that name is not in a cell with that uuid, 4 if a host with that name has instances (host not empty).

Note: The scheduler caches host-to-cell mapping information so when deleting a host the scheduler may need to be restarted or sent the SIGHUP signal.

Placement

nova-manage placement heal_allocations [--max-count <max_count>] [--verbose] [--skip-port-a

Iterates over non-cell0 cells looking for instances which do not have allocations in the Placement service and which are not undergoing a task state transition. For each instance found, allocations are created against the compute node resource provider for that instance based on the flavor associated with the instance.

Also if the instance has any port attached that has resource request (e.g. Quality of Service (QoS): Guaranteed Bandwidth) but the corresponding allocation is not found then the allocation is created against the network device resource providers according to the resource request of that port. It is possible that the missing allocation cannot be created either due to not having enough resource inventory on the host the instance resides on or because more than one resource provider could fulfill the request. In this case the instance needs to be manually deleted or the port needs to be detached. When nova supports migrating instances with guaranteed bandwidth ports, migration will heal missing allocations for these instances.

Before the allocations for the ports are persisted in placement nova-manage tries to update each port in neutron to refer to the resource provider UUID which provides the requested resources. If any of the port updates fail in neutron or the allocation update fails in placement the command tries to roll back the partial updates to the ports. If the roll back fails then the process stops with exit code `7` and the admin needs to do the rollback in neutron manually according to the description in the exit code section.

There is also a special case handled for instances that *do* have allocations created before Placement API microversion 1.8 where project_id and user_id values were required. For those types of allocations, the project_id and user_id are updated using the values from the instance.

Specify `--max-count` to control the maximum number of instances to process. If not specified, all instances in each cell will be mapped in batches of 50. If you have a large number of instances, consider specifying a custom value and run the command until it exits with 0 or 4.

Specify `--verbose` to get detailed progress output during execution.

Specify `--dry-run` to print output but not commit any changes. The return code should be 4. *(Since 20.0.0 Train)*

Specify `--instance` to process a specific instance given its UUID. If specified the `--max-count` option has no effect. *(Since 20.0.0 Train)*

Specify `--skip-port-allocations` to skip the healing of the resource allocations of bound ports, e.g. healing bandwidth resource allocation for ports having minimum QoS policy rules attached. If your deployment does not use such a feature then the performance impact of querying neutron ports for each instance can be avoided with this flag. *(Since 20.0.0 Train)*

This command requires that the *`api_database.connection`* and *`placement`* configuration options are set. Placement API >= 1.28 is required.

Return Codes

Return code	Description
0	Command completed successfully and allocations were created.
1	`--max-count` was reached and there are more instances to process.
2	Unable to find a compute node record for a given instance.
3	Unable to create (or update) allocations for an instance against its compute node resource provider.
4	Command completed successfully but no allocations were created.
5	Unable to query ports from neutron
6	Unable to update ports in neutron
7	Cannot roll back neutron port updates. Manual steps needed. The error message will indicate which neutron ports need to be changed to clean up `binding:profile` of the port: `$ openstack port unset <port_uuid> --binding-profile allocation`
127	Invalid input.
255	An unexpected error occurred.

`nova-manage placement sync_aggregates [--verbose]` Mirrors compute host aggregates to resource provider aggregates in the Placement service. Requires the *`api_database`* and *`placement`* sections of the nova configuration file to be populated.

Specify `--verbose` to get detailed progress output during execution.

Note: Depending on the size of your deployment and the number of compute hosts in aggregates, this command could cause a non-negligible amount of traffic to the placement service and therefore is recommended to be run during maintenance windows.

New in version Rocky.

Return Codes

Return code	Description
0	Successful run
1	A host was found with more than one matching compute node record
2	An unexpected error occurred while working with the placement API
3	Failed updating provider aggregates in placement
4	Host mappings not found for one or more host aggregate members
5	Compute node records not found for one or more hosts
6	Resource provider not found by uuid for a given host
255	An unexpected error occurred.

See Also

- OpenStack Nova

Bugs

- Nova bugs are managed at Launchpad

nova-status

CLI interface for nova status commands

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-status <category> <action> [<args>]
```

Description

nova-status is a tool that provides routines for checking the status of a Nova deployment.

Options

The standard pattern for executing a **nova-status** command is:

```
nova-status <category> <command> [<args>]
```

Run without arguments to see a list of available command categories:

```
nova-status
```

Categories are:

- `upgrade`

Detailed descriptions are below.

You can also run with a category argument such as `upgrade` to see a list of all commands in that category:

```
nova-status upgrade
```

These sections describe the available categories and arguments for **nova-status**.

Upgrade

nova-status upgrade check Performs a release-specific readiness check before restarting services with new code. This command expects to have complete configuration and access to databases and services within a cell. For example, this check may query the Nova API database and one or more cell databases. It may also make requests to other services such as the Placement REST API via the Keystone service catalog.

Return Codes

Return code	Description
0	All upgrade readiness checks passed successfully and there is nothing to do.
1	At least one check encountered an issue and requires further investigation. This is considered a warning but the upgrade may be OK.
2	There was an upgrade status check failure that needs to be investigated. This should be considered something that stops an upgrade.
255	An unexpected error occurred.

History of Checks

15.0.0 (Ocata)

- Checks are added for cells v2 so `nova-status upgrade check` should be run *after* running the `nova-manage cell_v2 simple_cell_setup` command.
- Checks are added for the Placement API such that there is an endpoint in the Keystone service catalog, the service is running and the check can make a successful request to the endpoint. The command also checks to see that there are compute node resource providers checking in with the Placement service. More information on the Placement service can be found at Placement API.

16.0.0 (Pike)

- Checks for the Placement API are modified to require version 1.4, that is needed in Pike and further for nova-scheduler to work correctly.

17.0.0 (Queens)

- Checks for the Placement API are modified to require version 1.17.

18.0.0 (Rocky)

- Checks for the Placement API are modified to require version 1.28.
- Checks that ironic instances have had their embedded flavors migrated to use custom resource classes.
- Checks for `nova-osapi_compute` service versions that are less than 15 across all cell mappings which might cause issues when querying instances depending on how the **nova-api** service is configured. See https://bugs.launchpad.net/nova/+bug/1759316 for details.
- Checks that existing instances have been migrated to have a matching request spec in the API DB.

19.0.0 (Stein)

- Checks for the Placement API are modified to require version 1.30.
- Checks are added for the **nova-consoleauth** service to warn and provide additional instructions to set **[workarounds]enable_consoleauth = True** while performing a live/rolling upgrade.
- The Resource Providers upgrade check was removed since the placement service code is being extracted from nova and the related tables are no longer used in the `nova_api` database.
- The API Service Version upgrade check was removed since the corresponding code for that check was removed in Stein.

20.0.0 (Train)

- Checks for the Placement API are modified to require version 1.32.
- Checks to ensure block-storage (cinder) API version 3.44 is available in order to support multi-attach volumes. If `[cinder]/auth_type` is not configured this is a no-op check.
- The **nova-consoleauth** service upgrade check was removed since the service was removed in Train.
- The `Request Spec Migration` check was removed.

See Also

- OpenStack Nova

Bugs

- Nova bugs are managed at Launchpad

4.4.2 Service Daemons

The service daemons make up a functioning nova environment. All of these are expected to be started by an init system, expect to read a nova.conf file, and daemonize correctly after starting up.

nova-api

Server for the OpenStack Compute APIs

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-api  [options]
```

Description

nova-api is a server daemon that serves the metadata and compute APIs in separate greenthreads.

Options

General options

Files

- `/etc/nova/nova.conf`
- `/etc/nova/api-paste.ini`
- `/etc/nova/policy.json`
- `/etc/nova/rootwrap.conf`
- `/etc/nova/rootwrap.d/`

See Also

- OpenStack Nova
- Using WSGI with Nova

Bugs

- Nova bugs are managed at Launchpad

nova-compute

Nova Compute Server

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-compute [options]
```

Description

nova-compute is a server daemon that serves the Nova Compute service, which is responsible for building a disk image, launching an instance via the underlying virtualization driver, responding to calls to check the instances state, attaching persistent storage, and terminating the instance.

Options

General options

Files

- `/etc/nova/nova.conf`
- `/etc/nova/policy.json`
- `/etc/nova/rootwrap.conf`
- `/etc/nova/rootwrap.d/`

See Also

- OpenStack Nova

Bugs

- Nova bugs are managed at Launchpad

nova-conductor

Server for the Nova Conductor

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-conductor [options]
```

Description

nova-conductor is a server daemon that serves the Nova Conductor service, which provides coordination and database query support for nova.

Options

General options

Files

- `/etc/nova/nova.conf`

See Also

- OpenStack Nova

Bugs

- Nova bugs are managed at Launchpad

nova-novncproxy

Websocket novnc Proxy for OpenStack Nova noVNC consoles

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-novncproxy [options]
```

Description

nova-novncproxy is a server daemon that serves the Nova noVNC Websocket Proxy service, which provides a websocket proxy that is compatible with OpenStack Nova noVNC consoles.

Options

General options

Files

- `/etc/nova/nova.conf`
- `/etc/nova/policy.json`
- `/etc/nova/rootwrap.conf`
- `/etc/nova/rootwrap.d/`

See Also

- OpenStack Nova

Bugs

- Nova bugs are managed at Launchpad

nova-scheduler

Nova Scheduler

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-scheduler [options]
```

Description

nova-scheduler is a server daemon that serves the Nova Scheduler service, which is responsible for picking a compute node to run a given instance on.

Options

General options

Files

- /etc/nova/nova.conf
- /etc/nova/policy.json
- /etc/nova/rootwrap.conf
- /etc/nova/rootwrap.d/

See Also

- OpenStack Nova

Bugs

- Nova bugs are managed at Launchpad

nova-serialproxy

Websocket serial Proxy for OpenStack Nova serial ports

Author openstack@lists.launchpad.net

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-serialproxy [options]
```

Description

nova-serialproxy is a server daemon that serves the Nova Serial Websocket Proxy service, which provides a websocket proxy that is compatible with OpenStack Nova serial ports.

Options

General options

Files

- `/etc/nova/nova.conf`
- `/etc/nova/policy.json`
- `/etc/nova/rootwrap.conf`
- `/etc/nova/rootwrap.d/`

See Also

- OpenStack Nova

Bugs

- Nova bugs are managed at Launchpad

nova-spicehtml5proxy

Websocket Proxy for OpenStack Nova SPICE HTML5 consoles

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-spicehtml5proxy [options]
```

Description

nova-spicehtml5proxy is a server daemon that serves the Nova SPICE HTML5 Websocket Proxy service, which provides a websocket proxy that is compatible with OpenStack Nova SPICE HTML5 consoles.

Options

General options

Files

- `/etc/nova/nova.conf`
- `/etc/nova/policy.json`
- `/etc/nova/rootwrap.conf`
- `/etc/nova/rootwrap.d/`

See Also

- OpenStack Nova

Bugs

- Nova bugs are managed at Launchpad

4.4.3 WSGI Services

Starting in the Pike release, the preferred way to deploy the nova api is in a wsgi container (uwsgi or apache/mod_wsgi). These are the wsgi entry points to do that.

nova-api-metadata

Server for the Nova Metadata API

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-api-metadata [options]
```

Description

nova-api-metadata is a server daemon that serves the Nova Metadata API. This daemon routes database requests via the `nova-conductor` service, so there are some considerations about using this in a *multi-cell layout*.

Options

General options

Files

- `/etc/nova/nova.conf`
- `/etc/nova/api-paste.ini`
- `/etc/nova/policy.json`
- `/etc/nova/rootwrap.conf`
- `/etc/nova/rootwrap.d/`

See Also

- OpenStack Nova
- Using WSGI with Nova

Bugs

- Nova bugs are managed at Launchpad

nova-api-os-compute

Server for the Nova OpenStack Compute APIs

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-api-os-compute  [options]
```

Description

nova-api-os-compute is a server daemon that serves the Nova OpenStack Compute API.

Options

General options

Files

- `/etc/nova/nova.conf`
- `/etc/nova/api-paste.ini`
- `/etc/nova/policy.json`
- `/etc/nova/rootwrap.conf`
- `/etc/nova/rootwrap.d/`

See Also

- OpenStack Nova
- Using WSGI with Nova

Bugs

- Nova bugs are managed at Launchpad

4.4.4 Additional Tools

There are a few additional cli tools which nova services call when appropriate. This should not need to be called directly by operators, but they are documented for completeness and debugging if something goes wrong.

nova-rootwrap

Root wrapper for Nova

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-rootwrap [options]
```

Description

nova-rootwrap is an application that filters which commands nova is allowed to run as another user.

To use this, you should set the following in `nova.conf`:

```
rootwrap_config=/etc/nova/rootwrap.conf
```

You also need to let the nova user run **nova-rootwrap** as root in `sudoers`:

```
nova ALL = (root) NOPASSWD: /usr/bin/nova-rootwrap /etc/nova/rootwrap.conf *
```

To make allowed commands node-specific, your packaging should only install `{compute,network}.filters` respectively on compute and network nodes, i.e. **nova-api** nodes should not have any of those files installed.

Note: **nova-rootwrap** is being slowly deprecated and replaced by `oslo.privsep`, and will eventually be removed.

Options

General options

Files

- `/etc/nova/nova.conf`
- `/etc/nova/rootwrap.conf`
- `/etc/nova/rootwrap.d/`

See Also

- OpenStack Nova

Bugs

- Nova bugs are managed at Launchpad

4.4.5 Deprecated Services

The following services are deprecated in nova. They should not be used in new deployments, but are documented for existing ones.

nova-dhcpbridge

Handles Lease Database updates from DHCP servers

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-dhcpbridge [options]
```

Description

nova-dhcpbridge is an application that handles lease database updates from DHCP servers. **nova-dhcpbridge** is used whenever nova is managing DHCP (vlan and flatDHCP). **nova-dhcpbridge** should not be run as a daemon.

> **Warning:** This application is only for use with `nova-network`, which is not recommended for new deployments.

Options

General options

Files

- `/etc/nova/nova.conf`
- `/etc/nova/api-paste.ini`
- `/etc/nova/policy.json`
- `/etc/nova/rootwrap.conf`
- `/etc/nova/rootwrap.d/`

See Also

- OpenStack Nova

Bugs

- Nova bugs are managed at Launchpad

nova-network

Nova Network Server

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-network [options]
```

Description

nova-network is a server daemon that serves the Nova Network service, which is responsible for allocating IPs and setting up the network

Deprecated since version 14.0.0: **nova-network** is deprecated and will be removed in an upcoming release. Use *neutron* or another networking solution instead.

Options

General options

Files

- `/etc/nova/nova.conf`
- `/etc/nova/policy.json`
- `/etc/nova/rootwrap.conf`
- `/etc/nova/rootwrap.d/`

See Also

- OpenStack Nova

Bugs

- Nova bugs are managed at Launchpad

nova-console

Nova Console Server

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-console [options]
```

Description

nova-console is a server daemon that serves the Nova Console service, which is a console proxy to set up multi-tenant VM console access, e.g. with *XVP*.

Deprecated since version 19.0.0: **nova-console** is deprecated since 19.0.0 (Stein) and will be removed in an upcoming release.

Options

General options

Files

- `/etc/nova/nova.conf`
- `/etc/nova/policy.json`
- `/etc/nova/rootwrap.conf`
- `/etc/nova/rootwrap.d/`

See Also

- OpenStack Nova

Bugs

- Nova bugs are managed at Launchpad

nova-xvpvncproxy

XVP VNC Console Proxy Server

Author openstack@lists.openstack.org

Copyright OpenStack Foundation

Manual section 1

Manual group cloud computing

Synopsis

```
nova-xvpvncproxy  [options]
```

Description

nova-xvpvncproxy is a server daemon that serves the Nova XVP VNC Console Proxy service, which provides an XVP-based VNC Console Proxy for use with the Xen hypervisor.

Deprecated since version 19.0.0: **nova-xvpvnxproxy** is deprecated since 19.0.0 (Stein) and will be removed in an upcoming release.

Options

General options

Files

- `/etc/nova/nova.conf`
- `/etc/nova/policy.json`
- `/etc/nova/rootwrap.conf`
- `/etc/nova/rootwrap.d/`

See Also

- OpenStack Nova

Bugs

- Nova bugs are managed at Launchpad

4.5 Configuration Guide

The static configuration for nova lives in two main files: `nova.conf` and `policy.json`. These are described below. For a bigger picture view on configuring nova to solve specific problems, refer to the *Nova Admin Guide*.

4.5.1 Configuration

- *Configuration Guide*: Detailed configuration guides for various parts of your Nova system. Helpful reference for setting up specific hypervisor backends.
- *Config Reference*: A complete reference of all configuration options available in the `nova.conf` file.

4.5.2 Policy

Nova, like most OpenStack projects, uses a policy language to restrict permissions on REST API actions.

- *Policy Reference*: A complete reference of all policy points in nova and what they impact.

Configuration Options

The following is an overview of all available configuration options in Nova.

DEFAULT

run_external_periodic_tasks

Type boolean

Default `true`

Some periodic tasks can be run in a separate process. Should we run them here?

backdoor_port

Type string

Default `<None>`

Enable eventlet backdoor. Acceptable values are 0, <port>, and <start>:<end>, where 0 results in listening on a random tcp port number; <port> results in listening on the specified port number (and not enabling backdoor if that port is in use); and <start>:<end> results in listening on the smallest unused port number within the specified range of port numbers. The chosen port is displayed in the services log file.

backdoor_socket

Type string

Default `<None>`

Enable eventlet backdoor, using the provided path as a unix socket that can receive connections. This option is mutually exclusive with backdoor_port in that only one should be provided. If both are provided then the existence of this option overrides the usage of that option. Inside the path {pid} will be replaced with the PID of the current process.

log_options

Type boolean

Default `true`

Enables or disables logging values of all registered options when starting a service (at DEBUG level).

graceful_shutdown_timeout

Type integer

Default `60`

Specify a timeout after which a gracefully shutdown server will exit. Zero value means endless wait.

rpc_conn_pool_size

Type integer

Default `30`

Minimum Value 1

Size of RPC connection pool.

Table 11: Deprecated Variations

Group	Name
DEFAULT	rpc_conn_pool_size

conn_pool_min_size

Type integer

Default `2`

The pool size limit for connections expiration policy

conn_pool_ttl

Type integer

Default `1200`

The time-to-live in sec of idle connections in the pool

executor_thread_pool_size

Type integer

Default `64`

Size of executor thread pool when executor is threading or eventlet.

Table 12: Deprecated Variations

Group	Name
DEFAULT	rpc_thread_pool_size

rpc_response_timeout

Type integer

Default `60`

Seconds to wait for a response from a call.

transport_url

Type string

Default `rabbit://`

The network address and optional user credentials for connecting to the messaging backend, in URL format. The expected format is:

driver://[user:pass@]host:port[,[userN:passN@]hostN:portN]/virtual_host?query

Example: rabbit://rabbitmq:password@127.0.0.1:5672//

For full details on the fields in the URL see the documentation of oslo_messaging.TransportURL at https://docs.openstack.org/oslo.messaging/latest/reference/transport.html

control_exchange

Type string

Default `openstack`

The default exchange under which topics are scoped. May be overridden by an exchange name specified in the transport_url option.

debug

Type boolean

Default `false`

Mutable This option can be changed without restarting.

If set to true, the logging level will be set to DEBUG instead of the default INFO level.

log_config_append

Type string

Default `<None>`

Mutable This option can be changed without restarting.

The name of a logging configuration file. This file is appended to any existing logging configuration files. For details about logging configuration files, see the Python logging module documentation. Note that when logging configuration files are used then all logging configuration is set in the configuration file and other logging configuration options are ignored (for example, log-date-format).

Table 13: Deprecated Variations

Group	Name
DEFAULT	log-config
DEFAULT	log_config

log_date_format

Type string

Default `%Y-%m-%d %H:%M:%S`

Defines the format string for %(asctime)s in log records. Default: the value above . This option is ignored if log_config_append is set.

log_file

Type string

Default `<None>`

(Optional) Name of log file to send logging output to. If no default is set, logging will go to stderr as defined by use_stderr. This option is ignored if log_config_append is set.

Table 14: Deprecated Variations

Group	Name
DEFAULT	logfile

log_dir

Type string

Default `<None>`

(Optional) The base directory used for relative log_file paths. This option is ignored if log_config_append is set.

Table 15: Deprecated Variations

Group	Name
DEFAULT	logdir

watch_log_file

Type boolean

Default `false`

Uses logging handler designed to watch file system. When log file is moved or removed this handler will open a new log file with specified path instantaneously. It makes sense only if log_file option is specified and Linux platform is used. This option is ignored if log_config_append is set.

use_syslog

Type boolean

Default `false`

Use syslog for logging. Existing syslog format is DEPRECATED and will be changed later to honor RFC5424. This option is ignored if log_config_append is set.

use_journal

Type boolean

Default `false`

Enable journald for logging. If running in a systemd environment you may wish to enable journal support. Doing so will use the journal native protocol which includes structured metadata in addition to log messages.This option is ignored if log_config_append is set.

syslog_log_facility

Type string

Default `LOG_USER`

Syslog facility to receive log lines. This option is ignored if log_config_append is set.

use_json

Type boolean

Default `false`

Use JSON formatting for logging. This option is ignored if log_config_append is set.

use_stderr

Type boolean

Default `false`

Log output to standard error. This option is ignored if log_config_append is set.

use_eventlog

Type boolean

Default `false`

Log output to Windows Event Log.

log_rotate_interval

Type integer

Default `1`

The amount of time before the log files are rotated. This option is ignored unless log_rotation_type is setto interval.

log_rotate_interval_type

Type string

Default `days`

Valid Values Seconds, Minutes, Hours, Days, Weekday, Midnight

Rotation interval type. The time of the last file change (or the time when the service was started) is used when scheduling the next rotation.

max_logfile_count

Type integer

Default `30`

Maximum number of rotated log files.

max_logfile_size_mb

Type integer

Default `200`

Log file maximum size in MB. This option is ignored if log_rotation_type is not set to size.

log_rotation_type

Type string

Default `none`

Valid Values interval, size, none

Log rotation type.

Possible values

interval Rotate logs at predefined time intervals.

size Rotate logs once they reach a predefined size.

none Do not rotate log files.

logging_context_format_string

Type string

Default `%(asctime)s.%(msecs)03d %(process)d %(levelname)s %(name)s [%(request_id)s %(user_identity)s] %(instance)s%(message)s`

Format string to use for log messages with context. Used by oslo_log.formatters.ContextFormatter

logging_default_format_string

Type string

Default `%(asctime)s.%(msecs)03d %(process)d %(levelname)s %(name)s [-] %(instance)s%(message)s`

Format string to use for log messages when context is undefined. Used by oslo_log.formatters.ContextFormatter

logging_debug_format_suffix

Type string

Default `%(funcName)s %(pathname)s:%(lineno)d`

Additional data to append to log message when logging level for the message is DEBUG. Used by oslo_log.formatters.ContextFormatter

logging_exception_prefix

Type string

Default `%(asctime)s.%(msecs)03d %(process)d ERROR %(name)s %(instance)s`

Prefix each line of exception output with this format. Used by oslo_log.formatters.ContextFormatter

logging_user_identity_format

Type string

Default `%(user)s %(tenant)s %(domain)s %(user_domain)s %(project_domain)s`

Defines the format string for %(user_identity)s that is used in logging_context_format_string. Used by oslo_log.formatters.ContextFormatter

default_log_levels

Type list

Default `amqp=WARN,amqplib=WARN,boto=WARN,qpid=WARN,sqlalchemy=WARN,suds=INFO,oslo.messaging=INFO,oslo_messaging=INFO,iso8601=WARN,requests.packages.urllib3.connectionpool=WARN,urllib3.connectionpool=WARN,websocket=WARN,requests.packages.urllib3.util.retry=WARN,urllib3.util.retry=WARN,keystonemiddleware=WARN,routes.middleware=WARN,stevedore=WARN,taskflow=WARN,keystoneauth=WARN,oslo.cache=INFO,oslo_policy=INFO,dogpile.core.dogpile=INFO`

List of package logging levels in logger=LEVEL pairs. This option is ignored if log_config_append is set.

publish_errors

Type boolean

Default `false`

Enables or disables publication of error events.

instance_format

Type string

Default `"[instance: %(uuid)s] "`

The format for an instance that is passed with the log message.

instance_uuid_format

Type string

Default `"[instance: %(uuid)s] "`

The format for an instance UUID that is passed with the log message.

rate_limit_interval

Type integer

Default `0`

Interval, number of seconds, of log rate limiting.

rate_limit_burst

Type integer

Default `0`

Maximum number of logged messages per rate_limit_interval.

rate_limit_except_level

Type string

Default `CRITICAL`

Log level name used by rate limiting: CRITICAL, ERROR, INFO, WARNING, DEBUG or empty string. Logs with level greater or equal to rate_limit_except_level are not filtered. An empty string means that all levels are filtered.

fatal_deprecations

Type boolean

Default `false`

Enables or disables fatal status of deprecations.

internal_service_availability_zone

Type string

Default `internal`

Availability zone for internal services.

This option determines the availability zone for the various internal nova services, such as nova-scheduler, nova-conductor, etc.

Possible values:

- Any string representing an existing availability zone name.

default_availability_zone

Type string

Default `nova`

Default availability zone for compute services.

This option determines the default availability zone for nova-compute services, which will be used if the service(s) do not belong to aggregates with availability zone metadata.

Possible values:

- Any string representing an existing availability zone name.

default_schedule_zone

Type string

Default `<None>`

Default availability zone for instances.

This option determines the default availability zone for instances, which will be used when a user does not specify one when creating an instance. The instance(s) will be bound to this availability zone for their lifetime.

Possible values:

- Any string representing an existing availability zone name.
- None, which means that the instance can move from one availability zone to another during its lifetime if it is moved from one compute node to another.

password_length

Type integer

Default `12`

Minimum Value 0

Length of generated instance admin passwords.

instance_usage_audit_period

Type string

Default `month`

Time period to generate instance usages for. It is possible to define optional offset to given period by appending @ character followed by a number defining offset.

Possible values:

- period, example: `hour`, `day`, `month` or ``year`
- period with offset, example: `month@15` will result in monthly audits starting on 15th day of month.

use_rootwrap_daemon

Type boolean

Default `false`

Start and use a daemon that can run the commands that need to be run with root privileges. This option is usually enabled on nodes that run nova compute processes.

rootwrap_config

Type string

Default `/etc/nova/rootwrap.conf`

Path to the rootwrap configuration file.

Goal of the root wrapper is to allow a service-specific unprivileged user to run a number of actions as the root user in the safest manner possible. The configuration file used here must match the one defined in the sudoers entry.

tempdir

Type string

Default `<None>`

Explicitly specify the temporary working directory.

compute_driver

Type string

Default `<None>`

Defines which driver to use for controlling virtualization.

Possible values:

- `libvirt.LibvirtDriver`
- `xenapi.XenAPIDriver`
- `fake.FakeDriver`
- `ironic.IronicDriver`
- `vmwareapi.VMwareVCDriver`
- `hyperv.HyperVDriver`
- `powervm.PowerVMDriver`
- `zvm.ZVMDriver`

`allow_resize_to_same_host`

Type boolean

Default `false`

Allow destination machine to match source for resize. Useful when testing in single-host environments. By default it is not allowed to resize to the same host. Setting this option to true will add the same host to the destination options. Also set to true if you allow the ServerGroupAffinityFilter and need to resize.

`non_inheritable_image_properties`

Type list

Default `cache_in_nova,bittorrent,img_signature_hash_method, img_signature,img_signature_key_type,img_signature_certificate_uuid`

Image properties that should not be inherited from the instance when taking a snapshot.

This option gives an opportunity to select which image-properties should not be inherited by newly created snapshots.

Possible values:

- A comma-separated list whose item is an image property. Usually only the image properties that are only needed by base images can be included here, since the snapshots that are created from the base images dont need them.
- **Default list: cache_in_nova, bittorrent, img_signature_hash_method,** img_signature, img_signature_key_type, img_signature_certificate_uuid

`max_local_block_devices`

Type integer

Default `3`

Maximum number of devices that will result in a local image being created on the hypervisor node.

A negative number means unlimited. Setting `max_local_block_devices` to 0 means that any request that attempts to create a local disk will fail. This option is meant to limit the number of local discs (so root local disc that is the result of `imageRef` being used when creating a server, and any other ephemeral and swap disks). 0 does not mean that images will be automatically converted to volumes and boot instances from volumes - it just means that all requests that attempt to create a local disk will fail.

Possible values:

- 0: Creating a local disk is not allowed.
- Negative number: Allows unlimited number of local discs.
- Positive number: Allows only these many number of local discs.

compute_monitors

Type list

Default ''

A comma-separated list of monitors that can be used for getting compute metrics. You can use the alias/name from the setuptools entry points for nova.compute.monitors.* namespaces. If no namespace is supplied, the cpu. namespace is assumed for backwards-compatibility.

NOTE: Only one monitor per namespace (For example: cpu) can be loaded at a time.

Possible values:

- An empty list will disable the feature (Default).
- An example value that would enable the CPU bandwidth monitor that uses the virt driver variant:

```
compute_monitors = cpu.virt_driver
```

default_ephemeral_format

Type string

Default <None>

The default format an ephemeral_volume will be formatted with on creation.

Possible values:

- ext2
- ext3
- ext4
- xfs
- ntfs (only for Windows guests)

vif_plugging_is_fatal

Type boolean

Default true

Determine if instance should boot or fail on VIF plugging timeout.

Nova sends a port update to Neutron after an instance has been scheduled, providing Neutron with the necessary information to finish setup of the port. Once completed, Neutron notifies Nova that it has finished setting up the port, at which point Nova resumes the boot of the instance since network connectivity is now supposed to be present. A timeout will occur if the reply is not received after a given interval.

This option determines what Nova does when the VIF plugging timeout event happens. When enabled, the instance will error out. When disabled, the instance will continue to boot on the assumption that the port is ready.

Possible values:

- True: Instances should fail after VIF plugging timeout
- False: Instances should continue booting after VIF plugging timeout

vif_plugging_timeout

Type integer

Default `300`

Minimum Value 0

Timeout for Neutron VIF plugging event message arrival.

Number of seconds to wait for Neutron vif plugging events to arrive before continuing or failing (see vif_plugging_is_fatal).

If you are hitting timeout failures at scale, consider running rootwrap in daemon mode in the neutron agent via the `[agent]/root_helper_daemon` neutron configuration option.

Related options:

- vif_plugging_is_fatal - If `vif_plugging_timeout` is set to zero and `vif_plugging_is_fatal` is False, events should not be expected to arrive at all.

injected_network_template

Type string

Default `$pybasedir/nova/virt/interfaces.template`

Path to /etc/network/interfaces template.

The path to a template file for the /etc/network/interfaces-style file, which will be populated by nova and subsequently used by cloudinit. This provides a method to configure network connectivity in environments without a DHCP server.

The template will be rendered using Jinja2 template engine, and receive a top-level key called `interfaces`. This key will contain a list of dictionaries, one for each interface.

Refer to the cloudinit documentaion for more information:

https://cloudinit.readthedocs.io/en/latest/topics/datasources.html

Possible values:

- A path to a Jinja2-formatted template for a Debian /etc/network/interfaces file. This applies even if using a non Debian-derived guest.

Related options:

- `flat_inject`: This must be set to `True` to ensure nova embeds network configuration information in the metadata provided through the config drive.

preallocate_images

Type string

Default `none`

Valid Values none, space

The image preallocation mode to use.

Image preallocation allows storage for instance images to be allocated up front when the instance is initially provisioned. This ensures immediate feedback is given if enough space isnt available. In addition, it should significantly improve performance on writes to new blocks and may even improve I/O performance to prewritten blocks due to reduced fragmentation.

Possible values

none No storage provisioning is done up front

space Storage is fully allocated at instance start

use_cow_images

Type boolean

Default `true`

Enable use of copy-on-write (cow) images.

QEMU/KVM allow the use of qcow2 as backing files. By disabling this, backing files will not be used.

force_raw_images

Type boolean

Default `true`

Force conversion of backing images to raw format.

Possible values:

- True: Backing image files will be converted to raw image format
- False: Backing image files will not be converted

Related options:

- `compute_driver`: Only the libvirt driver uses this option.
- `[libvirt]/images_type`: If images_type is rbd, setting this option to False is not allowed. See the bug https://bugs.launchpad.net/nova/+bug/1816686 for more details.

virt_mkfs

Type multi-valued

Default `''`

Name of the mkfs commands for ephemeral device.

The format is <os_type>=<mkfs command>

resize_fs_using_block_device

Type boolean

Default `false`

Enable resizing of filesystems via a block device.

If enabled, attempt to resize the filesystem by accessing the image over a block device. This is done by the host and may not be necessary if the image contains a recent version of cloud-init. Possible mechanisms require the nbd driver (for qcow and raw), or loop (for raw).

timeout_nbd

Type integer

Default `10`

Minimum Value 0

Amount of time, in seconds, to wait for NBD device start up.

image_cache_subdirectory_name

Type string

Default `_base`

Location of cached images.

This is NOT the full path - just a folder name relative to $instances_path. For per-compute-host cached images, set to _base_$my_ip

remove_unused_base_images

Type boolean

Default `true`

Should unused base images be removed?

remove_unused_original_minimum_age_seconds

Type integer

Default `86400`

Unused unresized base images younger than this will not be removed.

pointer_model

Type string

Default `usbtablet`

Valid Values ps2mouse, usbtablet, <None>

Generic property to specify the pointer type.

Input devices allow interaction with a graphical framebuffer. For example to provide a graphic tablet for absolute cursor movement.

If set, the hw_pointer_model image property takes precedence over this configuration option.

Related options:

- usbtablet must be configured with VNC enabled or SPICE enabled and SPICE agent disabled. When used with libvirt the instance mode should be configured as HVM.

Possible values

ps2mouse Uses relative movement. Mouse connected by PS2

usbtablet Uses absolute movement. Tablet connect by USB

<None> Uses default behavior provided by drivers (mouse on PS2 for libvirt x86)

vcpu_pin_set

Type string

Default `<None>`

Mask of host CPUs that can be used for `VCPU` resources.

The behavior of this option depends on the definition of the `[compute] cpu_dedicated_set` option and affects the behavior of the `[compute] cpu_shared_set` option.

- If `[compute] cpu_dedicated_set` is defined, defining this option will result in an error.

- If `[compute] cpu_dedicated_set` is not defined, this option will be used to determine inventory for `VCPU` resources and to limit the host CPUs that both pinned and unpinned instances can be scheduled to, overriding the `[compute] cpu_shared_set` option.

Possible values:

- A comma-separated list of physical CPU numbers that virtual CPUs can be allocated from. Each element should be either a single CPU number, a range of CPU numbers, or a caret followed by a CPU number to be excluded from a previous range. For example:

```
vcpu_pin_set = "4-12,^8,15"
```

Related options:

- `[compute] cpu_dedicated_set`
- `[compute] cpu_shared_set`

Warning: This option is deprecated for removal since 20.0.0. Its value may be silently ignored in the future.

Reason This option has been superseded by the `[compute] cpu_dedicated_set` and `[compute] cpu_shared_set` options, which allow things like the co-existence of pinned and unpinned instances on the same host (for the libvirt driver).

reserved_huge_pages

Type unknown type

Default `<None>`

Number of huge/large memory pages to reserved per NUMA host cell.

Possible values:

- A list of valid key=value which reflect NUMA node ID, page size (Default unit is KiB) and number of pages to be reserved. For example:

```
reserved_huge_pages = node:0,size:2048,count:64
reserved_huge_pages = node:1,size:1GB,count:1
```

In this example we are reserving on NUMA node 0 64 pages of 2MiB and on NUMA node 1 1 page of 1GiB.

reserved_host_disk_mb

Type integer

Default `0`

Minimum Value 0

Amount of disk resources in MB to make them always available to host. The disk usage gets reported back to the scheduler from nova-compute running on the compute nodes. To prevent the disk resources from being considered as available, this option can be used to reserve disk space for that host.

Possible values:

- Any positive integer representing amount of disk in MB to reserve for the host.

reserved_host_memory_mb

Type integer

Default 512

Minimum Value 0

Amount of memory in MB to reserve for the host so that it is always available to host processes. The host resources usage is reported back to the scheduler continuously from nova-compute running on the compute node. To prevent the host memory from being considered as available, this option is used to reserve memory for the host.

Possible values:

- Any positive integer representing amount of memory in MB to reserve for the host.

reserved_host_cpus

Type integer

Default 0

Minimum Value 0

Number of host CPUs to reserve for host processes.

The host resources usage is reported back to the scheduler continuously from nova-compute running on the compute node. This value is used to determine the `reserved` value reported to placement.

This option cannot be set if the `[compute] cpu_shared_set` or `[compute] cpu_dedicated_set` config options have been defined. When these options are defined, any host CPUs not included in these values are considered reserved for the host.

Possible values:

- Any positive integer representing number of physical CPUs to reserve for the host.

Related options:

- `[compute] cpu_shared_set`
- `[compute] cpu_dedicated_set`

cpu_allocation_ratio

Type floating point

Default <None>

Minimum Value 0.0

Virtual CPU to physical CPU allocation ratio.

This option is used to influence the hosts selected by the Placement API by configuring the allocation ratio for `VCPU` inventory. In addition, the `AggregateCoreFilter` (deprecated) will fall back to this configuration value if no per-aggregate setting is found.

Note: This option does not affect `PCPU` inventory, which cannot be overcommitted.

Note: If this option is set to something *other than* `None` or `0.0`, the allocation ratio will be overwritten by the value of this option, otherwise, the allocation ratio will not change. Once set to a non-default value, it is not possible to unset the config to get back to the default behavior. If you want to reset back to the initial value, explicitly specify it to the value of `initial_cpu_allocation_ratio`.

Possible values:

- Any valid positive integer or float value

Related options:

- initial_cpu_allocation_ratio

ram_allocation_ratio

Type floating point

Default <None>

Minimum Value 0.0

Virtual RAM to physical RAM allocation ratio.

This option is used to influence the hosts selected by the Placement API by configuring the allocation ratio for MEMORY_MB inventory. In addition, the AggregateRamFilter (deprecated) will fall back to this configuration value if no per-aggregate setting is found.

Note: If this option is set to something *other than* None or 0.0, the allocation ratio will be overwritten by the value of this option, otherwise, the allocation ratio will not change. Once set to a non-default value, it is not possible to unset the config to get back to the default behavior. If you want to reset back to the initial value, explicitly specify it to the value of initial_ram_allocation_ratio.

Possible values:

- Any valid positive integer or float value

Related options:

- initial_ram_allocation_ratio

disk_allocation_ratio

Type floating point

Default <None>

Minimum Value 0.0

Virtual disk to physical disk allocation ratio.

This option is used to influence the hosts selected by the Placement API by configuring the allocation ratio for DISK_GB inventory. In addition, the AggregateDiskFilter (deprecated) will fall back to this configuration value if no per-aggregate setting is found.

When configured, a ratio greater than 1.0 will result in over-subscription of the available physical disk, which can be useful for more efficiently packing instances created with images that do not use the entire virtual disk, such as sparse or compressed images. It can be set to a value between 0.0 and 1.0 in order to preserve a percentage of the disk for uses other than instances.

Note: If the value is set to >1, we recommend keeping track of the free disk space, as the value approaching 0 may result in the incorrect functioning of instances using it at the moment.

Note: If this option is set to something *other than* None or 0.0, the allocation ratio will be overwritten by the value of this option, otherwise, the allocation ratio will not change. Once set to a non-default value, it is not possible to unset the config to get back to the default behavior. If you want to reset back to the initial value, explicitly specify it to the value of initial_disk_allocation_ratio.

Possible values:

- Any valid positive integer or float value

Related options:

- `initial_disk_allocation_ratio`

`initial_cpu_allocation_ratio`

Type floating point

Default `16.0`

Minimum Value 0.0

Initial virtual CPU to physical CPU allocation ratio.

This is only used when initially creating the `computes_nodes` table record for a given nova-compute service.

See https://docs.openstack.org/nova/latest/admin/configuration/schedulers.html for more details and usage scenarios.

Related options:

- `cpu_allocation_ratio`

`initial_ram_allocation_ratio`

Type floating point

Default `1.5`

Minimum Value 0.0

Initial virtual RAM to physical RAM allocation ratio.

This is only used when initially creating the `computes_nodes` table record for a given nova-compute service.

See https://docs.openstack.org/nova/latest/admin/configuration/schedulers.html for more details and usage scenarios.

Related options:

- `ram_allocation_ratio`

`initial_disk_allocation_ratio`

Type floating point

Default `1.0`

Minimum Value 0.0

Initial virtual disk to physical disk allocation ratio.

This is only used when initially creating the `computes_nodes` table record for a given nova-compute service.

See https://docs.openstack.org/nova/latest/admin/configuration/schedulers.html for more details and usage scenarios.

Related options:

- `disk_allocation_ratio`

`console_host`

Type string

Default `<current_hostname>`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

Console proxy host to be used to connect to instances on this host. It is the publicly visible name for the console host.

Possible values:

- Current hostname (default) or any string representing hostname.

default_access_ip_network_name

Type string

Default <None>

Name of the network to be used to set access IPs for instances. If there are multiple IPs to choose from, an arbitrary one will be chosen.

Possible values:

- None (default)
- Any string representing network name.

instances_path

Type string

Default $state_path/instances

This option has a sample default set, which means that its actual default value may vary from the one documented above.

Specifies where instances are stored on the hypervisors disk. It can point to locally attached storage or a directory on NFS.

Possible values:

- $state_path/instances where state_path is a config option that specifies the top-level directory for maintaining novas state. (default) or Any string representing directory path.

Related options:

- [workarounds]/ensure_libvirt_rbd_instance_dir_cleanup

instance_usage_audit

Type boolean

Default false

This option enables periodic compute.instance.exists notifications. Each compute node must be configured to generate system usage data. These notifications are consumed by OpenStack Telemetry service.

live_migration_retry_count

Type integer

Default 30

Minimum Value 0

Maximum number of 1 second retries in live_migration. It specifies number of retries to iptables when it complains. It happens when an user continuously sends live-migration request to same host leading to concurrent request to iptables.

Possible values:

- Any positive integer representing retry count.

resume_guests_state_on_host_boot

Type boolean

Default `false`

This option specifies whether to start guests that were running before the host rebooted. It ensures that all of the instances on a Nova compute node resume their state each time the compute node boots or restarts.

network_allocate_retries

Type integer

Default `0`

Minimum Value 0

Number of times to retry network allocation. It is required to attempt network allocation retries if the virtual interface plug fails.

Possible values:

- Any positive integer representing retry count.

max_concurrent_builds

Type integer

Default `10`

Minimum Value 0

Limits the maximum number of instance builds to run concurrently by nova-compute. Compute service can attempt to build an infinite number of instances, if asked to do so. This limit is enforced to avoid building unlimited instance concurrently on a compute node. This value can be set per compute node.

Possible Values:

- 0 : treated as unlimited.
- Any positive integer representing maximum concurrent builds.

max_concurrent_live_migrations

Type integer

Default `1`

Minimum Value 0

Maximum number of live migrations to run concurrently. This limit is enforced to avoid outbound live migrations overwhelming the host/network and causing failures. It is not recommended that you change this unless you are very sure that doing so is safe and stable in your environment.

Possible values:

- 0 : treated as unlimited.
- Any positive integer representing maximum number of live migrations to run concurrently.

block_device_allocate_retries

Type integer

Default `60`

Minimum Value 0

The number of times to check for a volume to be available before attaching it during server create.

When creating a server with block device mappings where `source_type` is one of `blank`, `image` or `snapshot` and the `destination_type` is `volume`, the `nova-compute` service will create a volume and then attach it to the server. Before the volume can be attached, it must be in status available. This option controls how many times to check for the created volume to be available before it is attached.

If the operation times out, the volume will be deleted if the block device mapping `delete_on_termination` value is True.

It is recommended to configure the image cache in the block storage service to speed up this operation. See https://docs.openstack.org/cinder/latest/admin/blockstorage-image-volume-cache.html for details.

Possible values:

- 60 (default)
- If value is 0, then one attempt is made.
- For any value > 0, total attempts are (value + 1)

Related options:

- `block_device_allocate_retries_interval` - controls the interval between checks

`sync_power_state_pool_size`

Type integer

Default `1000`

Number of greenthreads available for use to sync power states.

This option can be used to reduce the number of concurrent requests made to the hypervisor or system with real instance power states for performance reasons, for example, with Ironic.

Possible values:

- Any positive integer representing greenthreads count.

`image_cache_manager_interval`

Type integer

Default `2400`

Minimum Value -1

Number of seconds to wait between runs of the image cache manager.

Possible values: * 0: run at the default rate. * -1: disable * Any other value

`bandwidth_poll_interval`

Type integer

Default `600`

Interval to pull network bandwidth usage info.

Not supported on all hypervisors. If a hypervisor doesnt support bandwidth usage, it will not get the info in the usage events.

Possible values:

- 0: Will run at the default periodic interval.
- Any value < 0: Disables the option.

- Any positive integer in seconds.

sync_power_state_interval

Type integer

Default `600`

Interval to sync power states between the database and the hypervisor.

The interval that Nova checks the actual virtual machine power state and the power state that Nova has in its database. If a user powers down their VM, Nova updates the API to report the VM has been powered down. Should something turn on the VM unexpectedly, Nova will turn the VM back off to keep the system in the expected state.

Possible values:

- 0: Will run at the default periodic interval.
- Any value < 0: Disables the option.
- Any positive integer in seconds.

Related options:

- If `handle_virt_lifecycle_events` in the `workarounds` group is false and this option is negative, then instances that get out of sync between the hypervisor and the Nova database will have to be synchronized manually.

heal_instance_info_cache_interval

Type integer

Default `60`

Interval between instance network information cache updates.

Number of seconds after which each compute node runs the task of querying Neutron for all of its instances networking information, then updates the Nova db with that information. Nova will never update its cache if this option is set to 0. If we dont update the cache, the metadata service and nova-api endpoints will be proxying incorrect network data about the instance. So, it is not recommended to set this option to 0.

Possible values:

- Any positive integer in seconds.
- Any value <=0 will disable the sync. This is not recommended.

reclaim_instance_interval

Type integer

Default `0`

Interval for reclaiming deleted instances.

A value greater than 0 will enable SOFT_DELETE of instances. This option decides whether the server to be deleted will be put into the SOFT_DELETED state. If this value is greater than 0, the deleted server will not be deleted immediately, instead it will be put into a queue until its too old (deleted time greater than the value of reclaim_instance_interval). The server can be recovered from the delete queue by using the restore action. If the deleted server remains longer than the value of reclaim_instance_interval, it will be deleted by a periodic task in the compute service automatically.

Note that this option is read from both the API and compute nodes, and must be set globally otherwise servers could be put into a soft deleted state in the API and never actually reclaimed (deleted) on the compute node.

Possible values:

- Any positive integer(in seconds) greater than 0 will enable this option.
- Any value <=0 will disable the option.

volume_usage_poll_interval

Type integer

Default `0`

Interval for gathering volume usages.

This option updates the volume usage cache for every volume_usage_poll_interval number of seconds.

Possible values:

- Any positive integer(in seconds) greater than 0 will enable this option.
- Any value <=0 will disable the option.

shelved_poll_interval

Type integer

Default `3600`

Interval for polling shelved instances to offload.

The periodic task runs for every shelved_poll_interval number of seconds and checks if there are any shelved instances. If it finds a shelved instance, based on the shelved_offload_time config value it offloads the shelved instances. Check shelved_offload_time config option description for details.

Possible values:

- Any value <= 0: Disables the option.
- Any positive integer in seconds.

Related options:

- `shelved_offload_time`

shelved_offload_time

Type integer

Default `0`

Time before a shelved instance is eligible for removal from a host.

By default this option is set to 0 and the shelved instance will be removed from the hypervisor immediately after shelve operation. Otherwise, the instance will be kept for the value of shelved_offload_time(in seconds) so that during the time period the unshelve action will be faster, then the periodic task will remove the instance from hypervisor after shelved_offload_time passes.

Possible values:

- **0: Instance will be immediately offloaded after being** shelved.
- Any value < 0: An instance will never offload.
- Any positive integer in seconds: The instance will exist for the specified number of seconds before being offloaded.

instance_delete_interval

Type integer

Default `300`

Interval for retrying failed instance file deletes.

This option depends on maximum_instance_delete_attempts. This option specifies how often to retry deletes whereas maximum_instance_delete_attempts specifies the maximum number of retry attempts that can be made.

Possible values:

- 0: Will run at the default periodic interval.
- Any value < 0: Disables the option.
- Any positive integer in seconds.

Related options:

- `maximum_instance_delete_attempts` from instance_cleaning_opts group.

`block_device_allocate_retries_interval`

Type integer

Default `3`

Minimum Value 0

Interval (in seconds) between block device allocation retries on failures.

This option allows the user to specify the time interval between consecutive retries. The `block_device_allocate_retries` option specifies the maximum number of retries.

Possible values:

- 0: Disables the option.
- Any positive integer in seconds enables the option.

Related options:

- `block_device_allocate_retries` - controls the number of retries

`scheduler_instance_sync_interval`

Type integer

Default `120`

Interval between sending the scheduler a list of current instance UUIDs to verify that its view of instances is in sync with nova.

If the CONF option scheduler_tracks_instance_changes is False, the sync calls will not be made. So, changing this option will have no effect.

If the out of sync situations are not very common, this interval can be increased to lower the number of RPC messages being sent. Likewise, if sync issues turn out to be a problem, the interval can be lowered to check more frequently.

Possible values:

- 0: Will run at the default periodic interval.
- Any value < 0: Disables the option.
- Any positive integer in seconds.

Related options:

- This option has no impact if `scheduler_tracks_instance_changes` is set to False.

`update_resources_interval`

Type integer

Default 0

Interval for updating compute resources.

This option specifies how often the update_available_resources periodic task should run. A number less than 0 means to disable the task completely. Leaving this at the default of 0 will cause this to run at the default periodic interval. Setting it to any positive value will cause it to run at approximately that number of seconds.

Possible values:

- 0: Will run at the default periodic interval.
- Any value < 0: Disables the option.
- Any positive integer in seconds.

reboot_timeout

Type integer

Default 0

Minimum Value 0

Time interval after which an instance is hard rebooted automatically.

When doing a soft reboot, it is possible that a guest kernel is completely hung in a way that causes the soft reboot task to not ever finish. Setting this option to a time period in seconds will automatically hard reboot an instance if it has been stuck in a rebooting state longer than N seconds.

Possible values:

- 0: Disables the option (default).
- Any positive integer in seconds: Enables the option.

instance_build_timeout

Type integer

Default 0

Minimum Value 0

Maximum time in seconds that an instance can take to build.

If this timer expires, instance status will be changed to ERROR. Enabling this option will make sure an instance will not be stuck in BUILD state for a longer period.

Possible values:

- 0: Disables the option (default)
- Any positive integer in seconds: Enables the option.

rescue_timeout

Type integer

Default 0

Minimum Value 0

Interval to wait before un-rescuing an instance stuck in RESCUE.

Possible values:

- 0: Disables the option (default)

- Any positive integer in seconds: Enables the option.

resize_confirm_window

Type integer

Default 0

Minimum Value 0

Automatically confirm resizes after N seconds.

Resize functionality will save the existing server before resizing. After the resize completes, user is requested to confirm the resize. The user has the opportunity to either confirm or revert all changes. Confirm resize removes the original server and changes server status from resized to active. Setting this option to a time period (in seconds) will automatically confirm the resize if the server is in resized state longer than that time.

Possible values:

- 0: Disables the option (default)
- Any positive integer in seconds: Enables the option.

shutdown_timeout

Type integer

Default 60

Minimum Value 0

Total time to wait in seconds for an instance to perform a clean shutdown.

It determines the overall period (in seconds) a VM is allowed to perform a clean shutdown. While performing stop, rescue and shelve, rebuild operations, configuring this option gives the VM a chance to perform a controlled shutdown before the instance is powered off. The default timeout is 60 seconds. A value of 0 (zero) means the guest will be powered off immediately with no opportunity for guest OS clean-up.

The timeout value can be overridden on a per image basis by means of os_shutdown_timeout that is an image metadata setting allowing different types of operating systems to specify how much time they need to shut down cleanly.

Possible values:

- A positive integer or 0 (default value is 60).

running_deleted_instance_action

Type string

Default reap

Valid Values reap, log, shutdown, noop

The compute service periodically checks for instances that have been deleted in the database but remain running on the compute node. The above option enables action to be taken when such instances are identified.

Related options:

- running_deleted_instance_poll_interval
- running_deleted_instance_timeout

Possible values

reap Powers down the instances and deletes them

log Logs warning message about deletion of the resource

shutdown Powers down instances and marks them as non-bootable which can be later used for debugging/analysis

noop Takes no action

running_deleted_instance_poll_interval

Type integer

Default `1800`

Time interval in seconds to wait between runs for the clean up action. If set to 0, above check will be disabled. If running_deleted_instance _action is set to log or reap, a value greater than 0 must be set.

Possible values:

- Any positive integer in seconds enables the option.
- 0: Disables the option.
- 1800: Default value.

Related options:

- running_deleted_instance_action

running_deleted_instance_timeout

Type integer

Default `0`

Time interval in seconds to wait for the instances that have been marked as deleted in database to be eligible for cleanup.

Possible values:

- Any positive integer in seconds(default is 0).

Related options:

- running_deleted_instance_action

maximum_instance_delete_attempts

Type integer

Default `5`

Minimum Value 1

The number of times to attempt to reap an instances files.

This option specifies the maximum number of retry attempts that can be made.

Possible values:

- Any positive integer defines how many attempts are made.

Related options:

- `[DEFAULT] instance_delete_interval` can be used to disable this option.

osapi_compute_unique_server_name_scope

Type string

Default ''

Valid Values , project, global

Sets the scope of the check for unique instance names.

The default doesnt check for unique names. If a scope for the name check is set, a launch of a new instance or an update of an existing instance with a duplicate name will result in an InstanceExists error. The uniqueness is case-insensitive. Setting this option can increase the usability for end users as they dont have to distinguish among instances with the same name by their IDs.

Possible values

An empty value means that no uniqueness check is done and duplicate names are possible

project The instance name check is done only for instances within the same project

global The instance name check is done for all instances regardless of the project

enable_new_services

Type boolean

Default `true`

Enable new nova-compute services on this host automatically.

When a new nova-compute service starts up, it gets registered in the database as an enabled service. Sometimes it can be useful to register new compute services in disabled state and then enabled them at a later point in time. This option only sets this behavior for nova-compute services, it does not auto-disable other services like nova-conductor, nova-scheduler, or nova-osapi_compute.

Possible values:

- `True`: Each new compute service is enabled as soon as it registers itself.
- `False`: Compute services must be enabled via an os-services REST API call or with the CLI with `nova service-enable <hostname> <binary>`, otherwise they are not ready to use.

instance_name_template

Type string

Default `instance-%08x`

Template string to be used to generate instance names.

This template controls the creation of the database name of an instance. This is *not* the display name you enter when creating an instance (via Horizon or CLI). For a new deployment it is advisable to change the default value (which uses the database autoincrement) to another value which makes use of the attributes of an instance, like `instance-%(uuid)s`. If you already have instances in your deployment when you change this, your deployment will break.

Possible values:

- A string which either uses the instance database ID (like the default)
- A string with a list of named database columns, for example `%(id)d` or `%(uuid)s` or `%(hostname)s`.

migrate_max_retries

Type integer

Default `-1`

Minimum Value -1

Number of times to retry live-migration before failing.

Possible values:

- If == -1, try until out of hosts (default)
- If == 0, only try once, no retries
- Integer greater than 0

`config_drive_format`

Type string

Default `iso9660`

Valid Values iso9660, vfat

Config drive format.

Config drive format that will contain metadata attached to the instance when it boots.

Related options:

- This option is meaningful when one of the following alternatives occur:
  1. `force_config_drive` option set to `true`
  2. the REST API call to create the instance contains an enable flag for config drive option
  3. the image used to create the instance requires a config drive, this is defined by `img_config_drive` property for that image.
- A compute node running Hyper-V hypervisor can be configured to attach config drive as a CD drive. To attach the config drive as a CD drive, set the `[hyperv] config_drive_cdrom` option to true.

Possible values

iso9660 A file system image standard that is widely supported across operating systems.

vfat Provided for legacy reasons and to enable live migration with the libvirt driver and non-shared storage

> **Warning:** This option is deprecated for removal since 19.0.0. Its value may be silently ignored in the future.
>
> **Reason** This option was originally added as a workaround for bug in libvirt, #1246201, that was resolved in libvirt v1.2.17. As a result, this option is no longer necessary or useful.

`force_config_drive`

Type boolean

Default `false`

Force injection to take place on a config drive

When this option is set to true config drive functionality will be forced enabled by default, otherwise users can still enable config drives via the REST API or image metadata properties. Launched instances are not affected by this option.

Possible values:

- **True: Force to use of config drive regardless the users input in the** REST API call.
- **False: Do not force use of config drive. Config drives can still be** enabled via the REST API or image metadata properties.

Related options:

- Use the mkisofs_cmd flag to set the path where you install the genisoimage program. If genisoimage is in same path as the nova-compute service, you do not need to set this flag.
- To use a config drive with Hyper-V, you must set the mkisofs_cmd value to the full path to an mkisofs.exe installation. Additionally, you must set the qemu_img_cmd value in the hyperv configuration section to the full path to an qemu-img command installation.

mkisofs_cmd

Type string

Default `genisoimage`

Name or path of the tool used for ISO image creation.

Use the `mkisofs_cmd` flag to set the path where you install the `genisoimage` program. If `genisoimage` is on the system path, you do not need to change the default value.

To use a config drive with Hyper-V, you must set the `mkisofs_cmd` value to the full path to an `mkisofs.exe` installation. Additionally, you must set the `qemu_img_cmd` value in the hyperv configuration section to the full path to an `qemu-img` command installation.

Possible values:

- Name of the ISO image creator program, in case it is in the same directory as the nova-compute service
- Path to ISO image creator program

Related options:

- This option is meaningful when config drives are enabled.
- To use config drive with Hyper-V, you must set the `qemu_img_cmd` value in the hyperv configuration section to the full path to an `qemu-img` command installation.

my_ip

Type string

Default `<host_ipv4>`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

The IP address which the host is using to connect to the management network.

Possible values:

- String with valid IP address. Default is IPv4 address of this host.

Related options:

- metadata_host

- my_block_storage_ip
- routing_source_ip
- vpn_ip

my_block_storage_ip

Type string

Default `$my_ip`

The IP address which is used to connect to the block storage network.

Possible values:

- String with valid IP address. Default is IP address of this host.

Related options:

- my_ip - if my_block_storage_ip is not set, then my_ip value is used.

host

Type string

Default `<current_hostname>`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

Hostname, FQDN or IP address of this host.

Used as:

- the oslo.messaging queue name for nova-compute worker
- we use this value for the binding_host sent to neutron. This means if you use a neutron agent, it should have the same value for host.
- cinder host attachment information

Must be valid within AMQP key.

Possible values:

- String with hostname, FQDN or IP address. Default is hostname of this host.

dhcpbridge_flagfile

Type multi-valued

Default `/etc/nova/nova-dhcpbridge.conf`

This option is a list of full paths to one or more configuration files for dhcpbridge. In most cases the default path of /etc/nova/nova-dhcpbridge.conf should be sufficient, but if you have special needs for configuring dhcpbridge, you can change or add to this list.

Possible values

- A list of strings, where each string is the full path to a dhcpbridge configuration file.

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

networks_path

Type string

Default `$state_path/networks`

The location where the network configuration files will be kept. The default is the networks directory off of the location where novas Python module is installed.

Possible values

- A string containing the full path to the desired configuration directory

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

public_interface

Type string

Default `eth0`

This is the name of the network interface for public IP addresses. The default is eth0.

Possible values:

- Any string representing a network interface name

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

dhcpbridge

Type string

Default `$bindir/nova-dhcpbridge`

The location of the binary nova-dhcpbridge. By default it is the binary named nova-dhcpbridge that is installed with all the other nova binaries.

Possible values:

- Any string representing the full path to the binary for dhcpbridge

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

routing_source_ip

Type string

Default `$my_ip`

The public IP address of the network host.

This is used when creating an SNAT rule.

Possible values:

- Any valid IP address

Related options:

- `force_snat_range`

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`dhcp_lease_time`

Type integer

Default `86400`

Minimum Value 1

The lifetime of a DHCP lease, in seconds. The default is 86400 (one day).

Possible values:

- Any positive integer value.

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`dns_server`

Type multi-valued

Default `''`

Despite the singular form of the name of this option, it is actually a list of zero or more server addresses that dnsmasq will use for DNS nameservers. If this is not empty, dnsmasq will not read /etc/resolv.conf, but will only use the servers specified in this option. If the option use_network_dns_servers is True, the dns1 and dns2 servers from the network will be appended to this list, and will be used as DNS servers, too.

Possible values:

- A list of strings, where each string is either an IP address or a FQDN.

Related options:

- `use_network_dns_servers`

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

use_network_dns_servers

Type boolean

Default `false`

When this option is set to True, the dns1 and dns2 servers for the network specified by the user on boot will be used for DNS, as well as any specified in the *dns_server* option.

Related options:

- `dns_server`

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

dmz_cidr

Type list

Default `''`

This option is a list of zero or more IP address ranges in your networks DMZ that should be accepted.

Possible values:

- A list of strings, each of which should be a valid CIDR.

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

force_snat_range

Type multi-valued

Default `''`

This is a list of zero or more IP ranges that traffic from the *routing_source_ip* will be SNATted to. If the list is empty, then no SNAT rules are created.

Possible values:

- A list of strings, each of which should be a valid CIDR.

Related options:

- `routing_source_ip`

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

dnsmasq_config_file

Type string

Default `''`

The path to the custom dnsmasq configuration file, if any.

Possible values:

- The full path to the configuration file, or an empty string if there is no custom dnsmasq configuration file.

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

`linuxnet_interface_driver`

Type string

Default `nova.network.linux_net.LinuxBridgeInterfaceDriver`

This is the class used as the ethernet device driver for linuxnet bridge operations. The default value should be all you need for most cases, but if you wish to use a customized class, set this option to the full dot-separated import path for that class.

Possible values:

- Any string representing a dot-separated class path that Nova can import.

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

`linuxnet_ovs_integration_bridge`

Type string

Default `br-int`

The name of the Open vSwitch bridge that is used with linuxnet when connecting with Open vSwitch.

Possible values:

- Any string representing a valid bridge name.

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

`send_arp_for_ha`

Type boolean

Default `false`

When True, when a device starts up, and upon binding floating IP addresses, arp messages will be sent to ensure that the arp caches on the compute hosts are up-to-date.

Related options:

- `send_arp_for_ha_count`

`send_arp_for_ha_count`

Type integer

Default `3`

When arp messages are configured to be sent, they will be sent with the count set to the value of this option. Of course, if this is set to zero, no arp messages will be sent.

Possible values:

- Any integer greater than or equal to 0

Related options:

- `send_arp_for_ha`

`use_single_default_gateway`

Type boolean

Default `false`

When set to True, only the firt nic of a VM will get its default gateway from the DHCP server.

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`forward_bridge_interface`

Type multi-valued

Default `all`

One or more interfaces that bridges can forward traffic to. If any of the items in this list is the special keyword all, then all traffic will be forwarded.

Possible values:

- A list of zero or more interface names, or the word all.

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`metadata_host`

Type string

Default `$my_ip`

This option determines the IP address for the network metadata API server.

This is really the client side of the metadata host equation that allows nova-network to find the metadata server when doing a default multi host networking.

Possible values:

- Any valid IP address. The default is the address of the Nova API server.

Related options:

- metadata_port

metadata_port

Type port number

Default 8775

Minimum Value 0

Maximum Value 65535

This option determines the port used for the metadata API server.

Related options:

- metadata_host

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

iptables_top_regex

Type string

Default ''

This expression, if defined, will select any matching iptables rules and place them at the top when applying metadata changes to the rules.

Possible values:

- Any string representing a valid regular expression, or an empty string

Related options:

- iptables_bottom_regex

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

iptables_bottom_regex

Type string

Default ''

This expression, if defined, will select any matching iptables rules and place them at the bottom when applying metadata changes to the rules.

Possible values:

- Any string representing a valid regular expression, or an empty string

Related options:

- iptables_top_regex

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

iptables_drop_action

Type string

Default `DROP`

By default, packets that do not pass the firewall are DROPped. In many cases, though, an operator may find it more useful to change this from DROP to REJECT, so that the user issuing those packets may have a better idea as to whats going on, or LOGDROP in order to record the blocked traffic before DROPping.

Possible values:

- A string representing an iptables chain. The default is DROP.

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

defer_iptables_apply

Type boolean

Default `false`

Defer application of IPTables rules until after init phase.

When a compute service is restarted each instance running on the host has its iptables rules built and applied sequentially during the host init stage. The impact of this, especially on a host running many instances, can be observed as a period where some instances are not accessible as the existing iptables rules have been torn down and not yet re-applied.

This is a workaround that prevents the application of the iptables rules until all instances on the host had been initialised then the rules for all instances are applied all at once preventing a blackout period.

> **Warning:** This option is deprecated for removal since 19.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

ovs_vsctl_timeout

Type integer

Default `120`

Minimum Value 0

This option represents the period of time, in seconds, that the ovs_vsctl calls will wait for a response from the database before timing out. A setting of 0 means that the utility should wait forever for a response.

Possible values:

- Any positive integer if a limited timeout is desired, or zero if the calls should wait forever for a response.

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

fake_network

Type boolean

Default `false`

This option is used mainly in testing to avoid calls to the underlying network utilities.

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

ebtables_exec_attempts

Type integer

Default `3`

Minimum Value 1

This option determines the number of times to retry ebtables commands before giving up. The minimum number of retries is 1.

Possible values:

- Any positive integer

Related options:

- `ebtables_retry_interval`

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

ebtables_retry_interval

Type floating point

Default `1.0`

This option determines the time, in seconds, that the system will sleep in between ebtables retries. Note that each successive retry waits a multiple of this value, so for example, if this is set to the default of 1.0 seconds, and ebtables_exec_attempts is 4, after the first failure, the system will sleep for 1 * 1.0 seconds, after the second failure it will sleep 2 * 1.0 seconds, and after the third failure it will sleep 3 * 1.0 seconds.

Possible values:

- Any non-negative float or integer. Setting this to zero will result in no waiting between attempts.

Related options:

- ebtables_exec_attempts

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

use_neutron

Type boolean

Default `true`

Enable neutron as the backend for networking.

Determine whether to use Neutron or Nova Network as the back end. Set to true to use neutron.

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

flat_injected

Type boolean

Default `false`

This option determines whether the network setup information is injected into the VM before it is booted. While it was originally designed to be used only by nova-network, it is also used by the vmware and xenapi virt drivers to control whether network information is injected into a VM. The libvirt virt driver also uses it when we use config_drive to configure network to control whether network information is injected into a VM.

flat_network_bridge

Type string

Default `<None>`

This option determines the bridge used for simple network interfaces when no bridge is specified in the VM creation request.

Please note that this option is only used when using nova-network instead of Neutron in your deployment.

Possible values:

- Any string representing a valid network bridge, such as br100

Related options:

- `use_neutron`

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

flat_network_dns

Type string

Default `8.8.4.4`

This is the address of the DNS server for a simple network. If this option is not specified, the default of 8.8.4.4 is used.

Please note that this option is only used when using nova-network instead of Neutron in your deployment.

Possible values:

- Any valid IP address.

Related options:

- `use_neutron`

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`flat_interface`

Type string

Default `<None>`

This option is the name of the virtual interface of the VM on which the bridge will be built. While it was originally designed to be used only by nova-network, it is also used by libvirt for the bridge interface name.

Possible values:

- Any valid virtual interface name, such as eth0

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`vlan_start`

Type integer

Default `100`

Minimum Value 1

Maximum Value 4094

This is the VLAN number used for private networks. Note that the when creating the networks, if the specified number has already been assigned, nova-network will increment this number until it finds an available VLAN.

Please note that this option is only used when using nova-network instead of Neutron in your deployment. It also will be ignored if the configuration option for *network_manager* is not set to the default of nova.network.manager.VlanManager.

Possible values:

- Any integer between 1 and 4094. Values outside of that range will raise a ValueError exception.

Related options:

- `network_manager`
- `use_neutron`

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

vlan_interface

Type string

Default `<None>`

This option is the name of the virtual interface of the VM on which the VLAN bridge will be built. While it was originally designed to be used only by nova-network, it is also used by libvirt and xenapi for the bridge interface name.

Please note that this setting will be ignored in nova-network if the configuration option for *network_manager* is not set to the default of nova.network.manager.VlanManager.

Possible values:

- Any valid virtual interface name, such as eth0

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options. While this option has an effect when using neutron, it incorrectly override the value provided by neutron and should therefore not be used.

num_networks

Type integer

Default `1`

Minimum Value 1

This option represents the number of networks to create if not explicitly specified when the network is created. The only time this is used is if a CIDR is specified, but an explicit network_size is not. In that case, the subnets are created by diving the IP address space of the CIDR by num_networks. The resulting subnet sizes cannot be larger than the configuration option *network_size*; in that event, they are reduced to *network_size*, and a warning is logged.

Please note that this option is only used when using nova-network instead of Neutron in your deployment.

Possible values:

- Any positive integer is technically valid, although there are practical limits based upon available IP address space and virtual interfaces.

Related options:

- `use_neutron`
- `network_size`

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

vpn_ip

Type string

Default `$my_ip`

This option is no longer used since the /os-cloudpipe API was removed in the 16.0.0 Pike release. This is the public IP address for the cloudpipe VPN servers. It defaults to the IP address of the host.

Please note that this option is only used when using nova-network instead of Neutron in your deployment. It also will be ignored if the configuration option for *network_manager* is not set to the default of nova.network.manager.VlanManager.

Possible values:

- Any valid IP address. The default is `$my_ip`, the IP address of the VM.

Related options:

- `network_manager`
- `use_neutron`
- `vpn_start`

Warning: This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

vpn_start

Type port number

Default `1000`

Minimum Value 0

Maximum Value 65535

This is the port number to use as the first VPN port for private networks.

Please note that this option is only used when using nova-network instead of Neutron in your deployment. It also will be ignored if the configuration option for *network_manager* is not set to the default of nova.network.manager.VlanManager, or if you specify a value the vpn_start parameter when creating a network.

Possible values:

- Any integer representing a valid port number. The default is 1000.

Related options:

- `use_neutron`
- `vpn_ip`
- `network_manager`

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

network_size

Type integer

Default 256

Minimum Value 1

This option determines the number of addresses in each private subnet.

Please note that this option is only used when using nova-network instead of Neutron in your deployment.

Possible values:

- Any positive integer that is less than or equal to the available network size. Note that if you are creating multiple networks, they must all fit in the available IP address space. The default is 256.

Related options:

- use_neutron
- num_networks

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

fixed_range_v6

Type string

Default fd00::/48

This option determines the fixed IPv6 address block when creating a network.

Please note that this option is only used when using nova-network instead of Neutron in your deployment.

Possible values:

- Any valid IPv6 CIDR

Related options:

- use_neutron

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

gateway

Type string

Default <None>

This is the default IPv4 gateway. It is used only in the testing suite.

Please note that this option is only used when using nova-network instead of Neutron in your deployment.

Possible values:

- Any valid IP address.

Related options:

- `use_neutron`
- `gateway_v6`

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`gateway_v6`

Type string

Default `<None>`

This is the default IPv6 gateway. It is used only in the testing suite.

Please note that this option is only used when using nova-network instead of Neutron in your deployment.

Possible values:

- Any valid IP address.

Related options:

- `use_neutron`
- `gateway`

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`cnt_vpn_clients`

Type integer

Default `0`

Minimum Value 0

This option represents the number of IP addresses to reserve at the top of the address range for VPN clients. It also will be ignored if the configuration option for *network_manager* is not set to the default of nova.network.manager.VlanManager.

Possible values:

- Any integer, 0 or greater.

Related options:

- `use_neutron`

- network_manager

Warning: This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

fixed_ip_disassociate_timeout

Type integer

Default 600

Minimum Value 0

This is the number of seconds to wait before disassociating a deallocated fixed IP address. This is only used with the nova-network service, and has no effect when using neutron for networking.

Possible values:

- Any integer, zero or greater.

Related options:

- use_neutron

Warning: This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

create_unique_mac_address_attempts

Type integer

Default 5

Minimum Value 1

This option determines how many times nova-network will attempt to create a unique MAC address before giving up and raising a *VirtualInterfaceMacAddressException* error.

Possible values:

- Any positive integer. The default is 5.

Related options:

- use_neutron

Warning: This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

teardown_unused_network_gateway

Type boolean

Default false

Determines whether unused gateway devices, both VLAN and bridge, are deleted if the network is in nova-network VLAN mode and is multi-hosted.

Related options:

- `use_neutron`
- `vpn_ip`
- `fake_network`

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`force_dhcp_release`

Type boolean

Default `true`

When this option is True, a call is made to release the DHCP for the instance when that instance is terminated.

Related options:

- `use_neutron`

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`update_dns_entries`

Type boolean

Default `false`

When this option is True, whenever a DNS entry must be updated, a fanout cast message is sent to all network hosts to update their DNS entries in multi-host mode.

Related options:

- `use_neutron`

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`dns_update_periodic_interval`

Type integer

Default `-1`

Minimum Value -1

This option determines the time, in seconds, to wait between refreshing DNS entries for the network.

Possible values:

- A positive integer
- -1 to disable updates

Related options:

- `use_neutron`

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`l3_lib`

Type string

Default `nova.network.l3.LinuxNetL3`

This option allows you to specify the L3 management library to be used.

Possible values:

- Any dot-separated string that represents the import path to an L3 networking library.

Related options:

- `use_neutron`

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`share_dhcp_address`

Type boolean

Default `false`

THIS VALUE SHOULD BE SET WHEN CREATING THE NETWORK.

If True in multi_host mode, all compute hosts share the same dhcp address. The same IP address used for DHCP will be added on each nova-network node which is only visible to the VMs on the same host.

The use of this configuration has been deprecated and may be removed in any release after Mitaka. It is recommended that instead of relying on this option, an explicit value should be passed to create_networks() as a keyword argument with the name share_address.

> **Warning:** This option is deprecated for removal since 2014.2. Its value may be silently ignored in the future.

`ldap_dns_url`

Type URI

Default `ldap://ldap.example.com:389`

URL for LDAP server which will store DNS entries

Possible values:

- A valid LDAP URL representing the server

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`ldap_dns_user`

Type string

Default `uid=admin,ou=people,dc=example,dc=org`

Bind user for LDAP server

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`ldap_dns_password`

Type string

Default `password`

Bind users password for LDAP server

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`ldap_dns_soa_hostmaster`

Type string

Default `hostmaster@example.org`

Hostmaster for LDAP DNS driver Statement of Authority

Possible values:

- Any valid string representing LDAP DNS hostmaster.

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`ldap_dns_servers`

Type multi-valued

Default `dns.example.org`

DNS Servers for LDAP DNS driver

Possible values:

- A valid URL representing a DNS server

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

`ldap_dns_base_dn`

Type string

Default `ou=hosts,dc=example,dc=org`

Base distinguished name for the LDAP search query

This option helps to decide where to look up the host in LDAP.

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

`ldap_dns_soa_refresh`

Type integer

Default `1800`

Refresh interval (in seconds) for LDAP DNS driver Start of Authority

Time interval, a secondary/slave DNS server waits before requesting for primary DNS servers current SOA record. If the records are different, secondary DNS server will request a zone transfer from primary.

NOTE: Lower values would cause more traffic.

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

`ldap_dns_soa_retry`

Type integer

Default `3600`

Retry interval (in seconds) for LDAP DNS driver Start of Authority

Time interval, a secondary/slave DNS server should wait, if an attempt to transfer zone failed during the previous refresh interval.

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

ldap_dns_soa_expiry

Type integer

Default `86400`

Expiry interval (in seconds) for LDAP DNS driver Start of Authority

Time interval, a secondary/slave DNS server holds the information before it is no longer considered authoritative.

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

ldap_dns_soa_minimum

Type integer

Default `7200`

Minimum interval (in seconds) for LDAP DNS driver Start of Authority

It is Minimum time-to-live applies for all resource records in the zone file. This value is supplied to other servers how long they should keep the data in cache.

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

multi_host

Type boolean

Default `false`

Default value for multi_host in networks.

nova-network service can operate in a multi-host or single-host mode. In multi-host mode each compute node runs a copy of nova-network and the instances on that compute node use the compute node as a gateway to the Internet. Where as in single-host mode, a central server runs the nova-network service. All compute nodes forward traffic from the instances to the cloud controller which then forwards traffic to the Internet.

If this options is set to true, some rpc network calls will be sent directly to host.

Note that this option is only used when using nova-network instead of Neutron in your deployment.

Related options:

- `use_neutron`

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

network_driver

Type string

Default `nova.network.linux_net`

Driver to use for network creation.

Network driver initializes (creates bridges and so on) only when the first VM lands on a host node. All network managers configure the network using network drivers. The driver is not tied to any particular network manager.

The default Linux driver implements vlans, bridges, and iptables rules using linux utilities.

Note that this option is only used when using nova-network instead of Neutron in your deployment.

Related options:

- `use_neutron`

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

firewall_driver

Type string

Default `nova.virt.firewall.NoopFirewallDriver`

Firewall driver to use with `nova-network` service.

This option only applies when using the `nova-network` service. When using another networking services, such as Neutron, this should be to set to the `nova.virt.firewall.NoopFirewallDriver`.

Possible values:

- `nova.virt.firewall.IptablesFirewallDriver`
- `nova.virt.firewall.NoopFirewallDriver`
- `nova.virt.libvirt.firewall.IptablesFirewallDriver`
- []

Related options:

- `use_neutron`: This must be set to `False` to enable `nova-network` networking

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

allow_same_net_traffic

Type boolean

Default `true`

Determine whether to allow network traffic from same network.

When set to true, hosts on the same subnet are not filtered and are allowed to pass all types of traffic between them. On a flat network, this allows all instances from all projects unfiltered communication. With VLAN networking, this allows access between instances within the same project.

This option only applies when using the `nova-network` service. When using another networking services, such as Neutron, security groups or other approaches should be used.

Possible values:

- True: Network traffic should be allowed pass between all instances on the same network, regardless of their tenant and security policies
- False: Network traffic should not be allowed pass between instances unless it is unblocked in a security group

Related options:

- `use_neutron`: This must be set to `False` to enable `nova-network` networking
- `firewall_driver`: This must be set to `nova.virt.libvirt.firewall.IptablesFirewallDriver` to ensure the libvirt firewall driver is enabled.

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

`default_floating_pool`

Type string

Default `nova`

Default pool for floating IPs.

This option specifies the default floating IP pool for allocating floating IPs.

While allocating a floating ip, users can optionally pass in the name of the pool they want to allocate from, otherwise it will be pulled from the default pool.

If this option is not set, then nova is used as default floating pool.

Possible values:

- Any string representing a floating IP pool name

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason This option was used for two purposes: to set the floating IP pool name for nova-network and to do the same for neutron. nova-network is deprecated, as are any related configuration options. Users of neutron, meanwhile, should use the default_floating_pool option in the [neutron] group.

`auto_assign_floating_ip`

Type boolean

Default `false`

Autoassigning floating IP to VM

When set to True, floating IP is auto allocated and associated to the VM upon creation.

Related options:

- use_neutron: this options only works with nova-network.

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`floating_ip_dns_manager`

Type string

Default `nova.network.noop_dns_driver.NoopDNSDriver`

Full class name for the DNS Manager for floating IPs.

This option specifies the class of the driver that provides functionality to manage DNS entries associated with floating IPs.

When a user adds a DNS entry for a specified domain to a floating IP, nova will add a DNS entry using the specified floating DNS driver. When a floating IP is deallocated, its DNS entry will automatically be deleted.

Possible values:

- Full Python path to the class to be used

Related options:

- use_neutron: this options only works with nova-network.

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

`instance_dns_manager`

Type string

Default `nova.network.noop_dns_driver.NoopDNSDriver`

Full class name for the DNS Manager for instance IPs.

This option specifies the class of the driver that provides functionality to manage DNS entries for instances.

On instance creation, nova will add DNS entries for the instance name and id, using the specified instance DNS driver and domain. On instance deletion, nova will remove the DNS entries.

Possible values:

- Full Python path to the class to be used

Related options:

- use_neutron: this options only works with nova-network.

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

instance_dns_domain

Type string

Default ''

If specified, Nova checks if the availability_zone of every instance matches what the database says the availability_zone should be for the specified dns_domain.

Related options:

- use_neutron: this options only works with nova-network.

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

use_ipv6

Type boolean

Default false

Assign IPv6 and IPv4 addresses when creating instances.

Related options:

- use_neutron: this only works with nova-network.

> **Warning:** This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

ipv6_backend

Type string

Default rfc2462

Valid Values rfc2462, account_identifier

Abstracts out IPv6 address generation to pluggable backends.

nova-network can be put into dual-stack mode, so that it uses both IPv4 and IPv6 addresses. In dual-stack mode, by default, instances acquire IPv6 global unicast addresses with the help of stateless address auto-configuration mechanism.

Related options:

- use_neutron: this option only works with nova-network.
- use_ipv6: this option only works if ipv6 is enabled for nova-network.

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

enable_network_quota

Type boolean

Default `false`

This option is used to enable or disable quota checking for tenant networks.

Related options:

- quota_networks

Warning: This option is deprecated for removal since 14.0.0. Its value may be silently ignored in the future.

Reason CRUD operations on tenant networks are only available when using nova-network and nova-network is itself deprecated.

quota_networks

Type integer

Default `3`

Minimum Value 0

This option controls the number of private networks that can be created per project (or per tenant).

Related options:

- enable_network_quota

Warning: This option is deprecated for removal since 14.0.0. Its value may be silently ignored in the future.

Reason CRUD operations on tenant networks are only available when using nova-network and nova-network is itself deprecated.

network_manager

Type string

Default `nova.network.manager.VlanManager`

Valid Values nova.network.manager.FlatManager, nova.network.manager.FlatDHCPManager, nova.network.manager.VlanManager

Full class name for the Manager for network

Warning: This option is deprecated for removal since 18.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

record

Type string

Default `<None>`

Filename that will be used for storing websocket frames received and sent by a proxy service (like VNC, spice, serial) running on this host. If this is not set, no recording will be done.

daemon

Type boolean

Default `false`

Run as a background process.

ssl_only

Type boolean

Default `false`

Disallow non-encrypted connections.

source_is_ipv6

Type boolean

Default `false`

Set to True if source host is addressed with IPv6.

cert

Type string

Default `self.pem`

Path to SSL certificate file.

key

Type string

Default `<None>`

SSL key file (if separate from cert).

web

Type string

Default `/usr/share/spice-html5`

Path to directory with content which will be served by a web server.

pybasedir

Type string

Default `<Path>`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

The directory where the Nova python modules are installed.

This directory is used to store template files for networking and remote console access. It is also the default path for other config options which need to persist Nova internal data. It is very unlikely that you need to change this option from its default value.

Possible values:

- The full path to a directory.

Related options:

- `state_path`

`bindir`

Type string

Default `/home/zuul/src/opendev.org/openstack/nova/.tox/docs/local/bin`

The directory where the Nova binaries are installed.

This option is only relevant if the networking capabilities from Nova are used (see services below). Novas networking capabilities are targeted to be fully replaced by Neutron in the future. It is very unlikely that you need to change this option from its default value.

Possible values:

- The full path to a directory.

`state_path`

Type string

Default `$pybasedir`

The top-level directory for maintaining Novas state.

This directory is used to store Novas internal state. It is used by a variety of other config options which derive from this. In some scenarios (for example migrations) it makes sense to use a storage location which is shared between multiple compute hosts (for example via NFS). Unless the option `instances_path` gets overwritten, this directory can grow very large.

Possible values:

- The full path to a directory. Defaults to value provided in `pybasedir`.

`long_rpc_timeout`

Type integer

Default `1800`

This option allows setting an alternate timeout value for RPC calls that have the potential to take a long time. If set, RPC calls to other services will use this value for the timeout (in seconds) instead of the global rpc_response_timeout value.

Operations with RPC calls that utilize this value:

- live migration
- scheduling
- enabling/disabling a compute service
- volume attach

Related options:

- rpc_response_timeout

report_interval

Type integer

Default `10`

Number of seconds indicating how frequently the state of services on a given hypervisor is reported. Nova needs to know this to determine the overall health of the deployment.

Related Options:

- service_down_time report_interval should be less than service_down_time. If service_down_time is less than report_interval, services will routinely be considered down, because they report in too rarely.

service_down_time

Type integer

Default `60`

Maximum time in seconds since last check-in for up service

Each compute node periodically updates their database status based on the specified report interval. If the compute node hasnt updated the status for more than service_down_time, then the compute node is considered down.

Related Options:

- report_interval (service_down_time should not be less than report_interval)
- scheduler.periodic_task_interval

periodic_enable

Type boolean

Default `true`

Enable periodic tasks.

If set to true, this option allows services to periodically run tasks on the manager.

In case of running multiple schedulers or conductors you may want to run periodic tasks on only one host - in this case disable this option for all hosts but one.

periodic_fuzzy_delay

Type integer

Default `60`

Minimum Value 0

Number of seconds to randomly delay when starting the periodic task scheduler to reduce stampeding.

When compute workers are restarted in unison across a cluster, they all end up running the periodic tasks at the same time causing problems for the external services. To mitigate this behavior, periodic_fuzzy_delay option allows you to introduce a random initial delay when starting the periodic task scheduler.

Possible Values:

- Any positive integer (in seconds)
- 0 : disable the random delay

enabled_apis

Type list

Default `osapi_compute,metadata`

List of APIs to be enabled by default.

enabled_ssl_apis

Type list

Default `''`

List of APIs with enabled SSL.

Nova provides SSL support for the API servers. enabled_ssl_apis option allows configuring the SSL support.

osapi_compute_listen

Type string

Default `0.0.0.0`

IP address on which the OpenStack API will listen.

The OpenStack API service listens on this IP address for incoming requests.

osapi_compute_listen_port

Type port number

Default `8774`

Minimum Value 0

Maximum Value 65535

Port on which the OpenStack API will listen.

The OpenStack API service listens on this port number for incoming requests.

osapi_compute_workers

Type integer

Default `<None>`

Minimum Value 1

Number of workers for OpenStack API service. The default will be the number of CPUs available.

OpenStack API services can be configured to run as multi-process (workers). This overcomes the problem of reduction in throughput when API request concurrency increases. OpenStack API service will run in the specified number of processes.

Possible Values:

- Any positive integer
- None (default value)

metadata_listen

Type string

Default `0.0.0.0`

IP address on which the metadata API will listen.

The metadata API service listens on this IP address for incoming requests.

metadata_listen_port

Type port number

Default `8775`

Minimum Value 0

Maximum Value 65535

Port on which the metadata API will listen.

The metadata API service listens on this port number for incoming requests.

`metadata_workers`

Type integer

Default `<None>`

Minimum Value 1

Number of workers for metadata service. If not specified the number of available CPUs will be used.

The metadata service can be configured to run as multi-process (workers). This overcomes the problem of reduction in throughput when API request concurrency increases. The metadata service will run in the specified number of processes.

Possible Values:

- Any positive integer
- None (default value)

`servicegroup_driver`

Type string

Default `db`

Valid Values db, mc

This option specifies the driver to be used for the servicegroup service.

ServiceGroup API in nova enables checking status of a compute node. When a compute worker running the nova-compute daemon starts, it calls the join API to join the compute group. Services like nova scheduler can query the ServiceGroup API to check if a node is alive. Internally, the ServiceGroup client driver automatically updates the compute worker status. There are multiple backend implementations for this service: Database ServiceGroup driver and Memcache ServiceGroup driver.

Related Options:

- `service_down_time` (maximum time since last check-in for up service)

Possible values

db Database ServiceGroup driver

mc Memcache ServiceGroup driver

api

Options under this group are used to define Nova API.

`auth_strategy`

Type string

Default `keystone`

Valid Values keystone, noauth2

Determine the strategy to use for authentication.

Possible values

keystone Use keystone for authentication.

noauth2 Designed for testing only, as it does no actual credential checking. noauth2 provides administrative credentials only if admin is specified as the username.

Table 16: Deprecated Variations

Group	Name
DEFAULT	auth_strategy

`use_forwarded_for`

Type boolean

Default `false`

When True, the X-Forwarded-For header is treated as the canonical remote address. When False (the default), the remote_address header is used.

You should only enable this if you have an HTML sanitizing proxy.

Table 17: Deprecated Variations

Group	Name
DEFAULT	use_forwarded_for

`config_drive_skip_versions`

Type string

Default `1.0 2007-01-19 2007-03-01 2007-08-29 2007-10-10 2007-12-15 2008-02-01 2008-09-01`

When gathering the existing metadata for a config drive, the EC2-style metadata is returned for all versions that dont appear in this option. As of the Liberty release, the available versions are:

- 1.0
- 2007-01-19
- 2007-03-01
- 2007-08-29
- 2007-10-10
- 2007-12-15
- 2008-02-01
- 2008-09-01
- 2009-04-04

The option is in the format of a single string, with each version separated by a space.

Possible values:

- Any string that represents zero or more versions, separated by spaces.

Table 18: Deprecated Variations

Group	Name
DEFAULT	config_drive_skip_versions

vendordata_providers

Type list

Default `StaticJSON`

A list of vendordata providers.

vendordata providers are how deployers can provide metadata via configdrive and metadata that is specific to their deployment.

For more information on the requirements for implementing a vendordata dynamic endpoint, please see the vendordata.rst file in the nova developer reference.

Related options:

- `vendordata_dynamic_targets`
- `vendordata_dynamic_ssl_certfile`
- `vendordata_dynamic_connect_timeout`
- `vendordata_dynamic_read_timeout`
- `vendordata_dynamic_failure_fatal`

Table 19: Deprecated Variations

Group	Name
DEFAULT	vendordata_providers

vendordata_dynamic_targets

Type list

Default `''`

A list of targets for the dynamic vendordata provider. These targets are of the form `<name>@<url>`.

The dynamic vendordata provider collects metadata by contacting external REST services and querying them for information about the instance. This behaviour is documented in the vendordata.rst file in the nova developer reference.

Table 20: Deprecated Variations

Group	Name
DEFAULT	vendordata_dynamic_targets

vendordata_dynamic_ssl_certfile

Type string

Default `''`

Path to an optional certificate file or CA bundle to verify dynamic vendordata REST services ssl certificates against.

Possible values:

- An empty string, or a path to a valid certificate file

Related options:

- vendordata_providers
- vendordata_dynamic_targets
- vendordata_dynamic_connect_timeout
- vendordata_dynamic_read_timeout
- vendordata_dynamic_failure_fatal

Table 21: Deprecated Variations

Group	Name
DEFAULT	vendordata_dynamic_ssl_certfile

vendordata_dynamic_connect_timeout

Type integer

Default 5

Minimum Value 3

Maximum wait time for an external REST service to connect.

Possible values:

- Any integer with a value greater than three (the TCP packet retransmission timeout). Note that instance start may be blocked during this wait time, so this value should be kept small.

Related options:

- vendordata_providers
- vendordata_dynamic_targets
- vendordata_dynamic_ssl_certfile
- vendordata_dynamic_read_timeout
- vendordata_dynamic_failure_fatal

Table 22: Deprecated Variations

Group	Name
DEFAULT	vendordata_dynamic_connect_timeout

vendordata_dynamic_read_timeout

Type integer

Default 5

Minimum Value 0

Maximum wait time for an external REST service to return data once connected.

Possible values:

- Any integer. Note that instance start is blocked during this wait time, so this value should be kept small.

Related options:

- vendordata_providers
- vendordata_dynamic_targets
- vendordata_dynamic_ssl_certfile
- vendordata_dynamic_connect_timeout
- vendordata_dynamic_failure_fatal

Table 23: Deprecated Variations

Group	Name
DEFAULT	vendordata_dynamic_read_timeout

vendordata_dynamic_failure_fatal

Type boolean

Default `false`

Should failures to fetch dynamic vendordata be fatal to instance boot?

Related options:

- vendordata_providers
- vendordata_dynamic_targets
- vendordata_dynamic_ssl_certfile
- vendordata_dynamic_connect_timeout
- vendordata_dynamic_read_timeout

metadata_cache_expiration

Type integer

Default `15`

Minimum Value 0

This option is the time (in seconds) to cache metadata. When set to 0, metadata caching is disabled entirely; this is generally not recommended for performance reasons. Increasing this setting should improve response times of the metadata API when under heavy load. Higher values may increase memory usage, and result in longer times for host metadata changes to take effect.

Table 24: Deprecated Variations

Group	Name
DEFAULT	metadata_cache_expiration

local_metadata_per_cell

Type boolean

Default `false`

Indicates that the nova-metadata API service has been deployed per-cell, so that we can have better performance and data isolation in a multi-cell deployment. Users should consider the use of this configuration depending on how neutron is setup. If you have networks that span cells, you might need to run nova-metadata API service

globally. If your networks are segmented along cell boundaries, then you can run nova-metadata API service per cell. When running nova-metadata API service per cell, you should also configure each Neutron metadata-agent to point to the corresponding nova-metadata API service.

dhcp_domain

Type string

Default `novalocal`

Domain name used to configure FQDN for instances.

This option has two purposes:

1. For *neutron* and *nova-network* users, it is used to configure a fully-qualified domain name for instance hostnames. If unset, only the hostname without a domain will be configured.
2. (Deprecated) For *nova-network* users, this option configures the DNS domains used for the DHCP server. Refer to the `--domain` option of the `dnsmasq` utility for more information. Like *nova-network* itself, this purpose is deprecated.

Possible values:

- Any string that is a valid domain name.

Related options:

- `use_neutron`

Table 25: Deprecated Variations

Group	Name
DEFAULT	dhcp_domain

vendordata_jsonfile_path

Type string

Default `<None>`

Cloud providers may store custom data in vendor data file that will then be available to the instances via the metadata service, and to the rendering of config-drive. The default class for this, JsonFileVendorData, loads this information from a JSON file, whose path is configured by this option. If there is no path set by this option, the class returns an empty dictionary.

Note that when using this to provide static vendor data to a configuration drive, the nova-compute service must be configured with this option and the file must be accessible from the nova-compute host.

Possible values:

- Any string representing the path to the data file, or an empty string (default).

Table 26: Deprecated Variations

Group	Name
DEFAULT	vendordata_jsonfile_path

max_limit

Type integer

Default `1000`

Minimum Value 0

As a query can potentially return many thousands of items, you can limit the maximum number of items in a single response by setting this option.

Table 27: Deprecated Variations

Group	Name
DEFAULT	osapi_max_limit

compute_link_prefix

Type string

Default `<None>`

This string is prepended to the normal URL that is returned in links to the OpenStack Compute API. If it is empty (the default), the URLs are returned unchanged.

Possible values:

- Any string, including an empty string (the default).

Table 28: Deprecated Variations

Group	Name
DEFAULT	osapi_compute_link_prefix

glance_link_prefix

Type string

Default `<None>`

This string is prepended to the normal URL that is returned in links to Glance resources. If it is empty (the default), the URLs are returned unchanged.

Possible values:

- Any string, including an empty string (the default).

Table 29: Deprecated Variations

Group	Name
DEFAULT	osapi_glance_link_prefix

instance_list_per_project_cells

Type boolean

Default `false`

When enabled, this will cause the API to only query cell databases in which the tenant has mapped instances. This requires an additional (fast) query in the API database before each list, but also (potentially) limits the number of cell databases that must be queried to provide the result. If you have a small number of cells, or tenants are likely to have instances in all cells, then this should be False. If you have many cells, especially if you confine tenants to a small subset of those cells, this should be True.

instance_list_cells_batch_strategy

Type string

Default `distributed`

Valid Values distributed, fixed

This controls the method by which the API queries cell databases in smaller batches during large instance list operations. If batching is performed, a large instance list operation will request some fraction of the overall API limit from each cell database initially, and will re-request that same batch size as records are consumed (returned) from each cell as necessary. Larger batches mean less chattiness between the API and the database, but potentially more wasted effort processing the results from the database which will not be returned to the user. Any strategy will yield a batch size of at least 100 records, to avoid a user causing many tiny database queries in their request.

Related options:

- instance_list_cells_batch_fixed_size
- max_limit

Possible values

distributed Divide the limit requested by the user by the number of cells in the system. This requires counting the cells in the system initially, which will not be refreshed until service restart or SIGHUP. The actual batch size will be increased by 10% over the result of ($limit / $num_cells).

fixed Request fixed-size batches from each cell, as defined by `instance_list_cells_batch_fixed_size`. If the limit is smaller than the batch size, the limit will be used instead. If you do not wish batching to be used at all, setting the fixed size equal to the `max_limit` value will cause only one request per cell database to be issued.

instance_list_cells_batch_fixed_size

Type integer

Default `100`

Minimum Value 100

This controls the batch size of instances requested from each cell database if `instance_list_cells_batch_strategy` is set to `fixed`. This integral value will define the limit issued to each cell every time a batch of instances is requested, regardless of the number of cells in the system or any other factors. Per the general logic called out in the documentation for `instance_list_cells_batch_strategy`, the minimum value for this is 100 records per batch.

Related options:

- instance_list_cells_batch_strategy
- max_limit

list_records_by_skipping_down_cells

Type boolean

Default `true`

When set to False, this will cause the API to return a 500 error if there is an infrastructure failure like non-responsive cells. If you want the API to skip the down cells and return the results from the up cells set this option to True.

Note that from API microversion 2.69 there could be transient conditions in the deployment where certain records are not available and the results could be partial for certain requests containing those records. In those cases this option will be ignored. See Handling Down Cells section of the Compute API guide (https://docs.openstack.org/api-guide/compute/down_cells.html) for more information.

use_neutron_default_nets

Type boolean

Default `false`

When True, the TenantNetworkController will query the Neutron API to get the default networks to use.

Related options:

- neutron_default_tenant_id

Table 30: Deprecated Variations

Group	Name
DEFAULT	use_neutron_default_nets

`neutron_default_tenant_id`

Type string

Default `default`

Tenant ID for getting the default network from Neutron API (also referred in some places as the project ID) to use.

Related options:

- use_neutron_default_nets

Table 31: Deprecated Variations

Group	Name
DEFAULT	neutron_default_tenant_id

`enable_instance_password`

Type boolean

Default `true`

Enables returning of the instance password by the relevant server API calls such as create, rebuild, evacuate, or rescue. If the hypervisor does not support password injection, then the password returned will not be correct, so if your hypervisor does not support password injection, set this to False.

Table 32: Deprecated Variations

Group	Name
DEFAULT	enable_instance_password

api_database

The *Nova API Database* is a separate database which is used for information which is used across *cells*. This database is mandatory since the Mitaka release (13.0.0).

This group should **not** be configured for the `nova-compute` service.

`connection`

Type string

Default `<None>`

The SQLAlchemy connection string to use to connect to the database. Do not set this for the `nova-compute` service.

connection_parameters

Type string

Default `''`

Optional URL parameters to append onto the connection URL at connect time; specify as param1=value1&param2=value2&

sqlite_synchronous

Type boolean

Default `true`

If True, SQLite uses synchronous mode.

slave_connection

Type string

Default `<None>`

The SQLAlchemy connection string to use to connect to the slave database.

mysql_sql_mode

Type string

Default `TRADITIONAL`

The SQL mode to be used for MySQL sessions. This option, including the default, overrides any server-set SQL mode. To use whatever SQL mode is set by the server configuration, set this to no value. Example: mysql_sql_mode=

connection_recycle_time

Type integer

Default `3600`

Connections which have been present in the connection pool longer than this number of seconds will be replaced with a new one the next time they are checked out from the pool.

Table 33: Deprecated Variations

Group	Name
api_database	idle_timeout

max_pool_size

Type integer

Default `<None>`

Maximum number of SQL connections to keep open in a pool. Setting a value of 0 indicates no limit.

max_retries

Type integer

Default `10`

Maximum number of database connection retries during startup. Set to -1 to specify an infinite retry count.

retry_interval

Type integer

Default `10`

Interval between retries of opening a SQL connection.

max_overflow

Type integer

Default `<None>`

If set, use this value for max_overflow with SQLAlchemy.

connection_debug

Type integer

Default `0`

Verbosity of SQL debugging information: 0=None, 100=Everything.

connection_trace

Type boolean

Default `false`

Add Python stack traces to SQL as comment strings.

pool_timeout

Type integer

Default `<None>`

If set, use this value for pool_timeout with SQLAlchemy.

barbican

barbican_endpoint

Type string

Default `<None>`

Use this endpoint to connect to Barbican, for example: http://localhost:9311/

barbican_api_version

Type string

Default `<None>`

Version of the Barbican API, for example: v1

auth_endpoint

Type string

Default `http://localhost/identity/v3`

Use this endpoint to connect to Keystone

Table 34: Deprecated Variations

Group	Name
key_manager	auth_url

retry_delay

Type integer

Default `1`

Number of seconds to wait before retrying poll for key creation completion

number_of_retries

Type integer

Default `60`

Number of times to retry poll for key creation completion

verify_ssl

Type boolean

Default `true`

Specifies if insecure TLS (https) requests. If False, the servers certificate will not be validated, if True, we can set the verify_ssl_path config meanwhile.

verify_ssl_path

Type string

Default `<None>`

A path to a bundle or CA certs to check against, or None for requests to attempt to locate and use certificates which verify_ssh is True. If verify_ssl is False, this is ignored.

barbican_endpoint_type

Type string

Default `public`

Valid Values public, internal, admin

Specifies the type of endpoint. Allowed values are: public, private, and admin

cache

config_prefix

Type string

Default `cache.oslo`

Prefix for building the configuration dictionary for the cache region. This should not need to be changed unless there is another dogpile.cache region with the same configuration name.

expiration_time

Type integer

Default `600`

Default TTL, in seconds, for any cached item in the dogpile.cache region. This applies to any cached method that doesnt have an explicit cache expiration time defined for it.

backend

Type string

Default `dogpile.cache.null`

Valid Values oslo_cache.memcache_pool, oslo_cache.dict, oslo_cache.mongo, oslo_cache.etcd3gw, dogpile.cache.memcached, dogpile.cache.pylibmc, dogpile.cache.bmemcached, dogpile.cache.dbm, dogpile.cache.redis, dogpile.cache.memory, dogpile.cache.memory_pickle, dogpile.cache.null

Cache backend module. For eventlet-based or environments with hundreds of threaded servers, Memcache with pooling (oslo_cache.memcache_pool) is recommended. For environments with less than 100 threaded servers, Memcached (dogpile.cache.memcached) or Redis (dogpile.cache.redis) is recommended. Test environments with a single instance of the server can use the dogpile.cache.memory backend.

backend_argument

Type multi-valued

Default `''`

Arguments supplied to the backend module. Specify this option once per argument to be passed to the dogpile.cache backend. Example format: <argname>:<value>.

proxies

Type list

Default `''`

Proxy classes to import that will affect the way the dogpile.cache backend functions. See the dogpile.cache documentation on changing-backend-behavior.

enabled

Type boolean

Default `false`

Global toggle for caching.

debug_cache_backend

Type boolean

Default `false`

Extra debugging from the cache backend (cache keys, get/set/delete/etc calls). This is only really useful if you need to see the specific cache-backend get/set/delete calls with the keys/values. Typically this should be left set to false.

memcache_servers

Type list

Default `localhost:11211`

Memcache servers in the format of host:port. (dogpile.cache.memcache and oslo_cache.memcache_pool backends only).

memcache_dead_retry

Type integer

Default `300`

Number of seconds memcached server is considered dead before it is tried again. (dogpile.cache.memcache and oslo_cache.memcache_pool backends only).

memcache_socket_timeout

Type floating point

Default `1.0`

Timeout in seconds for every call to a server. (dogpile.cache.memcache and oslo_cache.memcache_pool backends only).

memcache_pool_maxsize

Type integer

Default `10`

Max total number of open connections to every memcached server. (oslo_cache.memcache_pool backend only).

memcache_pool_unused_timeout

Type integer

Default `60`

Number of seconds a connection to memcached is held unused in the pool before it is closed. (oslo_cache.memcache_pool backend only).

memcache_pool_connection_get_timeout

Type integer

Default `10`

Number of seconds that an operation will wait to get a memcache client connection.

tls_enabled

Type boolean

Default `false`

Global toggle for TLS usage when comunicating with the caching servers.

tls_cafile

Type string

Default `<None>`

Path to a file of concatenated CA certificates in PEM format necessary to establish the caching servers authenticity. If tls_enabled is False, this option is ignored.

tls_certfile

Type string

Default `<None>`

Path to a single file in PEM format containing the clients certificate as well as any number of CA certificates needed to establish the certificates authenticity. This file is only required when client side authentication is necessary. If tls_enabled is False, this option is ignored.

tls_keyfile

Type string

Default `<None>`

Path to a single file containing the clients private key in. Otherwhise the private key will be taken from the file specified in tls_certfile. If tls_enabled is False, this option is ignored.

`tls_allowed_ciphers`

Type string

Default `<None>`

Set the available ciphers for sockets created with the TLS context. It should be a string in the OpenSSL cipher list format. If not specified, all OpenSSL enabled ciphers will be available.

cinder

`catalog_info`

Type string

Default `volumev3::publicURL`

Info to match when looking for cinder in the service catalog.

The `<service_name>` is optional and omitted by default since it should not be necessary in most deployments.

Possible values:

- Format is separated values of the form: <service_type>:<service_name>:<endpoint_type>

Note: Nova does not support the Cinder v2 API since the Nova 17.0.0 Queens release.

Related options:

- endpoint_template - Setting this option will override catalog_info

`endpoint_template`

Type string

Default `<None>`

If this option is set then it will override service catalog lookup with this template for cinder endpoint

Possible values:

- URL for cinder endpoint API e.g. http://localhost:8776/v3/%(project_id)s

Note: Nova does not support the Cinder v2 API since the Nova 17.0.0 Queens release.

Related options:

- catalog_info - If endpoint_template is not set, catalog_info will be used.

`os_region_name`

Type string

Default `<None>`

Region name of this node. This is used when picking the URL in the service catalog.

Possible values:

- Any string representing region name

http_retries

Type integer

Default 3

Minimum Value 0

Number of times cinderclient should retry on any failed http call. 0 means connection is attempted only once. Setting it to any positive integer means that on failure connection is retried that many times e.g. setting it to 3 means total attempts to connect will be 4.

Possible values:

- Any integer value. 0 means connection is attempted only once

cross_az_attach

Type boolean

Default true

Allow attach between instance and volume in different availability zones.

If False, volumes attached to an instance must be in the same availability zone in Cinder as the instance availability zone in Nova. This also means care should be taken when booting an instance from a volume where source is not volume because Nova will attempt to create a volume using the same availability zone as what is assigned to the instance. If that AZ is not in Cinder (or allow_availability_zone_fallback=False in cinder.conf), the volume create request will fail and the instance will fail the build request. By default there is no availability zone restriction on volume attach.

cafile

Type string

Default <None>

PEM encoded Certificate Authority to use when verifying HTTPs connections.

certfile

Type string

Default <None>

PEM encoded client certificate cert file

keyfile

Type string

Default <None>

PEM encoded client certificate key file

insecure

Type boolean

Default false

Verify HTTPS connections.

timeout

Type integer

Default <None>

Timeout value for http requests

collect_timing

Type boolean

Default `false`

Collect per-API call timing information.

split_loggers

Type boolean

Default `false`

Log requests to multiple loggers.

auth_type

Type unknown type

Default `<None>`

Authentication type to load

Table 35: Deprecated Variations

Group	Name
cinder	auth_plugin

auth_section

Type unknown type

Default `<None>`

Config Section from which to load plugin specific options

auth_url

Type unknown type

Default `<None>`

Authentication URL

system_scope

Type unknown type

Default `<None>`

Scope for system operations

domain_id

Type unknown type

Default `<None>`

Domain ID to scope to

domain_name

Type unknown type

Default `<None>`

Domain name to scope to

project_id

Type unknown type

Default <None>

Project ID to scope to

project_name

Type unknown type

Default <None>

Project name to scope to

project_domain_id

Type unknown type

Default <None>

Domain ID containing project

project_domain_name

Type unknown type

Default <None>

Domain name containing project

trust_id

Type unknown type

Default <None>

Trust ID

default_domain_id

Type unknown type

Default <None>

Optional domain ID to use with v3 and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

default_domain_name

Type unknown type

Default <None>

Optional domain name to use with v3 API and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

user_id

Type unknown type

Default <None>

User ID

username

Type unknown type

Default <None>

Username

Table 36: Deprecated Variations

Group	Name
cinder	user-name
cinder	user_name

user_domain_id

Type unknown type

Default `<None>`

Users domain id

user_domain_name

Type unknown type

Default `<None>`

Users domain name

password

Type unknown type

Default `<None>`

Users password

tenant_id

Type unknown type

Default `<None>`

Tenant ID

tenant_name

Type unknown type

Default `<None>`

Tenant Name

compute

consecutive_build_service_disable_threshold

Type integer

Default `10`

Enables reporting of build failures to the scheduler.

Any nonzero value will enable sending build failure statistics to the scheduler for use by the BuildFailureWeigher.

Possible values:

- Any positive integer enables reporting build failures.
- Zero to disable reporting build failures.

Related options:

- [filter_scheduler]/build_failure_weight_multiplier

shutdown_retry_interval

Type integer

Default 10

Minimum Value 1

Time to wait in seconds before resending an ACPI shutdown signal to instances.

The overall time to wait is set by shutdown_timeout.

Possible values:

- Any integer greater than 0 in seconds

Related options:

- shutdown_timeout

resource_provider_association_refresh

Type integer

Default 300

Minimum Value 0

Mutable This option can be changed without restarting.

Interval for updating nova-compute-side cache of the compute node resource providers inventories, aggregates, and traits.

This option specifies the number of seconds between attempts to update a providers inventories, aggregates and traits in the local cache of the compute node.

A value of zero disables cache refresh completely.

The cache can be cleared manually at any time by sending SIGHUP to the compute process, causing it to be repopulated the next time the data is accessed.

Possible values:

- Any positive integer in seconds, or zero to disable refresh.

cpu_shared_set

Type string

Default <None>

Mask of host CPUs that can be used for VCPU resources and offloaded emulator threads.

The behavior of this option depends on the definition of the deprecated vcpu_pin_set option.

- If vcpu_pin_set is not defined, [compute] cpu_shared_set will be be used to provide VCPU inventory and to determine the host CPUs that unpinned instances can be scheduled to. It will also be used to determine the host CPUS that instance emulator threads should be offloaded to for instances configured with the share emulator thread policy (hw:emulator_threads_policy=share).
- If vcpu_pin_set is defined, [compute] cpu_shared_set will only be used to determine the host CPUs that instance emulator threads should be offloaded to for instances configured with the share emulator thread policy (hw:emulator_threads_policy=share). vcpu_pin_set will be used

to provide VCPU inventory and to determine the host CPUs that both pinned and unpinned instances can be scheduled to.

This behavior will be simplified in a future release when vcpu_pin_set is removed.

Possible values:

- A comma-separated list of physical CPU numbers that instance VCPUs can be allocated from. Each element should be either a single CPU number, a range of CPU numbers, or a caret followed by a CPU number to be excluded from a previous range. For example:

```
cpu_shared_set = "4-12,^8,15"
```

Related options:

- [compute] cpu_dedicated_set: This is the counterpart option for defining where PCPU resources should be allocated from.
- vcpu_pin_set: A legacy option whose definition may change the behavior of this option.

cpu_dedicated_set

Type string

Default <None>

Mask of host CPUs that can be used for PCPU resources.

The behavior of this option affects the behavior of the deprecated vcpu_pin_set option.

- If this option is defined, defining vcpu_pin_set will result in an error.
- If this option is not defined, vcpu_pin_set will be used to determine inventory for VCPU resources and to limit the host CPUs that both pinned and unpinned instances can be scheduled to.

This behavior will be simplified in a future release when vcpu_pin_set is removed.

Possible values:

- A comma-separated list of physical CPU numbers that instance VCPUs can be allocated from. Each element should be either a single CPU number, a range of CPU numbers, or a caret followed by a CPU number to be excluded from a previous range. For example:

```
cpu_dedicated_set = "4-12,^8,15"
```

Related options:

- [compute] cpu_shared_set: This is the counterpart option for defining where VCPU resources should be allocated from.
- vcpu_pin_set: A legacy option that this option partially replaces.

live_migration_wait_for_vif_plug

Type boolean

Default true

Determine if the source compute host should wait for a network-vif-plugged event from the (neutron) networking service before starting the actual transfer of the guest to the destination compute host.

Note that this option is read on the destination host of a live migration. If you set this option the same on all of your compute hosts, which you should do if you use the same networking backend universally, you do not have to worry about this.

Before starting the transfer of the guest, some setup occurs on the destination compute host, including plugging virtual interfaces. Depending on the networking backend **on the destination host**, a `network-vif-plugged` event may be triggered and then received on the source compute host and the source compute can wait for that event to ensure networking is set up on the destination host before starting the guest transfer in the hypervisor.

Note: The compute service cannot reliably determine which types of virtual interfaces (`port.binding:vif_type`) will send `network-vif-plugged` events without an accompanying port `binding:host_id` change. Open vSwitch and linuxbridge should be OK, but OpenDaylight is at least one known backend that will not currently work in this case, see bug https://launchpad.net/bugs/1755890 for more details.

Possible values:

- True: wait for `network-vif-plugged` events before starting guest transfer
- False: do not wait for `network-vif-plugged` events before starting guest transfer (this is the legacy behavior)

Related options:

- [DEFAULT]/vif_plugging_is_fatal: if `live_migration_wait_for_vif_plug` is True and `vif_plugging_timeout` is greater than 0, and a timeout is reached, the live migration process will fail with an error but the guest transfer will not have started to the destination host
- [DEFAULT]/vif_plugging_timeout: if `live_migration_wait_for_vif_plug` is True, this controls the amount of time to wait before timing out and either failing if `vif_plugging_is_fatal` is True, or simply continuing with the live migration

`max_concurrent_disk_ops`

Type integer

Default `0`

Minimum Value 0

Number of concurrent disk-IO-intensive operations (glance image downloads, image format conversions, etc.) that we will do in parallel. If this is set too high then response time suffers. The default value of 0 means no limit.

`max_disk_devices_to_attach`

Type integer

Default `-1`

Minimum Value -1

Maximum number of disk devices allowed to attach to a single server. Note that the number of disks supported by an server depends on the bus used. For example, the `ide` disk bus is limited to 4 attached devices. The configured maximum is enforced during server create, rebuild, evacuate, unshelve, live migrate, and attach volume.

Usually, disk bus is determined automatically from the device type or disk device, and the virtualization type. However, disk bus can also be specified via a block device mapping or an image property. See the `disk_bus` field in *Block Device Mapping in Nova* for more information about specifying disk bus in a block device mapping, and see https://docs.openstack.org/glance/latest/admin/useful-image-properties.html for more information about the `hw_disk_bus` image property.

Operators changing the `[compute]/max_disk_devices_to_attach` on a compute service that is hosting servers should be aware that it could cause rebuilds to fail, if the maximum is decreased lower than the number of devices already attached to servers. For example, if server A has 26 devices attached and an operators changes `[compute]/max_disk_devices_to_attach` to 20, a request to rebuild server A will fail and go into ERROR state because 26 devices are already attached and exceed the new configured maximum of 20.

Operators setting `[compute]/max_disk_devices_to_attach` should also be aware that during a cold migration, the configured maximum is only enforced in-place and the destination is not checked before the move. This means if an operator has set a maximum of 26 on compute host A and a maximum of 20 on compute host B, a cold migration of a server with 26 attached devices from compute host A to compute host B will succeed. Then, once the server is on compute host B, a subsequent request to rebuild the server will fail and go into ERROR state because 26 devices are already attached and exceed the configured maximum of 20 on compute host B.

The configured maximum is not enforced on shelved offloaded servers, as they have no compute host.

> **Warning:** If this option is set to 0, the `nova-compute` service will fail to start, as 0 disk devices is an invalid configuration that would prevent instances from being able to boot.

Possible values:

- -1 means unlimited
- Any integer >= 1 represents the maximum allowed. A value of 0 will cause the `nova-compute` service to fail to start, as 0 disk devices is an invalid configuration that would prevent instances from being able to boot.

conductor

Options under this group are used to define Conductors communication, which manager should be act as a proxy between computes and database, and finally, how many worker processes will be used.

`workers`

Type integer

Default `<None>`

Number of workers for OpenStack Conductor service. The default will be the number of CPUs available.

console

Options under this group allow to tune the configuration of the console proxy service.

Note: in configuration of every compute is a `console_host` option, which allows to select the console proxy service to connect to.

`allowed_origins`

Type list

Default `''`

Adds list of allowed origins to the console websocket proxy to allow connections from other origin hostnames. Websocket proxy matches the host header with the origin header to prevent cross-site requests. This list specifies if any there are values other than host are allowed in the origin header.

Possible values:

- A list where each element is an allowed origin hostnames, else an empty list

Table 37: Deprecated Variations

Group	Name
DEFAULT	console_allowed_origins

consoleauth

token_ttl

Type integer

Default `600`

Minimum Value 0

The lifetime of a console auth token (in seconds).

A console auth token is used in authorizing console access for a user. Once the auth token time to live count has elapsed, the token is considered expired. Expired tokens are then deleted.

Table 38: Deprecated Variations

Group	Name
DEFAULT	console_token_ttl

cors

allowed_origin

Type list

Default `<None>`

Indicate whether this resource may be shared with the domain received in the requests origin header. Format: <protocol>://<host>[:<port>], no trailing slash. Example: https://horizon.example.com

allow_credentials

Type boolean

Default `true`

Indicate that the actual request can include user credentials

expose_headers

Type list

Default `X-Auth-Token,X-Openstack-Request-Id,X-Subject-Token, X-Service-Token`

Indicate which headers are safe to expose to the API. Defaults to HTTP Simple Headers.

max_age

Type integer

Default `3600`

Maximum cache age of CORS preflight requests.

allow_methods

Type list

Default `GET,PUT,POST,DELETE,PATCH`

Indicate which methods can be used during the actual request.

allow_headers

Type list

Default `X-Auth-Token,X-Openstack-Request-Id,X-Identity-Status, X-Roles,X-Service-Catalog,X-User-Id,X-Tenant-Id`

Indicate which header field names may be used during the actual request.

database

sqlite_synchronous

Type boolean

Default `true`

If True, SQLite uses synchronous mode.

Table 39: Deprecated Variations

Group	Name
DEFAULT	sqlite_synchronous

backend

Type string

Default `sqlalchemy`

The back end to use for the database.

Table 40: Deprecated Variations

Group	Name
DEFAULT	db_backend

connection

Type string

Default `<None>`

The SQLAlchemy connection string to use to connect to the database.

Table 41: Deprecated Variations

Group	Name
DEFAULT	sql_connection
DATABASE	sql_connection
sql	connection

slave_connection

Type string

Default `<None>`

The SQLAlchemy connection string to use to connect to the slave database.

mysql_sql_mode

Type string

Default `TRADITIONAL`

The SQL mode to be used for MySQL sessions. This option, including the default, overrides any server-set SQL mode. To use whatever SQL mode is set by the server configuration, set this to no value. Example: mysql_sql_mode=

mysql_enable_ndb

Type boolean

Default `false`

If True, transparently enables support for handling MySQL Cluster (NDB).

connection_recycle_time

Type integer

Default `3600`

Connections which have been present in the connection pool longer than this number of seconds will be replaced with a new one the next time they are checked out from the pool.

Table 42: Deprecated Variations

Group	Name
DATABASE	idle_timeout
database	idle_timeout
DEFAULT	sql_idle_timeout
DATABASE	sql_idle_timeout
sql	idle_timeout

max_pool_size

Type integer

Default `5`

Maximum number of SQL connections to keep open in a pool. Setting a value of 0 indicates no limit.

Table 43: Deprecated Variations

Group	Name
DEFAULT	sql_max_pool_size
DATABASE	sql_max_pool_size

max_retries

Type integer

Default `10`

Maximum number of database connection retries during startup. Set to -1 to specify an infinite retry count.

Table 44: Deprecated Variations

Group	Name
DEFAULT	sql_max_retries
DATABASE	sql_max_retries

retry_interval

Type integer

Default `10`

Interval between retries of opening a SQL connection.

Table 45: Deprecated Variations

Group	Name
DEFAULT	sql_retry_interval
DATABASE	reconnect_interval

max_overflow

Type integer

Default `50`

If set, use this value for max_overflow with SQLAlchemy.

Table 46: Deprecated Variations

Group	Name
DEFAULT	sql_max_overflow
DATABASE	sqlalchemy_max_overflow

connection_debug

Type integer

Default `0`

Minimum Value 0

Maximum Value 100

Verbosity of SQL debugging information: 0=None, 100=Everything.

Table 47: Deprecated Variations

Group	Name
DEFAULT	sql_connection_debug

connection_trace

Type boolean

Default `false`

Add Python stack traces to SQL as comment strings.

Table 48: Deprecated Variations

Group	Name
DEFAULT	sql_connection_trace

pool_timeout

Type integer

Default <None>

If set, use this value for pool_timeout with SQLAlchemy.

Table 49: Deprecated Variations

Group	Name
DATABASE	sqlalchemy_pool_timeout

use_db_reconnect

Type boolean

Default false

Enable the experimental use of database reconnect on connection lost.

db_retry_interval

Type integer

Default 1

Seconds between retries of a database transaction.

db_inc_retry_interval

Type boolean

Default true

If True, increases the interval between retries of a database operation up to db_max_retry_interval.

db_max_retry_interval

Type integer

Default 10

If db_inc_retry_interval is set, the maximum seconds between retries of a database operation.

db_max_retries

Type integer

Default 20

Maximum retries in case of connection error or deadlock error before error is raised. Set to -1 to specify an infinite retry count.

connection_parameters

Type string

Default ''

Optional URL parameters to append onto the connection URL at connect time; specify as param1=value1&param2=value2&

use_tpool

Type boolean

Default `false`

Enable the experimental use of thread pooling for all DB API calls

Table 50: Deprecated Variations

Group	Name
DEFAULT	dbapi_use_tpool

devices

enabled_vgpu_types

Type list

Default `''`

The vGPU types enabled in the compute node.

Some pGPUs (e.g. NVIDIA GRID K1) support different vGPU types. User can use this option to specify a list of enabled vGPU types that may be assigned to a guest instance. But please note that Nova only supports a single type in the Queens release. If more than one vGPU type is specified (as a comma-separated list), only the first one will be used. An example is as the following:

```
[devices]
enabled_vgpu_types = GRID K100,Intel GVT-g,MxGPU.2,nvidia-11
```

ephemeral_storage_encryption

enabled

Type boolean

Default `false`

Enables/disables LVM ephemeral storage encryption.

cipher

Type string

Default `aes-xts-plain64`

Cipher-mode string to be used.

The cipher and mode to be used to encrypt ephemeral storage. The set of cipher-mode combinations available depends on kernel support. According to the dm-crypt documentation, the cipher is expected to be in the format: <cipher>-<chainmode>-<ivmode>.

Possible values:

- Any crypto option listed in `/proc/crypto`.

key_size

Type integer

Default `512`

Minimum Value 1

Encryption key length in bits.

The bit length of the encryption key to be used to encrypt ephemeral storage. In XTS mode only half of the bits are used for encryption key.

filter_scheduler

`host_subset_size`

Type integer

Default `1`

Minimum Value 1

Size of subset of best hosts selected by scheduler.

New instances will be scheduled on a host chosen randomly from a subset of the N best hosts, where N is the value set by this option.

Setting this to a value greater than 1 will reduce the chance that multiple scheduler processes handling similar requests will select the same host, creating a potential race condition. By selecting a host randomly from the N hosts that best fit the request, the chance of a conflict is reduced. However, the higher you set this value, the less optimal the chosen host may be for a given request.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect.

Possible values:

- An integer, where the integer corresponds to the size of a host subset. Any integer is valid, although any value less than 1 will be treated as 1

Table 51: Deprecated Variations

Group	Name
DEFAULT	scheduler_host_subset_size

`max_io_ops_per_host`

Type integer

Default `8`

The number of instances that can be actively performing IO on a host.

Instances performing IO includes those in the following states: build, resize, snapshot, migrate, rescue, unshelve.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect. Also note that this setting only affects scheduling if the io_ops_filter filter is enabled.

Possible values:

- An integer, where the integer corresponds to the max number of instances that can be actively performing IO on any given host.

Table 52: Deprecated Variations

Group	Name
DEFAULT	max_io_ops_per_host

max_instances_per_host

Type integer

Default `50`

Minimum Value 1

Maximum number of instances that can exist on a host.

If you need to limit the number of instances on any given host, set this option to the maximum number of instances you want to allow. The NumInstancesFilter and AggregateNumInstancesFilter will reject any host that has at least as many instances as this options value.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect. Also note that this setting only affects scheduling if the `NumInstancesFilter` or `AggregateNumInstancesFilter` filter is enabled.

Possible values:

- An integer, where the integer corresponds to the max instances that can be scheduled on a host.

Table 53: Deprecated Variations

Group	Name
DEFAULT	max_instances_per_host

track_instance_changes

Type boolean

Default `true`

Enable querying of individual hosts for instance information.

The scheduler may need information about the instances on a host in order to evaluate its filters and weighers. The most common need for this information is for the (anti-)affinity filters, which need to choose a host based on the instances already running on a host.

If the configured filters and weighers do not need this information, disabling this option will improve performance. It may also be disabled when the tracking overhead proves too heavy, although this will cause classes requiring host usage data to query the database on each request instead.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect.

NOTE: In a multi-cell (v2) setup where the cell MQ is separated from the top-level, computes cannot directly communicate with the scheduler. Thus, this option cannot be enabled in that scenario. See also the [workarounds]/disable_group_policy_check_upcall option.

Table 54: Deprecated Variations

Group	Name
DEFAULT	scheduler_tracks_instance_changes

available_filters

Type multi-valued

Default `nova.scheduler.filters.all_filters`

Filters that the scheduler can use.

An unordered list of the filter classes the nova scheduler may apply. Only the filters specified in the enabled_filters option will be used, but any filter appearing in that option must also be included in this list.

By default, this is set to all filters that are included with nova.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect.

Possible values:

- A list of zero or more strings, where each string corresponds to the name of a filter that may be used for selecting a host

Related options:

- enabled_filters

Table 55: Deprecated Variations

Group	Name
DEFAULT	scheduler_available_filters

enabled_filters

Type list

Default `AvailabilityZoneFilter,ComputeFilter,ComputeCapabilitiesFilter, ImagePropertiesFilter,ServerGroupAntiAffinityFilter, ServerGroupAffinityFilter`

Filters that the scheduler will use.

An ordered list of filter class names that will be used for filtering hosts. These filters will be applied in the order they are listed so place your most restrictive filters first to make the filtering process more efficient.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect.

Possible values:

- A list of zero or more strings, where each string corresponds to the name of a filter to be used for selecting a host

Related options:

- All of the filters in this option *must* be present in the available_filters option, or a SchedulerHostFilterNotFound exception will be raised.

Table 56: Deprecated Variations

Group	Name
DEFAULT	scheduler_default_filters

weight_classes

Type list

Default `nova.scheduler.weights.all_weighers`

Weighers that the scheduler will use.

Only hosts which pass the filters are weighed. The weight for any host starts at 0, and the weighers order these hosts by adding to or subtracting from the weight assigned by the previous weigher. Weights may become negative. An instance will be scheduled to one of the N most-weighted hosts, where N is scheduler_host_subset_size.

By default, this is set to all weighers that are included with Nova.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect.

Possible values:

- A list of zero or more strings, where each string corresponds to the name of a weigher that will be used for selecting a host

Table 57: Deprecated Variations

Group	Name
DEFAULT	scheduler_weight_classes

ram_weight_multiplier

Type floating point

Default `1.0`

RAM weight multipler ratio.

This option determines how hosts with more or less available RAM are weighed. A positive value will result in the scheduler preferring hosts with more available RAM, and a negative number will result in the scheduler preferring hosts with less available RAM. Another way to look at it is that positive values for this option will tend to spread instances across many hosts, while negative values will tend to fill up (stack) hosts as much as possible before scheduling to a less-used host. The absolute value, whether positive or negative, controls how strong the RAM weigher is relative to other weighers.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect. Also note that this setting only affects scheduling if the ram weigher is enabled.

Possible values:

- An integer or float value, where the value corresponds to the multipler ratio for this weigher.

Table 58: Deprecated Variations

Group	Name
DEFAULT	ram_weight_multiplier

cpu_weight_multiplier

Type floating point

Default `1.0`

CPU weight multiplier ratio.

Multiplier used for weighting free vCPUs. Negative numbers indicate stacking rather than spreading.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect. Also note that this setting only affects scheduling if the cpu weigher is enabled.

Possible values:

- An integer or float value, where the value corresponds to the multipler ratio for this weigher.

Related options:

- `filter_scheduler.weight_classes`: This weigher must be added to list of enabled weight classes if the `weight_classes` setting is set to a non-default value.

disk_weight_multiplier

Type floating point

Default `1.0`

Disk weight multipler ratio.

Multiplier used for weighing free disk space. Negative numbers mean to stack vs spread.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect. Also note that this setting only affects scheduling if the disk weigher is enabled.

Possible values:

- An integer or float value, where the value corresponds to the multipler ratio for this weigher.

Table 59: Deprecated Variations

Group	Name
DEFAULT	disk_weight_multiplier

io_ops_weight_multiplier

Type floating point

Default `-1.0`

IO operations weight multipler ratio.

This option determines how hosts with differing workloads are weighed. Negative values, such as the default, will result in the scheduler preferring hosts with lighter workloads whereas positive values will prefer hosts with heavier workloads. Another way to look at it is that positive values for this option will tend to schedule instances onto hosts that are already busy, while negative values will tend to distribute the workload across more hosts. The absolute value, whether positive or negative, controls how strong the io_ops weigher is relative to other weighers.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect. Also note that this setting only affects scheduling if the io_ops weigher is enabled.

Possible values:

- An integer or float value, where the value corresponds to the multipler ratio for this weigher.

Table 60: Deprecated Variations

Group	Name
DEFAULT	io_ops_weight_multiplier

pci_weight_multiplier

Type floating point

Default `1.0`

Minimum Value 0.0

PCI device affinity weight multiplier.

The PCI device affinity weighter computes a weighting based on the number of PCI devices on the host and the number of PCI devices requested by the instance. The `NUMATopologyFilter` filter must be enabled for this to have any significance. For more information, refer to the filter documentation:

https://docs.openstack.org/nova/latest/user/filter-scheduler.html

Possible values:

- A positive integer or float value, where the value corresponds to the multiplier ratio for this weigher.

soft_affinity_weight_multiplier

Type floating point

Default `1.0`

Minimum Value 0.0

Multiplier used for weighing hosts for group soft-affinity.

Possible values:

- A non-negative integer or float value, where the value corresponds to weight multiplier for hosts with group soft affinity.

soft_anti_affinity_weight_multiplier

Type floating point

Default `1.0`

Minimum Value 0.0

Multiplier used for weighing hosts for group soft-anti-affinity.

Possible values:

- A non-negative integer or float value, where the value corresponds to weight multiplier for hosts with group soft anti-affinity.

build_failure_weight_multiplier

Type floating point

Default `1000000.0`

Multiplier used for weighing hosts that have had recent build failures.

This option determines how much weight is placed on a compute node with recent build failures. Build failures may indicate a failing, misconfigured, or otherwise ailing compute node, and avoiding it during scheduling may be beneficial. The weight is inversely proportional to the number of recent build failures the compute node has experienced. This value should be set to some high value to offset weight given by other enabled weighers due to available resources. To disable weighing compute hosts by the number of recent failures, set this to zero.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect.

Possible values:

- An integer or float value, where the value corresponds to the multiplier ratio for this weigher.

Related options:

- [compute]/consecutive_build_service_disable_threshold - Must be nonzero for a compute to report data considered by this weigher.

shuffle_best_same_weighed_hosts

Type boolean

Default `false`

Enable spreading the instances between hosts with the same best weight.

Enabling it is beneficial for cases when host_subset_size is 1 (default), but there is a large number of hosts with same maximal weight. This scenario is common in Ironic deployments where there are typically many baremetal nodes with identical weights returned to the scheduler. In such case enabling this option will reduce contention and chances for rescheduling events. At the same time it will make the instance packing (even in unweighed case) less dense.

image_properties_default_architecture

Type string

Default `<None>`

Valid Values alpha, armv6, armv7l, armv7b, aarch64, cris, i686, ia64, lm32, m68k, microblaze, microblazeel, mips, mipsel, mips64, mips64el, openrisc, parisc, parisc64, ppc, ppcle, ppc64, ppc64le, ppcemb, s390, s390x, sh4, sh4eb, sparc, sparc64, unicore32, x86_64, xtensa, xtensaeb

The default architecture to be used when using the image properties filter.

When using the ImagePropertiesFilter, it is possible that you want to define a default architecture to make the user experience easier and avoid having something like x86_64 images landing on aarch64 compute nodes because the user did not specify the hw_architecture property in Glance.

Possible values:

- CPU Architectures such as x86_64, aarch64, s390x.

isolated_images

Type list

Default `''`

List of UUIDs for images that can only be run on certain hosts.

If there is a need to restrict some images to only run on certain designated hosts, list those image UUIDs here.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect. Also note that this setting only affects scheduling if the IsolatedHostsFilter filter is enabled.

Possible values:

- A list of UUID strings, where each string corresponds to the UUID of an image

Related options:

- scheduler/isolated_hosts
- scheduler/restrict_isolated_hosts_to_isolated_images

Table 61: Deprecated Variations

Group	Name
DEFAULT	isolated_images

isolated_hosts

Type list

Default `''`

List of hosts that can only run certain images.

If there is a need to restrict some images to only run on certain designated hosts, list those host names here.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect. Also note that this setting only affects scheduling if the IsolatedHostsFilter filter is enabled.

Possible values:

- A list of strings, where each string corresponds to the name of a host

Related options:

- scheduler/isolated_images
- scheduler/restrict_isolated_hosts_to_isolated_images

Table 62: Deprecated Variations

Group	Name
DEFAULT	isolated_hosts

restrict_isolated_hosts_to_isolated_images

Type boolean

Default `true`

Prevent non-isolated images from being built on isolated hosts.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect. Also note that this setting only affects scheduling if the IsolatedHostsFilter filter is enabled. Even then, this option doesnt affect the behavior of requests for isolated images, which will *always* be restricted to isolated hosts.

Related options:

- scheduler/isolated_images
- scheduler/isolated_hosts

Table 63: Deprecated Variations

Group	Name
DEFAULT	restrict_isolated_hosts_to_isolated_images

aggregate_image_properties_isolation_namespace

Type string

Default `<None>`

Image property namespace for use in the host aggregate.

Images and hosts can be configured so that certain images can only be scheduled to hosts in a particular aggregate. This is done with metadata values set on the host aggregate that are identified by beginning with the value of this option. If the host is part of an aggregate with such a metadata key, the image in the request spec must have the value of that metadata in its properties in order for the scheduler to consider the host as acceptable.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect. Also note that this setting only affects scheduling if the aggregate_image_properties_isolation filter is enabled.

Possible values:

- A string, where the string corresponds to an image property namespace

Related options:

- aggregate_image_properties_isolation_separator

Table 64: Deprecated Variations

Group	Name
DEFAULT	aggregate_image_properties_isolation_namespace

aggregate_image_properties_isolation_separator

Type string

Default .

Separator character(s) for image property namespace and name.

When using the aggregate_image_properties_isolation filter, the relevant metadata keys are prefixed with the namespace defined in the aggregate_image_properties_isolation_namespace configuration option plus a separator. This option defines the separator to be used.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect. Also note that this setting only affects scheduling if the aggregate_image_properties_isolation filter is enabled.

Possible values:

- A string, where the string corresponds to an image property namespace separator character

Related options:

- aggregate_image_properties_isolation_namespace

Table 65: Deprecated Variations

Group	Name
DEFAULT	aggregate_image_properties_isolation_separator

glance

Configuration options for the Image service

api_servers

Type list

Default `<None>`

List of glance api servers endpoints available to nova.

https is used for ssl-based glance api servers.

NOTE: The preferred mechanism for endpoint discovery is via keystoneauth1 loading options. Only use api_servers if you need multiple endpoints and are unable to use a load balancer for some reason.

Possible values:

- A list of any fully qualified url of the form scheme://hostname:port[/path] (i.e. http://10.0.1.0:9292 or https://my.glance.server/image).

num_retries

Type integer

Default `3`

Minimum Value 0

Enable glance operation retries.

Specifies the number of retries when uploading / downloading an image to / from glance. 0 means no retries.

allowed_direct_url_schemes

Type list

Default ''

List of url schemes that can be directly accessed.

This option specifies a list of url schemes that can be downloaded directly via the direct_url. This direct_URL can be fetched from Image metadata which can be used by nova to get the image more efficiently. nova-compute could benefit from this by invoking a copy when it has access to the same file system as glance.

Possible values:

- [file], Empty list (default)

Warning: This option is deprecated for removal since 17.0.0. Its value may be silently ignored in the future.

Reason This was originally added for the nova.image.download.file FileTransfer extension which was removed in the 16.0.0 Pike release. The nova.image.download.modules extension point is not maintained and there is no indication of its use in production clouds.

verify_glance_signatures

Type boolean

Default false

Enable image signature verification.

nova uses the image signature metadata from glance and verifies the signature of a signed image while downloading that image. If the image signature cannot be verified or if the image signature metadata is either incomplete or unavailable, then nova will not boot the image and instead will place the instance into an error state. This provides end users with stronger assurances of the integrity of the image data they are using to create servers.

Related options:

- The options in the *key_manager* group, as the key_manager is used for the signature validation.
- Both enable_certificate_validation and default_trusted_certificate_ids below depend on this option being enabled.

enable_certificate_validation

Type boolean

Default false

Enable certificate validation for image signature verification.

During image signature verification nova will first verify the validity of the images signing certificate using the set of trusted certificates associated with the instance. If certificate validation fails, signature verification will not be performed and the instance will be placed into an error state. This provides end users with stronger assurances that the image data is unmodified and trustworthy. If left disabled, image signature verification can still occur but the end user will not have any assurance that the signing certificate used to generate the image signature is still trustworthy.

Related options:

- This option only takes effect if verify_glance_signatures is enabled.
- The value of default_trusted_certificate_ids may be used when this option is enabled.

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason This option is intended to ease the transition for deployments leveraging image signature verification. The intended state long-term is for signature verification and certificate validation to always happen together.

default_trusted_certificate_ids

Type list

Default ''

List of certificate IDs for certificates that should be trusted.

May be used as a default list of trusted certificate IDs for certificate validation. The value of this option will be ignored if the user provides a list of trusted certificate IDs with an instance API request. The value of this option will be persisted with the instance data if signature verification and certificate validation are enabled and if the user did not provide an alternative list. If left empty when certificate validation is enabled the user must provide a list of trusted certificate IDs otherwise certificate validation will fail.

Related options:

- The value of this option may be used if both verify_glance_signatures and enable_certificate_validation are enabled.

debug

Type boolean

Default false

Enable or disable debug logging with glanceclient.

cafile

Type string

Default <None>

PEM encoded Certificate Authority to use when verifying HTTPs connections.

certfile

Type string

Default <None>

PEM encoded client certificate cert file

keyfile

Type string

Default <None>

PEM encoded client certificate key file

insecure

Type boolean

Default `false`

Verify HTTPS connections.

timeout

Type integer

Default `<None>`

Timeout value for http requests

collect_timing

Type boolean

Default `false`

Collect per-API call timing information.

split_loggers

Type boolean

Default `false`

Log requests to multiple loggers.

service_type

Type string

Default `image`

The default service_type for endpoint URL discovery.

service_name

Type string

Default `<None>`

The default service_name for endpoint URL discovery.

valid_interfaces

Type list

Default `internal,public`

List of interfaces, in order of preference, for endpoint URL.

region_name

Type string

Default `<None>`

The default region_name for endpoint URL discovery.

endpoint_override

Type string

Default `<None>`

Always use this endpoint URL for requests for this client. NOTE: The unversioned endpoint should be specified here; to request a particular API version, use the *version*, *min-version*, and/or *max-version* options.

connect_retries

Type integer

Default <None>

The maximum number of retries that should be attempted for connection errors.

connect_retry_delay

Type floating point

Default <None>

Delay (in seconds) between two retries for connection errors. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

status_code_retries

Type integer

Default <None>

The maximum number of retries that should be attempted for retriable HTTP status codes.

status_code_retry_delay

Type floating point

Default <None>

Delay (in seconds) between two retries for retriable status codes. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

guestfs

libguestfs is a set of tools for accessing and modifying virtual machine (VM) disk images. You can use this for viewing and editing files inside guests, scripting changes to VMs, monitoring disk used/free statistics, creating guests, P2V, V2V, performing backups, cloning VMs, building VMs, formatting disks and resizing disks.

debug

Type boolean

Default false

Enable/disables guestfs logging.

This configures guestfs to debug messages and push them to OpenStack logging system. When set to True, it traces libguestfs API calls and enable verbose debug messages. In order to use the above feature, libguestfs package must be installed.

Related options:

Since libguestfs access and modifies VMs managed by libvirt, below options should be set to give access to those VMs.

- libvirt.inject_key
- libvirt.inject_partition
- libvirt.inject_password

healthcheck

path

Type string

Default `/healthcheck`

The path to respond to healtcheck requests on.

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.

detailed

Type boolean

Default `false`

Show more detailed information as part of the response. Security note: Enabling this option may expose sensitive details about the service being monitored. Be sure to verify that it will not violate your security policies.

backends

Type list

Default `''`

Additional backends that can perform health checks and report that information back as part of a request.

disable_by_file_path

Type string

Default `<None>`

Check the presence of a file to determine if an application is running on a port. Used by DisableByFileHealthcheck plugin.

disable_by_file_paths

Type list

Default `''`

Check the presence of a file based on a port to determine if an application is running on a port. Expects a port:path list of strings. Used by DisableByFilesPortsHealthcheck plugin.

hyperv

The hyperv feature allows you to configure the Hyper-V hypervisor driver to be used within an OpenStack deployment.

dynamic_memory_ratio

Type floating point

Default `1.0`

Dynamic memory ratio

Enables dynamic memory allocation (ballooning) when set to a value greater than 1. The value expresses the ratio between the total RAM assigned to an instance and its startup RAM amount. For example a ratio of 2.0 for an instance with 1024MB of RAM implies 512MB of RAM allocated at startup.

Possible values:

- 1.0: Disables dynamic memory allocation (Default).
- Float values greater than 1.0: Enables allocation of total implied RAM divided by this value for startup.

enable_instance_metrics_collection

Type boolean

Default `false`

Enable instance metrics collection

Enables metrics collections for an instance by using Hyper-Vs metric APIs. Collected data can be retrieved by other apps and services, e.g.: Ceilometer.

instances_path_share

Type string

Default `''`

Instances path share

The name of a Windows share mapped to the instances_path dir and used by the resize feature to copy files to the target host. If left blank, an administrative share (hidden network share) will be used, looking for the same instances_path used locally.

Possible values:

- : An administrative share will be used (Default).
- Name of a Windows share.

Related options:

- instances_path: The directory which will be used if this option here is left blank.

limit_cpu_features

Type boolean

Default `false`

Limit CPU features

This flag is needed to support live migration to hosts with different CPU features and checked during instance creation in order to limit the CPU features used by the instance.

mounted_disk_query_retry_count

Type integer

Default `10`

Minimum Value 0

Mounted disk query retry count

The number of times to retry checking for a mounted disk. The query runs until the device can be found or the retry count is reached.

Possible values:

- Positive integer values. Values greater than 1 is recommended (Default: 10).

Related options:

- Time interval between disk mount retries is declared with mounted_disk_query_retry_interval option.

mounted_disk_query_retry_interval

Type integer

Default 5

Minimum Value 0

Mounted disk query retry interval

Interval between checks for a mounted disk, in seconds.

Possible values:

- Time in seconds (Default: 5).

Related options:

- This option is meaningful when the mounted_disk_query_retry_count is greater than 1.
- The retry loop runs with mounted_disk_query_retry_count and mounted_disk_query_retry_interval configuration options.

power_state_check_timeframe

Type integer

Default 60

Minimum Value 0

Power state check timeframe

The timeframe to be checked for instance power state changes. This option is used to fetch the state of the instance from Hyper-V through the WMI interface, within the specified timeframe.

Possible values:

- Timeframe in seconds (Default: 60).

power_state_event_polling_interval

Type integer

Default 2

Minimum Value 0

Power state event polling interval

Instance power state change event polling frequency. Sets the listener interval for power state events to the given value. This option enhances the internal lifecycle notifications of instances that reboot themselves. It is unlikely that an operator has to change this value.

Possible values:

- Time in seconds (Default: 2).

qemu_img_cmd

Type string

Default qemu-img.exe

qemu-img command

qemu-img is required for some of the image related operations like converting between different image types. You can get it from here: (http://qemu.weilnetz.de/) or you can install the Cloudbase OpenStack Hyper-V Compute Driver (https://cloudbase.it/openstack-hyperv-driver/) which automatically sets the proper path for this

config option. You can either give the full path of qemu-img.exe or set its path in the PATH environment variable and leave this option to the default value.

Possible values:

- Name of the qemu-img executable, in case it is in the same directory as the nova-compute service or its path is in the PATH environment variable (Default).
- Path of qemu-img command (DRIVELETTER:PATHTOQEMU-IMGCOMMAND).

Related options:

- If the config_drive_cdrom option is False, qemu-img will be used to convert the ISO to a VHD, otherwise the config drive will remain an ISO. To use config drive with Hyper-V, you must set the `mkisofs_cmd` value to the full path to an `mkisofs.exe` installation.

`vswitch_name`

Type string

Default `<None>`

External virtual switch name

The Hyper-V Virtual Switch is a software-based layer-2 Ethernet network switch that is available with the installation of the Hyper-V server role. The switch includes programmatically managed and extensible capabilities to connect virtual machines to both virtual networks and the physical network. In addition, Hyper-V Virtual Switch provides policy enforcement for security, isolation, and service levels. The vSwitch represented by this config option must be an external one (not internal or private).

Possible values:

- If not provided, the first of a list of available vswitches is used. This list is queried using WQL.
- Virtual switch name.

`wait_soft_reboot_seconds`

Type integer

Default `60`

Minimum Value 0

Wait soft reboot seconds

Number of seconds to wait for instance to shut down after soft reboot request is made. We fall back to hard reboot if instance does not shutdown within this window.

Possible values:

- Time in seconds (Default: 60).

`config_drive_cdrom`

Type boolean

Default `false`

Mount config drive as a CD drive.

OpenStack can be configured to write instance metadata to a config drive, which is then attached to the instance before it boots. The config drive can be attached as a disk drive (default) or as a CD drive.

Related options:

- This option is meaningful with `force_config_drive` option set to `True` or when the REST API call to create an instance will have `--config-drive=True` flag.

- `config_drive_format` option must be set to `iso9660` in order to use CD drive as the config drive image.
- To use config drive with Hyper-V, you must set the `mkisofs_cmd` value to the full path to an `mkisofs.exe` installation. Additionally, you must set the `qemu_img_cmd` value to the full path to an `qemu-img` command installation.
- You can configure the Compute service to always create a configuration drive by setting the `force_config_drive` option to `True`.

`config_drive_inject_password`

Type boolean

Default `false`

Inject password to config drive.

When enabled, the admin password will be available from the config drive image.

Related options:

- This option is meaningful when used with other options that enable config drive usage with Hyper-V, such as `force_config_drive`.

`volume_attach_retry_count`

Type integer

Default `10`

Minimum Value 0

Volume attach retry count

The number of times to retry attaching a volume. Volume attachment is retried until success or the given retry count is reached.

Possible values:

- Positive integer values (Default: 10).

Related options:

- Time interval between attachment attempts is declared with volume_attach_retry_interval option.

`volume_attach_retry_interval`

Type integer

Default `5`

Minimum Value 0

Volume attach retry interval

Interval between volume attachment attempts, in seconds.

Possible values:

- Time in seconds (Default: 5).

Related options:

- This options is meaningful when volume_attach_retry_count is greater than 1.
- The retry loop runs with volume_attach_retry_count and volume_attach_retry_interval configuration options.

enable_remotefx

Type boolean

Default `false`

Enable RemoteFX feature

This requires at least one DirectX 11 capable graphics adapter for Windows / Hyper-V Server 2012 R2 or newer and RDS-Virtualization feature has to be enabled.

Instances with RemoteFX can be requested with the following flavor extra specs:

os:resolution. Guest VM screen resolution size. Acceptable values:

```
1024x768, 1280x1024, 1600x1200, 1920x1200, 2560x1600, 3840x2160
```

`3840x2160` is only available on Windows / Hyper-V Server 2016.

os:monitors. Guest VM number of monitors. Acceptable values:

```
[1, 4] - Windows / Hyper-V Server 2012 R2
[1, 8] - Windows / Hyper-V Server 2016
```

os:vram. Guest VM VRAM amount. Only available on Windows / Hyper-V Server 2016. Acceptable values:

```
64, 128, 256, 512, 1024
```

use_multipath_io

Type boolean

Default `false`

Use multipath connections when attaching iSCSI or FC disks.

This requires the Multipath IO Windows feature to be enabled. MPIO must be configured to claim such devices.

iscsi_initiator_list

Type list

Default `''`

List of iSCSI initiators that will be used for estabilishing iSCSI sessions.

If none are specified, the Microsoft iSCSI initiator service will choose the initiator.

ironic

Configuration options for Ironic driver (Bare Metal). If using the Ironic driver following options must be set: * auth_type * auth_url * project_name * username * password * project_domain_id or project_domain_name * user_domain_id or user_domain_name

api_max_retries

Type integer

Default `60`

Minimum Value 0

The number of times to retry when a request conflicts. If set to 0, only try once, no retries.

Related options:

- api_retry_interval

api_retry_interval

Type integer

Default 2

Minimum Value 0

The number of seconds to wait before retrying the request.

Related options:

- api_max_retries

serial_console_state_timeout

Type integer

Default 10

Minimum Value 0

Timeout (seconds) to wait for node serial console state changed. Set to 0 to disable timeout.

partition_key

Type string

Default <None>

Mutable This option can be changed without restarting.

Case-insensitive key to limit the set of nodes that may be managed by this service to the set of nodes in Ironic which have a matching conductor_group property. If unset, all available nodes will be eligible to be managed by this service. Note that setting this to the empty string (`""`) will match the default conductor group, and is different than leaving the option unset.

peer_list

Type list

Default ''

Mutable This option can be changed without restarting.

List of hostnames for all nova-compute services (including this host) with this partition_key config value. Nodes matching the partition_key value will be distributed between all services specified here. If partition_key is unset, this option is ignored.

cafile

Type string

Default <None>

PEM encoded Certificate Authority to use when verifying HTTPs connections.

certfile

Type string

Default <None>

PEM encoded client certificate cert file

keyfile

Type string

Default `<None>`

PEM encoded client certificate key file

`insecure`

Type boolean

Default `false`

Verify HTTPS connections.

`timeout`

Type integer

Default `<None>`

Timeout value for http requests

`collect_timing`

Type boolean

Default `false`

Collect per-API call timing information.

`split_loggers`

Type boolean

Default `false`

Log requests to multiple loggers.

`auth_type`

Type unknown type

Default `<None>`

Authentication type to load

Table 66: Deprecated Variations

Group	Name
ironic	auth_plugin

`auth_section`

Type unknown type

Default `<None>`

Config Section from which to load plugin specific options

`auth_url`

Type unknown type

Default `<None>`

Authentication URL

`system_scope`

Type unknown type

Default `<None>`

Scope for system operations

domain_id

Type unknown type

Default <None>

Domain ID to scope to

domain_name

Type unknown type

Default <None>

Domain name to scope to

project_id

Type unknown type

Default <None>

Project ID to scope to

project_name

Type unknown type

Default <None>

Project name to scope to

project_domain_id

Type unknown type

Default <None>

Domain ID containing project

project_domain_name

Type unknown type

Default <None>

Domain name containing project

trust_id

Type unknown type

Default <None>

Trust ID

user_id

Type unknown type

Default <None>

User ID

username

Type unknown type

Default <None>

Username

Table 67: Deprecated Variations

Group	Name
ironic	user-name
ironic	user_name

user_domain_id

Type unknown type

Default `<None>`

Users domain id

user_domain_name

Type unknown type

Default `<None>`

Users domain name

password

Type unknown type

Default `<None>`

Users password

service_type

Type string

Default `baremetal`

The default service_type for endpoint URL discovery.

service_name

Type string

Default `<None>`

The default service_name for endpoint URL discovery.

valid_interfaces

Type list

Default `internal,public`

List of interfaces, in order of preference, for endpoint URL.

region_name

Type string

Default `<None>`

The default region_name for endpoint URL discovery.

endpoint_override

Type string

Default `<None>`

Always use this endpoint URL for requests for this client. NOTE: The unversioned endpoint should be specified here; to request a particular API version, use the *version*, *min-version*, and/or *max-version* options.

connect_retries

Type integer

Default `<None>`

The maximum number of retries that should be attempted for connection errors.

connect_retry_delay

Type floating point

Default `<None>`

Delay (in seconds) between two retries for connection errors. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

status_code_retries

Type integer

Default `<None>`

The maximum number of retries that should be attempted for retriable HTTP status codes.

status_code_retry_delay

Type floating point

Default `<None>`

Delay (in seconds) between two retries for retriable status codes. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

key_manager

fixed_key

Type string

Default `<None>`

Fixed key returned by key manager, specified in hex.

Possible values:

- Empty string or a key in hex value

Table 68: Deprecated Variations

Group	Name
keymgr	fixed_key

backend

Type string

Default `barbican`

Specify the key manager implementation. Options are barbican and vault. Default is barbican. Will support the values earlier set using [key_manager]/api_class for some time.

Table 69: Deprecated Variations

Group	Name
key_manager	api_class

auth_type

Type string

Default <None>

The type of authentication credential to create. Possible values are token, password, keystone_token, and keystone_password. Required if no context is passed to the credential factory.

token

Type string

Default <None>

Token for authentication. Required for token and keystone_token auth_type if no context is passed to the credential factory.

username

Type string

Default <None>

Username for authentication. Required for password auth_type. Optional for the keystone_password auth_type.

password

Type string

Default <None>

Password for authentication. Required for password and keystone_password auth_type.

auth_url

Type string

Default <None>

Use this endpoint to connect to Keystone.

user_id

Type string

Default <None>

User ID for authentication. Optional for keystone_token and keystone_password auth_type.

user_domain_id

Type string

Default <None>

Users domain ID for authentication. Optional for keystone_token and keystone_password auth_type.

user_domain_name

Type string

Default <None>

Users domain name for authentication. Optional for keystone_token and keystone_password auth_type.

trust_id

Type string

Default <None>

Trust ID for trust scoping. Optional for keystone_token and keystone_password auth_type.

domain_id

Type string

Default <None>

Domain ID for domain scoping. Optional for keystone_token and keystone_password auth_type.

domain_name

Type string

Default <None>

Domain name for domain scoping. Optional for keystone_token and keystone_password auth_type.

project_id

Type string

Default <None>

Project ID for project scoping. Optional for keystone_token and keystone_password auth_type.

project_name

Type string

Default <None>

Project name for project scoping. Optional for keystone_token and keystone_password auth_type.

project_domain_id

Type string

Default <None>

Projects domain ID for project. Optional for keystone_token and keystone_password auth_type.

project_domain_name

Type string

Default <None>

Projects domain name for project. Optional for keystone_token and keystone_password auth_type.

reauthenticate

Type boolean

Default true

Allow fetching a new token if the current one is going to expire. Optional for keystone_token and keystone_password auth_type.

keystone

Configuration options for the identity service

cafile

Type string

Default <None>

PEM encoded Certificate Authority to use when verifying HTTPs connections.

certfile

Type string

Default <None>

PEM encoded client certificate cert file

keyfile

Type string

Default <None>

PEM encoded client certificate key file

insecure

Type boolean

Default false

Verify HTTPS connections.

timeout

Type integer

Default <None>

Timeout value for http requests

collect_timing

Type boolean

Default false

Collect per-API call timing information.

split_loggers

Type boolean

Default false

Log requests to multiple loggers.

service_type

Type string

Default identity

The default service_type for endpoint URL discovery.

service_name

Type string

Default `<None>`

The default service_name for endpoint URL discovery.

valid_interfaces

Type list

Default `internal,public`

List of interfaces, in order of preference, for endpoint URL.

region_name

Type string

Default `<None>`

The default region_name for endpoint URL discovery.

endpoint_override

Type string

Default `<None>`

Always use this endpoint URL for requests for this client. NOTE: The unversioned endpoint should be specified here; to request a particular API version, use the *version*, *min-version*, and/or *max-version* options.

connect_retries

Type integer

Default `<None>`

The maximum number of retries that should be attempted for connection errors.

connect_retry_delay

Type floating point

Default `<None>`

Delay (in seconds) between two retries for connection errors. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

status_code_retries

Type integer

Default `<None>`

The maximum number of retries that should be attempted for retriable HTTP status codes.

status_code_retry_delay

Type floating point

Default `<None>`

Delay (in seconds) between two retries for retriable status codes. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

keystone_authtoken

www_authenticate_uri

Type string

Default `<None>`

Complete public Identity API endpoint. This endpoint should not be an admin endpoint, as it should be accessible by all end users. Unauthenticated clients are redirected to this endpoint to authenticate. Although this endpoint should ideally be unversioned, client support in the wild varies. If youre using a versioned v2 endpoint here, then this should *not* be the same endpoint the service user utilizes for validating tokens, because normal end users may not be able to reach that endpoint.

Table 70: Deprecated Variations

Group	Name
keystone_authtoken	auth_uri

auth_uri

Type string

Default `<None>`

Complete public Identity API endpoint. This endpoint should not be an admin endpoint, as it should be accessible by all end users. Unauthenticated clients are redirected to this endpoint to authenticate. Although this endpoint should ideally be unversioned, client support in the wild varies. If youre using a versioned v2 endpoint here, then this should *not* be the same endpoint the service user utilizes for validating tokens, because normal end users may not be able to reach that endpoint. This option is deprecated in favor of www_authenticate_uri and will be removed in the S release.

Warning: This option is deprecated for removal since Queens. Its value may be silently ignored in the future.

Reason The auth_uri option is deprecated in favor of www_authenticate_uri and will be removed in the S release.

auth_version

Type string

Default `<None>`

API version of the Identity API endpoint.

interface

Type string

Default `admin`

Interface to use for the Identity API endpoint. Valid values are public, internal or admin(default).

delay_auth_decision

Type boolean

Default `false`

Do not handle authorization requests within the middleware, but delegate the authorization decision to downstream WSGI components.

http_connect_timeout

Type integer

Default `<None>`

Request timeout value for communicating with Identity API server.

http_request_max_retries

Type integer

Default `3`

How many times are we trying to reconnect when communicating with Identity API Server.

cache

Type string

Default `<None>`

Request environment key where the Swift cache object is stored. When auth_token middleware is deployed with a Swift cache, use this option to have the middleware share a caching backend with swift. Otherwise, use the `memcached_servers` option instead.

certfile

Type string

Default `<None>`

Required if identity server requires client certificate

keyfile

Type string

Default `<None>`

Required if identity server requires client certificate

cafile

Type string

Default `<None>`

A PEM encoded Certificate Authority to use when verifying HTTPs connections. Defaults to system CAs.

insecure

Type boolean

Default `false`

Verify HTTPS connections.

region_name

Type string

Default `<None>`

The region in which the identity server can be found.

memcached_servers

Type list

Default `<None>`

Optionally specify a list of memcached server(s) to use for caching. If left undefined, tokens will instead be cached in-process.

Table 71: Deprecated Variations

Group	Name
keystone_authtoken	memcache_servers

token_cache_time

Type integer

Default `300`

In order to prevent excessive effort spent validating tokens, the middleware caches previously-seen tokens for a configurable duration (in seconds). Set to -1 to disable caching completely.

memcache_security_strategy

Type string

Default `None`

Valid Values None, MAC, ENCRYPT

(Optional) If defined, indicate whether token data should be authenticated or authenticated and encrypted. If MAC, token data is authenticated (with HMAC) in the cache. If ENCRYPT, token data is encrypted and authenticated in the cache. If the value is not one of these options or empty, auth_token will raise an exception on initialization.

memcache_secret_key

Type string

Default `<None>`

(Optional, mandatory if memcache_security_strategy is defined) This string is used for key derivation.

memcache_pool_dead_retry

Type integer

Default `300`

(Optional) Number of seconds memcached server is considered dead before it is tried again.

memcache_pool_maxsize

Type integer

Default `10`

(Optional) Maximum total number of open connections to every memcached server.

memcache_pool_socket_timeout

Type integer

Default `3`

(Optional) Socket timeout in seconds for communicating with a memcached server.

memcache_pool_unused_timeout

Type integer

Default `60`

(Optional) Number of seconds a connection to memcached is held unused in the pool before it is closed.

memcache_pool_conn_get_timeout

Type integer

Default `10`

(Optional) Number of seconds that an operation will wait to get a memcached client connection from the pool.

memcache_use_advanced_pool

Type boolean

Default `false`

(Optional) Use the advanced (eventlet safe) memcached client pool. The advanced pool will only work under python 2.x.

include_service_catalog

Type boolean

Default `true`

(Optional) Indicate whether to set the X-Service-Catalog header. If False, middleware will not ask for service catalog on token validation and will not set the X-Service-Catalog header.

enforce_token_bind

Type string

Default `permissive`

Used to control the use and type of token binding. Can be set to: disabled to not check token binding. permissive (default) to validate binding information if the bind type is of a form known to the server and ignore it if not. strict like permissive but if the bind type is unknown the token will be rejected. required any form of token binding is needed to be allowed. Finally the name of a binding method that must be present in tokens.

service_token_roles

Type list

Default `service`

A choice of roles that must be present in a service token. Service tokens are allowed to request that an expired token can be used and so this check should tightly control that only actual services should be sending this token. Roles here are applied as an ANY check so any role in this list must be present. For backwards compatibility reasons this currently only affects the allow_expired check.

service_token_roles_required

Type boolean

Default `false`

For backwards compatibility reasons we must let valid service tokens pass that dont pass the service_token_roles check as valid. Setting this true will become the default in a future release and should be enabled if possible.

service_type

Type string

Default `<None>`

The name or type of the service as it appears in the service catalog. This is used to validate tokens that have restricted access rules.

auth_type

Type unknown type

Default <None>

Authentication type to load

Table 72: Deprecated Variations

Group	Name
keystone_authtoken	auth_plugin

auth_section

Type unknown type

Default <None>

Config Section from which to load plugin specific options

libvirt

Libvirt options allows cloud administrator to configure related libvirt hypervisor driver to be used within an OpenStack deployment.

Almost all of the libvirt config options are influence by `virt_type` config which describes the virtualization type (or so called domain type) libvirt should use for specific features such as live migration, snapshot.

rescue_image_id

Type string

Default <None>

The ID of the image to boot from to rescue data from a corrupted instance.

If the rescue REST API operation doesnt provide an ID of an image to use, the image which is referenced by this ID is used. If this option is not set, the image from the instance is used.

Possible values:

- An ID of an image or nothing. If it points to an *Amazon Machine Image* (AMI), consider to set the config options `rescue_kernel_id` and `rescue_ramdisk_id` too. If nothing is set, the image of the instance is used.

Related options:

- `rescue_kernel_id`: If the chosen rescue image allows the separate definition of its kernel disk, the value of this option is used, if specified. This is the case when *Amazon*s AMI/AKI/ARI image format is used for the rescue image.
- `rescue_ramdisk_id`: If the chosen rescue image allows the separate definition of its RAM disk, the value of this option is used if, specified. This is the case when *Amazon*s AMI/AKI/ARI image format is used for the rescue image.

rescue_kernel_id

Type string

Default <None>

The ID of the kernel (AKI) image to use with the rescue image.

If the chosen rescue image allows the separate definition of its kernel disk, the value of this option is used, if specified. This is the case when *Amazon*s AMI/AKI/ARI image format is used for the rescue image.

Possible values:

- An ID of an kernel image or nothing. If nothing is specified, the kernel disk from the instance is used if it was launched with one.

Related options:

- `rescue_image_id`: If that option points to an image in *Amazon*s AMI/AKI/ARI image format, its useful to use `rescue_kernel_id` too.

`rescue_ramdisk_id`

Type string

Default `<None>`

The ID of the RAM disk (ARI) image to use with the rescue image.

If the chosen rescue image allows the separate definition of its RAM disk, the value of this option is used, if specified. This is the case when *Amazon*s AMI/AKI/ARI image format is used for the rescue image.

Possible values:

- An ID of a RAM disk image or nothing. If nothing is specified, the RAM disk from the instance is used if it was launched with one.

Related options:

- `rescue_image_id`: If that option points to an image in *Amazon*s AMI/AKI/ARI image format, its useful to use `rescue_ramdisk_id` too.

`virt_type`

Type string

Default `kvm`

Valid Values kvm, lxc, qemu, uml, xen, parallels

Describes the virtualization type (or so called domain type) libvirt should use.

The choice of this type must match the underlying virtualization strategy you have chosen for this host.

Related options:

- `connection_uri`: depends on this
- `disk_prefix`: depends on this
- `cpu_mode`: depends on this
- `cpu_models`: depends on this

`connection_uri`

Type string

Default `''`

Overrides the default libvirt URI of the chosen virtualization type.

If set, Nova will use this URI to connect to libvirt.

Possible values:

- An URI like `qemu:///system` or `xen+ssh://oirase/` for example. This is only necessary if the URI differs to the commonly known URIs for the chosen virtualization type.

Related options:

- `virt_type`: Influences what is used as default value here.

inject_password

Type boolean

Default `false`

Allow the injection of an admin password for instance only at `create` and `rebuild` process.

There is no agent needed within the image to do this. If *libguestfs* is available on the host, it will be used. Otherwise *nbd* is used. The file system of the image will be mounted and the admin password, which is provided in the REST API call will be injected as password for the root user. If no root user is available, the instance wont be launched and an error is thrown. Be aware that the injection is *not* possible when the instance gets launched from a volume.

Linux distribution guest only.

Possible values:

- True: Allows the injection.
- False: Disallows the injection. Any via the REST API provided admin password will be silently ignored.

Related options:

- `inject_partition`: That option will decide about the discovery and usage of the file system. It also can disable the injection at all.

inject_key

Type boolean

Default `false`

Allow the injection of an SSH key at boot time.

There is no agent needed within the image to do this. If *libguestfs* is available on the host, it will be used. Otherwise *nbd* is used. The file system of the image will be mounted and the SSH key, which is provided in the REST API call will be injected as SSH key for the root user and appended to the `authorized_keys` of that user. The SELinux context will be set if necessary. Be aware that the injection is *not* possible when the instance gets launched from a volume.

This config option will enable directly modifying the instance disk and does not affect what cloud-init may do using data from config_drive option or the metadata service.

Linux distribution guest only.

Related options:

- `inject_partition`: That option will decide about the discovery and usage of the file system. It also can disable the injection at all.

inject_partition

Type integer

Default `-2`

Minimum Value -2

Determines the way how the file system is chosen to inject data into it.

libguestfs will be used a first solution to inject data. If thats not available on the host, the image will be locally mounted on the host as a fallback solution. If libguestfs is not able to determine the root partition (because there are more or less than one root partition) or cannot mount the file system it will result in an error and the instance wont be boot.

Possible values:

- -2 => disable the injection of data.
- -1 => find the root partition with the file system to mount with libguestfs
- 0 => The image is not partitioned
- >0 => The number of the partition to use for the injection

Linux distribution guest only.

Related options:

- `inject_key`: If this option allows the injection of a SSH key it depends on value greater or equal to -1 for `inject_partition`.
- `inject_password`: If this option allows the injection of an admin password it depends on value greater or equal to -1 for `inject_partition`.
- `guestfs` You can enable the debug log level of libguestfs with this config option. A more verbose output will help in debugging issues.
- `virt_type`: If you use `lxc` as virt_type it will be treated as a single partition image

use_usb_tablet

Type boolean

Default `true`

Enable a mouse cursor within a graphical VNC or SPICE sessions.

This will only be taken into account if the VM is fully virtualized and VNC and/or SPICE is enabled. If the node doesnt support a graphical framebuffer, then it is valid to set this to False.

Related options:

- `[vnc]enabled`: If VNC is enabled, `use_usb_tablet` will have an effect.
- `[spice]enabled` + `[spice].agent_enabled`: If SPICE is enabled and the spice agent is disabled, the config value of `use_usb_tablet` will have an effect.

Warning: This option is deprecated for removal since 14.0.0. Its value may be silently ignored in the future.

Reason This option is being replaced by the pointer_model option.

live_migration_scheme

Type string

Default `<None>`

URI scheme used for live migration.

Override the default libvirt live migration scheme (which is dependent on virt_type). If this option is set to None, nova will automatically choose a sensible default based on the hypervisor. It is not recommended that you change this unless you are very sure that hypervisor supports a particular scheme.

Related options:

- `virt_type`: This option is meaningful only when `virt_type` is set to *kvm* or *qemu*.
- `live_migration_uri`: If `live_migration_uri` value is not None, the scheme used for live migration is taken from `live_migration_uri` instead.

live_migration_inbound_addr

Type host address

Default `<None>`

Target used for live migration traffic.

If this option is set to None, the hostname of the migration target compute node will be used.

This option is useful in environments where the live-migration traffic can impact the network plane significantly. A separate network for live-migration traffic can then use this config option and avoids the impact on the management network.

Related options:

- `live_migration_tunnelled`: The live_migration_inbound_addr value is ignored if tunneling is enabled.

`live_migration_uri`

Type string

Default `<None>`

Live migration target URI to use.

Override the default libvirt live migration target URI (which is dependent on virt_type). Any included %s is replaced with the migration target hostname.

If this option is set to None (which is the default), Nova will automatically generate the *live_migration_uri* value based on only 4 supported *virt_type* in following list:

- kvm: qemu+tcp://%s/system
- qemu: qemu+tcp://%s/system
- xen: xenmigr://%s/system
- parallels: parallels+tcp://%s/system

Related options:

- `live_migration_inbound_addr`: If `live_migration_inbound_addr` value is not None and `live_migration_tunnelled` is False, the ip/hostname address of target compute node is used instead of `live_migration_uri` as the uri for live migration.
- `live_migration_scheme`: If `live_migration_uri` is not set, the scheme used for live migration is taken from `live_migration_scheme` instead.

Warning: This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.

Reason live_migration_uri is deprecated for removal in favor of two other options that allow to change live migration scheme and target URI: `live_migration_scheme` and `live_migration_inbound_addr` respectively.

`live_migration_tunnelled`

Type boolean

Default `false`

Enable tunnelled migration.

This option enables the tunnelled migration feature, where migration data is transported over the libvirtd connection. If enabled, we use the VIR_MIGRATE_TUNNELLED migration flag, avoiding the need to configure

the network to allow direct hypervisor to hypervisor communication. If False, use the native transport. If not set, Nova will choose a sensible default based on, for example the availability of native encryption support in the hypervisor. Enabling this option will definitely impact performance massively.

Note that this option is NOT compatible with use of block migration.

Related options:

- `live_migration_inbound_addr`: The live_migration_inbound_addr value is ignored if tunneling is enabled.

`live_migration_bandwidth`

Type integer

Default `0`

Maximum bandwidth(in MiB/s) to be used during migration.

If set to 0, the hypervisor will choose a suitable default. Some hypervisors do not support this feature and will return an error if bandwidth is not 0. Please refer to the libvirt documentation for further details.

`live_migration_downtime`

Type integer

Default `500`

Minimum Value 100

Maximum permitted downtime, in milliseconds, for live migration switchover.

Will be rounded up to a minimum of 100ms. You can increase this value if you want to allow live-migrations to complete faster, or avoid live-migration timeout errors by allowing the guest to be paused for longer during the live-migration switch over.

Related options:

- live_migration_completion_timeout

`live_migration_downtime_steps`

Type integer

Default `10`

Minimum Value 3

Number of incremental steps to reach max downtime value.

Will be rounded up to a minimum of 3 steps.

`live_migration_downtime_delay`

Type integer

Default `75`

Minimum Value 3

Time to wait, in seconds, between each step increase of the migration downtime.

Minimum delay is 3 seconds. Value is per GiB of guest RAM + disk to be transferred, with lower bound of a minimum of 2 GiB per device.

`live_migration_completion_timeout`

Type integer

Default `800`

Minimum Value 0

Mutable This option can be changed without restarting.

Time to wait, in seconds, for migration to successfully complete transferring data before aborting the operation.

Value is per GiB of guest RAM + disk to be transferred, with lower bound of a minimum of 2 GiB. Should usually be larger than downtime delay * downtime steps. Set to 0 to disable timeouts.

Related options:

- live_migration_downtime
- live_migration_downtime_steps
- live_migration_downtime_delay

`live_migration_timeout_action`

Type string

Default `abort`

Valid Values abort, force_complete

Mutable This option can be changed without restarting.

This option will be used to determine what action will be taken against a VM after `live_migration_completion_timeout` expires. By default, the live migrate operation will be aborted after completion timeout. If it is set to `force_complete`, the compute service will either pause the VM or trigger post-copy depending on if post copy is enabled and available (`live_migration_permit_post_copy` is set to True).

Related options:

- live_migration_completion_timeout
- live_migration_permit_post_copy

`live_migration_permit_post_copy`

Type boolean

Default `false`

This option allows nova to switch an on-going live migration to post-copy mode, i.e., switch the active VM to the one on the destination node before the migration is complete, therefore ensuring an upper bound on the memory that needs to be transferred. Post-copy requires libvirt>=1.3.3 and QEMU>=2.5.0.

When permitted, post-copy mode will be automatically activated if we reach the timeout defined by `live_migration_completion_timeout` and `live_migration_timeout_action` is set to force_complete. Note if you change to no timeout or choose to use abort, i.e. `live_migration_completion_timeout = 0`, then there will be no automatic switch to post-copy.

The live-migration force complete API also uses post-copy when permitted. If post-copy mode is not available, force complete falls back to pausing the VM to ensure the live-migration operation will complete.

When using post-copy mode, if the source and destination hosts lose network connectivity, the VM being live-migrated will need to be rebooted. For more details, please see the Administration guide.

Related options:

- live_migration_permit_auto_converge
- live_migration_timeout_action

live_migration_permit_auto_converge

Type boolean

Default `false`

This option allows nova to start live migration with auto converge on.

Auto converge throttles down CPU if a progress of on-going live migration is slow. Auto converge will only be used if this flag is set to True and post copy is not permitted or post copy is unavailable due to the version of libvirt and QEMU in use.

Related options:

- live_migration_permit_post_copy

snapshot_image_format

Type string

Default `<None>`

Valid Values raw, qcow2, vmdk, vdi

Determine the snapshot image format when sending to the image service.

If set, this decides what format is used when sending the snapshot to the image service. If not set, defaults to same type as source image.

Possible values

raw RAW disk format

qcow2 KVM default disk format

vmdk VMWare default disk format

vdi VirtualBox default disk format

live_migration_with_native_tls

Type boolean

Default `false`

Use QEMU-native TLS encryption when live migrating.

This option will allow both migration stream (guest RAM plus device state) *and* disk stream to be transported over native TLS, i.e. TLS support built into QEMU.

Prerequisite: TLS environment is configured correctly on all relevant Compute nodes. This means, Certificate Authority (CA), server, client certificates, their corresponding keys, and their file permisssions are in place, and are validated.

Notes:

- To have encryption for migration stream and disk stream (also called: block migration), `live_migration_with_native_tls` is the preferred config attribute instead of `live_migration_tunnelled`.
- The `live_migration_tunnelled` will be deprecated in the long-term for two main reasons: (a) it incurs a huge performance penalty; and (b) it is not compatible with block migration. Therefore, if your compute nodes have at least libvirt 4.4.0 and QEMU 2.11.0, it is strongly recommended to use `live_migration_with_native_tls`.

- The `live_migration_tunnelled` and `live_migration_with_native_tls` should not be used at the same time.
- Unlike `live_migration_tunnelled`, the `live_migration_with_native_tls` *is* compatible with block migration. That is, with this option, NBD stream, over which disks are migrated to a target host, will be encrypted.

Related options:

`live_migration_tunnelled`: This transports migration stream (but not disk stream) over libvirtd.

`disk_prefix`

Type string

Default `<None>`

Override the default disk prefix for the devices attached to an instance.

If set, this is used to identify a free disk device name for a bus.

Possible values:

- Any prefix which will result in a valid disk device name like sda or hda for example. This is only necessary if the device names differ to the commonly known device name prefixes for a virtualization type such as: sd, xvd, uvd, vd.

Related options:

- `virt_type`: Influences which device type is used, which determines the default disk prefix.

`wait_soft_reboot_seconds`

Type integer

Default `120`

Number of seconds to wait for instance to shut down after soft reboot request is made. We fall back to hard reboot if instance does not shutdown within this window.

`cpu_mode`

Type string

Default `<None>`

Valid Values host-model, host-passthrough, custom, none

Is used to set the CPU mode an instance should have.

If `virt_type="kvm|qemu"`, it will default to `host-model`, otherwise it will default to `none`.

Related options:

- `cpu_models`: This should be set ONLY when `cpu_mode` is set to `custom`. Otherwise, it would result in an error and the instance launch will fail.

Possible values

host-model Clone the host CPU feature flags

host-passthrough Use the host CPU model exactly

custom Use the CPU model in `[libvirt]cpu_models`

none Dont set a specific CPU model. For instances with `[libvirt] virt_type` as KVM/QEMU, the default CPU model from QEMU will be used, which provides a basic set of CPU features that are compatible with most hosts

cpu_models

Type list

Default `''`

An ordered list of CPU models the host supports.

It is expected that the list is ordered so that the more common and less advanced CPU models are listed earlier. Here is an example: `SandyBridge,IvyBridge,Haswell,Broadwell`, the latter CPU models features is richer that the previous CPU model.

Possible values:

- The named CPU models listed in `/usr/share/libvirt/cpu_map.xml` for libvirt prior to version 4.7.0 or `/usr/share/libvirt/cpu_map/*.xml` for version 4.7.0 and higher.

Related options:

- `cpu_mode`: This should be set to `custom` ONLY when you want to configure (via `cpu_models`) a specific named CPU model. Otherwise, it would result in an error and the instance launch will fail.
- `virt_type`: Only the virtualization types `kvm` and `qemu` use this.

Note: Be careful to only specify models which can be fully supported in hardware.

Table 73: Deprecated Variations

Group	Name
libvirt	cpu_model

cpu_model_extra_flags

Type list

Default `''`

This allows specifying granular CPU feature flags when configuring CPU models. For example, to explicitly specify the `pcid` (Process-Context ID, an Intel processor feature – which is now required to address the guest performance degradation as a result of applying the Meltdown CVE fixes to certain Intel CPU models) flag to the IvyBridge virtual CPU model:

```
[libvirt]
cpu_mode = custom
cpu_models = IvyBridge
cpu_model_extra_flags = pcid
```

To specify multiple CPU flags (e.g. the Intel `VMX` to expose the virtualization extensions to the guest, or `pdpe1gb` to configure 1GB huge pages for CPU models that do not provide it):

```
[libvirt]
cpu_mode = custom
cpu_models = Haswell-noTSX-IBRS
cpu_model_extra_flags = PCID, VMX, pdpe1gb
```

As it can be noticed from above, the `cpu_model_extra_flags` config attribute is case insensitive. And specifying extra flags is valid in combination with all the three possible values for `cpu_mode`: `custom` (this also requires an explicit `cpu_models` to be specified), `host-model`, or `host-passthrough`. A valid example for allowing extra CPU flags even for `host-passthrough` mode is that sometimes QEMU may disable certain CPU features – e.g. Intels invtsc, Invariable Time Stamp Counter, CPU flag. And if you need to expose that CPU flag to the Nova instance, the you need to explicitly ask for it.

The possible values for `cpu_model_extra_flags` depends on the CPU model in use. Refer to `/usr/share/libvirt/cpu_map.xml` for libvirt prior to version 4.7.0 or `/usr/share/libvirt/cpu_map/*.xml` thereafter for possible CPU feature flags for a given CPU model.

Note that when using this config attribute to set the PCID CPU flag with the `custom` CPU mode, not all virtual (i.e. libvirt / QEMU) CPU models need it:

- The only virtual CPU models that include the PCID capability are Intel Haswell, Broadwell, and Skylake variants.
- The libvirt / QEMU CPU models Nehalem, Westmere, SandyBridge, and IvyBridge will _not_ expose the PCID capability by default, even if the host CPUs by the same name include it. I.e. PCID needs to be explicitly specified when using the said virtual CPU models.

The libvirt drivers default CPU mode, `host-model`, will do the right thing with respect to handling PCID CPU flag for the guest – *assuming* you are running updated processor microcode, host and guest kernel, libvirt, and QEMU. The other mode, `host-passthrough`, checks if PCID is available in the hardware, and if so directly passes it through to the Nova guests. Thus, in context of PCID, with either of these CPU modes (`host-model` or `host-passthrough`), there is no need to use the `cpu_model_extra_flags`.

Related options:

- cpu_mode
- cpu_models

`snapshots_directory`

Type string

Default `$instances_path/snapshots`

Location where libvirt driver will store snapshots before uploading them to image service

`xen_hvmloader_path`

Type string

Default `/usr/lib/xen/boot/hvmloader`

Location where the Xen hvmloader is kept

`disk_cachemodes`

Type list

Default `''`

Specific cache modes to use for different disk types.

For example: file=directsync,block=none,network=writeback

For local or direct-attached storage, it is recommended that you use writethrough (default) mode, as it ensures data integrity and has acceptable I/O performance for applications running in the guest, especially for read operations. However, caching mode none is recommended for remote NFS storage, because direct I/O operations (O_DIRECT) perform better than synchronous I/O operations (with O_SYNC). Caching mode none effectively turns all guest I/O operations into direct I/O operations on the host, which is the NFS client in this environment.

Possible cache modes:

- default: It Depends – For Nova-managed disks, `none`, if the host file system is capable of Linuxs O_DIRECT semantics; otherwise `writeback`. For volume drivers, the default is driver-dependent: `none` for everything except for SMBFS and Virtuzzo (which use `writeback`).
- none: With caching mode set to none, the host page cache is disabled, but the disk write cache is enabled for the guest. In this mode, the write performance in the guest is optimal because write operations bypass the host page cache and go directly to the disk write cache. If the disk write cache is battery-backed, or if the applications or storage stack in the guest transfer data properly (either through fsync operations or file system barriers), then data integrity can be ensured. However, because the host page cache is disabled, the read performance in the guest would not be as good as in the modes where the host page cache is enabled, such as writethrough mode. Shareable disk devices, like for a multi-attachable block storage volume, will have their cache mode set to none regardless of configuration.
- writethrough: With caching set to writethrough mode, the host page cache is enabled, but the disk write cache is disabled for the guest. Consequently, this caching mode ensures data integrity even if the applications and storage stack in the guest do not transfer data to permanent storage properly (either through fsync operations or file system barriers). Because the host page cache is enabled in this mode, the read performance for applications running in the guest is generally better. However, the write performance might be reduced because the disk write cache is disabled.
- writeback: With caching set to writeback mode, both the host page cache and the disk write cache are enabled for the guest. Because of this, the I/O performance for applications running in the guest is good, but the data is not protected in a power failure. As a result, this caching mode is recommended only for temporary data where potential data loss is not a concern. NOTE: Certain backend disk mechanisms may provide safe writeback cache semantics. Specifically those that bypass the host page cache, such as QEMUs integrated RBD driver. Ceph documentation recommends setting this to writeback for maximum performance while maintaining data safety.
- directsync: Like writethrough, but it bypasses the host page cache.
- unsafe: Caching mode of unsafe ignores cache transfer operations completely. As its name implies, this caching mode should be used only for temporary data where data loss is not a concern. This mode can be useful for speeding up guest installations, but you should switch to another caching mode in production environments.

`rng_dev_path`

Type string

Default `/dev/urandom`

The path to an RNG (Random Number Generator) device that will be used as the source of entropy on the host. Since libvirt 1.3.4, any path (that returns random numbers when read) is accepted. The recommended source of entropy is `/dev/urandom` – it is non-blocking, therefore relatively fast; and avoids the limitations of `/dev/random`, which is a legacy interface. For more details (and comparision between different RNG sources), refer to the Usage section in the Linux kernel API documentation for `[u]random`: http://man7.org/linux/man-pages/man4/urandom.4.html and http://man7.org/linux/man-pages/man7/random.7.html.

`hw_machine_type`

Type list

Default `<None>`

For qemu or KVM guests, set this option to specify a default machine type per host architecture. You can find a list of supported machine types in your environment by checking the output of the **`virsh capabilities`** command. The format of the value for this config option is `host-arch=machine-type`. For example: `x86_64=machinetype1,armv7l=machinetype2`.

sysinfo_serial

Type string

Default `unique`

Valid Values none, os, hardware, auto, unique

The data source used to the populate the host serial UUID exposed to guest in the virtual BIOS. All choices except `unique` will change the serial when migrating the instance to another host. Changing the choice of this option will also affect existing instances on this host once they are stopped and started again. It is recommended to use the default choice (`unique`) since that will not change when an instance is migrated. However, if you have a need for per-host serials in addition to per-instance serial numbers, then consider restricting flavors via host aggregates.

Possible values

none A serial number entry is not added to the guest domain xml.

os A UUID serial number is generated from the host `/etc/machine-id` file.

hardware A UUID for the host hardware as reported by libvirt. This is typically from the host SMBIOS data, unless it has been overridden in `libvirtd.conf`.

auto Uses the os source if possible, else hardware.

unique Uses instance UUID as the serial number.

mem_stats_period_seconds

Type integer

Default `10`

A number of seconds to memory usage statistics period. Zero or negative value mean to disable memory usage statistics.

uid_maps

Type list

Default `''`

List of uid targets and ranges.Syntax is guest-uid:host-uid:count. Maximum of 5 allowed.

gid_maps

Type list

Default `''`

List of guid targets and ranges.Syntax is guest-gid:host-gid:count. Maximum of 5 allowed.

realtime_scheduler_priority

Type integer

Default `1`

In a realtime host context vCPUs for guest will run in that scheduling priority. Priority depends on the host kernel (usually 1-99)

enabled_perf_events

Type list

Default `''`

This will allow you to specify a list of events to monitor low-level performance of guests, and collect related statsitics via the libvirt driver, which in turn uses the Linux kernels *perf* infrastructure. With this config attribute set, Nova will generate libvirt guest XML to monitor the specified events. For more information, refer to the Performance monitoring events section here: https://libvirt.org/formatdomain.html#elementsPerf. And here: https://libvirt.org/html/libvirt-libvirt-domain.html – look for `VIR_PERF_PARAM_*`

For example, to monitor the count of CPU cycles (total/elapsed) and the count of cache misses, enable them as follows:

```
[libvirt]
enabled_perf_events = cpu_clock, cache_misses
```

Possible values: A string list. The list of supported events can be found here: https://libvirt.org/formatdomain.html#elementsPerf.

Note that support for Intel CMT events (*cmt*, *mbmbt*, *mbml*) is deprecated, and will be removed in the Stein release. Thats because the upstream Linux kernel (from 4.14 onwards) has deleted support for Intel CMT, because it is broken by design.

`num_pcie_ports`

Type integer

Default `0`

Minimum Value 0

Maximum Value 28

The number of PCIe ports an instance will get.

Libvirt allows a custom number of PCIe ports (pcie-root-port controllers) a target instance will get. Some will be used by default, rest will be available for hotplug use.

By default we have just 1-2 free ports which limits hotplug.

More info: https://github.com/qemu/qemu/blob/master/docs/pcie.txt

Due to QEMU limitations for aarch64/virt maximum value is set to 28.

Default value 0 moves calculating amount of ports to libvirt.

`file_backed_memory`

Type integer

Default `0`

Minimum Value 0

Available capacity in MiB for file-backed memory.

Set to 0 to disable file-backed memory.

When enabled, instances will create memory files in the directory specified in `/etc/libvirt/qemu.confs memory_backing_dir` option. The default location is `/var/lib/libvirt/qemu/ram`.

When enabled, the value defined for this option is reported as the node memory capacity. Compute node system memory will be used as a cache for file-backed memory, via the kernels pagecache mechanism.

Note: This feature is not compatible with hugepages.

Note: This feature is not compatible with memory overcommit.

Related options:

- `virt_type` must be set to `kvm` or `qemu`.
- `ram_allocation_ratio` must be set to 1.0.

`num_memory_encrypted_guests`

Type integer

Default `<None>`

Minimum Value 0

Maximum number of guests with encrypted memory which can run concurrently on this compute host.

For now this is only relevant for AMD machines which support SEV (Secure Encrypted Virtualization). Such machines have a limited number of slots in their memory controller for storing encryption keys. Each running guest with encrypted memory will consume one of these slots.

The option may be reused for other equivalent technologies in the future. If the machine does not support memory encryption, the option will be ignored and inventory will be set to 0.

If the machine does support memory encryption, *for now* a value of `None` means an effectively unlimited inventory, i.e. no limit will be imposed by Nova on the number of SEV guests which can be launched, even though the underlying hardware will enforce its own limit. However it is expected that in the future, auto-detection of the inventory from the hardware will become possible, at which point `None` will cause auto-detection to automatically impose the correct limit.

Note: It is recommended to read *the deployment documentations section on this option* before deciding whether to configure this setting or leave it at the default.

Related options:

- *`libvirt.virt_type`* must be set to `kvm`.
- Its recommended to consider including `x86_64=q35` in *`libvirt.hw_machine_type`*; see *Deploying SEV-capable infrastructure* for more on this.

`images_type`

Type string

Default `default`

Valid Values raw, flat, qcow2, lvm, rbd, ploop, default

VM Images format.

If default is specified, then use_cow_images flag is used instead of this one.

Related options:

- compute.use_cow_images
- images_volume_group
- [workarounds]/ensure_libvirt_rbd_instance_dir_cleanup
- compute.force_raw_images

images_volume_group

Type string

Default <None>

LVM Volume Group that is used for VM images, when you specify images_type=lvm

Related options:

- images_type

sparse_logical_volumes

Type boolean

Default false

Create sparse logical volumes (with virtualsize) if this flag is set to True.

Warning: This option is deprecated for removal since 18.0.0. Its value may be silently ignored in the future.

Reason Sparse logical volumes is a feature that is not tested hence not supported. LVM logical volumes are preallocated by default. If you want thin provisioning, use Cinder thin-provisioned volumes.

images_rbd_pool

Type string

Default rbd

The RADOS pool in which rbd volumes are stored

images_rbd_ceph_conf

Type string

Default ''

Path to the ceph configuration file to use

hw_disk_discard

Type string

Default <None>

Valid Values ignore, unmap

Discard option for nova managed disks.

Requires:

- Libvirt >= 1.0.6
- Qemu >= 1.5 (raw format)
- Qemu >= 1.6 (qcow2 format)

remove_unused_resized_minimum_age_seconds

Type integer

Default 3600

Unused resized base images younger than this will not be removed

volume_clear

Type string

Default `zero`

Valid Values zero, shred, none

Method used to wipe ephemeral disks when they are deleted. Only takes effect if LVM is set as backing storage.

Related options:

- images_type - must be set to `lvm`
- volume_clear_size

Possible values

zero Overwrite volumes with zeroes

shred Overwrite volumes repeatedly

none Do not wipe deleted volumes

volume_clear_size

Type integer

Default `0`

Minimum Value 0

Size of area in MiB, counting from the beginning of the allocated volume, that will be cleared using method set in `volume_clear` option.

Possible values:

- 0 - clear whole volume
- >0 - clear specified amount of MiB

Related options:

- images_type - must be set to `lvm`
- volume_clear - must be set and the value must be different than `none` for this option to have any impact

snapshot_compression

Type boolean

Default `false`

Enable snapshot compression for `qcow2` images.

Note: you can set `snapshot_image_format` to `qcow2` to force all snapshots to be in `qcow2` format, independently from their original image type.

Related options:

- snapshot_image_format

use_virtio_for_bridges

Type boolean

Default `true`

Use virtio for bridge interfaces with KVM/QEMU

volume_use_multipath

Type boolean

Default `false`

Use multipath connection of the iSCSI or FC volume

Volumes can be connected in the LibVirt as multipath devices. This will provide high availability and fault tolerance.

Table 74: Deprecated Variations

Group	Name
libvirt	iscsi_use_multipath

num_volume_scan_tries

Type integer

Default `5`

Number of times to scan given storage protocol to find volume.

Table 75: Deprecated Variations

Group	Name
libvirt	num_iscsi_scan_tries

num_aoe_discover_tries

Type integer

Default `3`

Number of times to rediscover AoE target to find volume.

Nova provides support for block storage attaching to hosts via AOE (ATA over Ethernet). This option allows the user to specify the maximum number of retry attempts that can be made to discover the AoE device.

iscsi_iface

Type string

Default `<None>`

The iSCSI transport iface to use to connect to target in case offload support is desired.

Default format is of the form <transport_name>.<hwaddress> where <transport_name> is one of (be2iscsi, bnx2i, cxgb3i, cxgb4i, qla4xxx, ocs) and <hwaddress> is the MAC address of the interface and can be generated via the iscsiadm -m iface command. Do not confuse the iscsi_iface parameter to be provided here with the actual transport name.

Table 76: Deprecated Variations

Group	Name
libvirt	iscsi_transport

num_iser_scan_tries

Type integer

Default `5`

Number of times to scan iSER target to find volume.

iSER is a server network protocol that extends iSCSI protocol to use Remote Direct Memory Access (RDMA). This option allows the user to specify the maximum number of scan attempts that can be made to find iSER volume.

`iser_use_multipath`

Type boolean

Default `false`

Use multipath connection of the iSER volume.

iSER volumes can be connected as multipath devices. This will provide high availability and fault tolerance.

`rbd_user`

Type string

Default `<None>`

The RADOS client name for accessing rbd(RADOS Block Devices) volumes.

Libvirt will refer to this user when connecting and authenticating with the Ceph RBD server.

`rbd_secret_uuid`

Type string

Default `<None>`

The libvirt UUID of the secret for the rbd_user volumes.

`rbd_connect_timeout`

Type integer

Default `5`

The RADOS client timeout in seconds when initially connecting to the cluster.

`nfs_mount_point_base`

Type string

Default `$state_path/mnt`

Directory where the NFS volume is mounted on the compute node. The default is mnt directory of the location where novas Python module is installed.

NFS provides shared storage for the OpenStack Block Storage service.

Possible values:

- A string representing absolute path of mount point.

`nfs_mount_options`

Type string

Default `<None>`

Mount options passed to the NFS client. See section of the nfs man page for details.

Mount options controls the way the filesystem is mounted and how the NFS client behaves when accessing files on this mount point.

Possible values:

- Any string representing mount options separated by commas.
- Example string: vers=3,lookupcache=pos

quobyte_mount_point_base

Type string

Default `$state_path/mnt`

Directory where the Quobyte volume is mounted on the compute node.

Nova supports Quobyte volume driver that enables storing Block Storage service volumes on a Quobyte storage back end. This Option specifies the path of the directory where Quobyte volume is mounted.

Possible values:

- A string representing absolute path of mount point.

quobyte_client_cfg

Type string

Default `<None>`

Path to a Quobyte Client configuration file.

smbfs_mount_point_base

Type string

Default `$state_path/mnt`

Directory where the SMBFS shares are mounted on the compute node.

smbfs_mount_options

Type string

Default `''`

Mount options passed to the SMBFS client.

Provide SMBFS options as a single string containing all parameters. See mount.cifs man page for details. Note that the libvirt-qemu `uid` and `gid` must be specified.

remote_filesystem_transport

Type string

Default `ssh`

Valid Values ssh, rsync

libvirts transport method for remote file operations.

Because libvirt cannot use RPC to copy files over network to/from other compute nodes, other method must be used for:

- creating directory on remote host
- creating file on remote host
- removing file from remote host
- copying file to remote host

vzstorage_mount_point_base

Type string

Default `$state_path/mnt`

Directory where the Virtuozzo Storage clusters are mounted on the compute node.

This option defines non-standard mountpoint for Vzstorage cluster.

Related options:

- vzstorage_mount_* group of parameters

vzstorage_mount_user

Type string

Default `stack`

Mount owner user name.

This option defines the owner user of Vzstorage cluster mountpoint.

Related options:

- vzstorage_mount_* group of parameters

vzstorage_mount_group

Type string

Default `qemu`

Mount owner group name.

This option defines the owner group of Vzstorage cluster mountpoint.

Related options:

- vzstorage_mount_* group of parameters

vzstorage_mount_perms

Type string

Default `0770`

Mount access mode.

This option defines the access bits of Vzstorage cluster mountpoint, in the format similar to one of chmod(1) utility, like this: 0770. It consists of one to four digits ranging from 0 to 7, with missing lead digits assumed to be 0s.

Related options:

- vzstorage_mount_* group of parameters

vzstorage_log_path

Type string

Default `/var/log/vstorage/%(cluster_name)s/nova.log.gz`

Path to vzstorage client log.

This option defines the log of cluster operations, it should include %(cluster_name)s template to separate logs from multiple shares.

Related options:

- vzstorage_mount_opts may include more detailed logging options.

vzstorage_cache_path

Type string

Default <None>

Path to the SSD cache file.

You can attach an SSD drive to a client and configure the drive to store a local cache of frequently accessed data. By having a local cache on a clients SSD drive, you can increase the overall cluster performance by up to 10 and more times. WARNING! There is a lot of SSD models which are not server grade and may loose arbitrary set of data changes on power loss. Such SSDs should not be used in Vstorage and are dangerous as may lead to data corruptions and inconsistencies. Please consult with the manual on which SSD models are known to be safe or verify it using vstorage-hwflush-check(1) utility.

This option defines the path which should include %(cluster_name)s template to separate caches from multiple shares.

Related options:

- vzstorage_mount_opts may include more detailed cache options.

vzstorage_mount_opts

Type list

Default ''

Extra mount options for pstorage-mount

For full description of them, see https://static.openvz.org/vz-man/man1/pstorage-mount.1.gz.html Format is a python string representation of arguments list, like: [-v, -R, 500] Shouldnt include -c, -l, -C, -u, -g and -m as those have explicit vzstorage_* options.

Related options:

- All other vzstorage_* options

rx_queue_size

Type unknown type

Default <None>

Valid Values 256, 512, 1024

Configure virtio rx queue size.

This option is only usable for virtio-net device with vhost and vhost-user backend. Available only with QEMU/KVM. Requires libvirt v2.3 QEMU v2.7.

tx_queue_size

Type unknown type

Default <None>

Valid Values 256, 512, 1024

Configure virtio tx queue size.

This option is only usable for virtio-net device with vhost-user backend. Available only with QEMU/KVM. Requires libvirt v3.7 QEMU v2.10.

max_queues

Type integer

Default `<None>`

Minimum Value 1

The maximum number of virtio queue pairs that can be enabled when creating a multiqueue guest. The number of virtio queues allocated will be the lesser of the CPUs requested by the guest and the max value defined. By default, this value is set to none meaning the legacy limits based on the reported kernel major version will be used.

num_nvme_discover_tries

Type integer

Default `5`

Number of times to rediscover NVMe target to find volume

Nova provides support for block storage attaching to hosts via NVMe (Non-Volatile Memory Express). This option allows the user to specify the maximum number of retry attempts that can be made to discover the NVMe device.

pmem_namespaces

Type list

Default `''`

Configure persistent memory(pmem) namespaces.

These namespaces must have been already created on the host. This config option is in the following format:

```
"$LABEL:$NSNAME[|$NSNAME][,$LABEL:$NSNAME[|$NSNAME]]"
```

- `$NSNAME` is the name of the pmem namespace.
- **`$LABEL` represents one resource class, this is used to generate** the resource class name as `CUSTOM_PMEM_NAMESPACE_$LABEL`.

For example:

```
[libvirt]
pmem_namespaces=128G:ns0|ns1|ns2|ns3,262144MB:ns4|ns5,MEDIUM:ns6|ns7
```

metrics

Configuration options for metrics

Options under this group allow to adjust how values assigned to metrics are calculated.

weight_multiplier

Type floating point

Default `1.0`

When using metrics to weight the suitability of a host, you can use this option to change how the calculated weight influences the weight assigned to a host as follows:

- >1.0: increases the effect of the metric on overall weight
- 1.0: no change to the calculated weight
- >0.0,<1.0: reduces the effect of the metric on overall weight

- 0.0: the metric value is ignored, and the value of the weight_of_unavailable option is returned instead
- >-1.0,<0.0: the effect is reduced and reversed
- -1.0: the effect is reversed
- <-1.0: the effect is increased proportionally and reversed

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect.

Possible values:

- An integer or float value, where the value corresponds to the multipler ratio for this weigher.

Related options:

- weight_of_unavailable

weight_setting

Type list

Default ''

This setting specifies the metrics to be weighed and the relative ratios for each metric. This should be a single string value, consisting of a series of one or more name=ratio pairs, separated by commas, where name is the name of the metric to be weighed, and ratio is the relative weight for that metric.

Note that if the ratio is set to 0, the metric value is ignored, and instead the weight will be set to the value of the weight_of_unavailable option.

As an example, lets consider the case where this option is set to:

```
name1=1.0, name2=-1.3
```

The final weight will be:

```
(name1.value * 1.0) + (name2.value * -1.3)
```

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect.

Possible values:

- A list of zero or more key/value pairs separated by commas, where the key is a string representing the name of a metric and the value is a numeric weight for that metric. If any value is set to 0, the value is ignored and the weight will be set to the value of the weight_of_unavailable option.

Related options:

- weight_of_unavailable

required

Type boolean

Default `true`

This setting determines how any unavailable metrics are treated. If this option is set to True, any hosts for which a metric is unavailable will raise an exception, so it is recommended to also use the MetricFilter to filter out those hosts before weighing.

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect.

Possible values:

- True or False, where False ensures any metric being unavailable for a host will set the host weight to weight_of_unavailable.

Related options:

- weight_of_unavailable

weight_of_unavailable

Type floating point

Default `-10000.0`

When any of the following conditions are met, this value will be used in place of any actual metric value:

- One of the metrics named in weight_setting is not available for a host, and the value of required is False
- The ratio specified for a metric in weight_setting is 0
- The weight_multiplier option is set to 0

This option is only used by the FilterScheduler and its subclasses; if you use a different scheduler, this option has no effect.

Possible values:

- An integer or float value, where the value corresponds to the multipler ratio for this weigher.

Related options:

- weight_setting
- required
- weight_multiplier

mks

Nova compute node uses WebMKS, a desktop sharing protocol to provide instance console access to VMs created by VMware hypervisors.

Related options: Following options must be set to provide console access. * mksproxy_base_url * enabled

mksproxy_base_url

Type URI

Default `http://127.0.0.1:6090/`

Location of MKS web console proxy

The URL in the response points to a WebMKS proxy which starts proxying between client and corresponding vCenter server where instance runs. In order to use the web based console access, WebMKS proxy should be installed and configured

Possible values:

- Must be a valid URL of the form:`http://host:port/` or `https://host:port/`

enabled

Type boolean

Default `false`

Enables graphical console access for virtual machines.

neutron

Configuration options for neutron (network connectivity as a service).

ovs_bridge

Type string

Default `br-int`

Default name for the Open vSwitch integration bridge.

Specifies the name of an integration bridge interface used by OpenvSwitch. This option is only used if Neutron does not specify the OVS bridge name in port binding responses.

default_floating_pool

Type string

Default `nova`

Default name for the floating IP pool.

Specifies the name of floating IP pool used for allocating floating IPs. This option is only used if Neutron does not specify the floating IP pool name in port binding reponses.

extension_sync_interval

Type integer

Default `600`

Minimum Value 0

Integer value representing the number of seconds to wait before querying Neutron for extensions. After this number of seconds the next time Nova needs to create a resource in Neutron it will requery Neutron for the extensions that it has loaded. Setting value to 0 will refresh the extensions with no wait.

physnets

Type list

Default `''`

List of physnets present on this host.

For each *physnet* listed, an additional section, `[neutron_physnet_$PHYSNET]`, will be added to the configuration file. Each section must be configured with a single configuration option, `numa_nodes`, which should be a list of node IDs for all NUMA nodes this physnet is associated with. For example:

```
[neutron]
physnets = foo, bar

[neutron_physnet_foo]
numa_nodes = 0

[neutron_physnet_bar]
numa_nodes = 0,1
```

Any *physnet* that is not listed using this option will be treated as having no particular NUMA node affinity.

Tunnelled networks (VXLAN, GRE,) cannot be accounted for in this way and are instead configured using the `[neutron_tunnel]` group. For example:

```
[neutron_tunnel]
numa_nodes = 1
```

Related options:

- `[neutron_tunnel] numa_nodes` can be used to configure NUMA affinity for all tunneled networks
- `[neutron_physnet_$PHYSNET] numa_nodes` must be configured for each value of `$PHYSNET` specified by this option

`http_retries`

Type integer

Default `3`

Minimum Value 0

Number of times neutronclient should retry on any failed http call.

0 means connection is attempted only once. Setting it to any positive integer means that on failure connection is retried that many times e.g. setting it to 3 means total attempts to connect will be 4.

Possible values:

- Any integer value. 0 means connection is attempted only once

`service_metadata_proxy`

Type boolean

Default `false`

When set to True, this option indicates that Neutron will be used to proxy metadata requests and resolve instance ids. Otherwise, the instance ID must be passed to the metadata request in the X-Instance-ID header.

Related options:

- metadata_proxy_shared_secret

`metadata_proxy_shared_secret`

Type string

Default `''`

This option holds the shared secret string used to validate proxy requests to Neutron metadata requests. In order to be used, the X-Metadata-Provider-Signature header must be supplied in the request.

Related options:

- service_metadata_proxy

`cafile`

Type string

Default `<None>`

PEM encoded Certificate Authority to use when verifying HTTPs connections.

`certfile`

Type string

Default `<None>`

PEM encoded client certificate cert file

keyfile

Type string

Default `<None>`

PEM encoded client certificate key file

insecure

Type boolean

Default `false`

Verify HTTPS connections.

timeout

Type integer

Default `<None>`

Timeout value for http requests

collect_timing

Type boolean

Default `false`

Collect per-API call timing information.

split_loggers

Type boolean

Default `false`

Log requests to multiple loggers.

auth_type

Type unknown type

Default `<None>`

Authentication type to load

Table 77: Deprecated Variations

Group	Name
neutron	auth_plugin

auth_section

Type unknown type

Default `<None>`

Config Section from which to load plugin specific options

auth_url

Type unknown type

Default `<None>`

Authentication URL

system_scope

Type unknown type

Default <None>

Scope for system operations

domain_id

Type unknown type

Default <None>

Domain ID to scope to

domain_name

Type unknown type

Default <None>

Domain name to scope to

project_id

Type unknown type

Default <None>

Project ID to scope to

project_name

Type unknown type

Default <None>

Project name to scope to

project_domain_id

Type unknown type

Default <None>

Domain ID containing project

project_domain_name

Type unknown type

Default <None>

Domain name containing project

trust_id

Type unknown type

Default <None>

Trust ID

default_domain_id

Type unknown type

Default <None>

Optional domain ID to use with v3 and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

default_domain_name

Type unknown type

Default <None>

Optional domain name to use with v3 API and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

user_id

Type unknown type

Default <None>

User ID

username

Type unknown type

Default <None>

Username

Table 78: Deprecated Variations

Group	Name
neutron	user-name
neutron	user_name

user_domain_id

Type unknown type

Default <None>

Users domain id

user_domain_name

Type unknown type

Default <None>

Users domain name

password

Type unknown type

Default <None>

Users password

tenant_id

Type unknown type

Default <None>

Tenant ID

tenant_name

Type unknown type

Default <None>

Tenant Name

service_type

Type string

Default `network`

The default service_type for endpoint URL discovery.

service_name

Type string

Default `<None>`

The default service_name for endpoint URL discovery.

valid_interfaces

Type list

Default `internal,public`

List of interfaces, in order of preference, for endpoint URL.

region_name

Type string

Default `<None>`

The default region_name for endpoint URL discovery.

endpoint_override

Type string

Default `<None>`

Always use this endpoint URL for requests for this client. NOTE: The unversioned endpoint should be specified here; to request a particular API version, use the *version*, *min-version*, and/or *max-version* options.

connect_retries

Type integer

Default `<None>`

The maximum number of retries that should be attempted for connection errors.

connect_retry_delay

Type floating point

Default `<None>`

Delay (in seconds) between two retries for connection errors. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

status_code_retries

Type integer

Default `<None>`

The maximum number of retries that should be attempted for retriable HTTP status codes.

status_code_retry_delay

Type floating point

Default `<None>`

Delay (in seconds) between two retries for retriable status codes. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

notifications

Most of the actions in Nova which manipulate the system state generate notifications which are posted to the messaging component (e.g. RabbitMQ) and can be consumed by any service outside the OpenStack. More technical details at https://docs.openstack.org/nova/latest/reference/notifications.html

`notify_on_state_change`

Type string

Default `<None>`

Valid Values <None>, vm_state, vm_and_task_state

If set, send compute.instance.update notifications on instance state changes.

Please refer to https://docs.openstack.org/nova/latest/reference/notifications.html for additional information on notifications.

Possible values

<None> no notifications

vm_state Notifications are sent with VM state transition information in the `old_state` and `state` fields. The `old_task_state` and `new_task_state` fields will be set to the current task_state of the instance

vm_and_task_state Notifications are sent with VM and task state transition information

Table 79: Deprecated Variations

Group	Name
DEFAULT	notify_on_state_change

`default_level`

Type string

Default `INFO`

Valid Values DEBUG, INFO, WARN, ERROR, CRITICAL

Default notification level for outgoing notifications.

Table 80: Deprecated Variations

Group	Name
DEFAULT	default_notification_level

`notification_format`

Type string

Default `unversioned`

Valid Values both, versioned, unversioned

Specifies which notification format shall be emitted by nova.

The versioned notification interface are in feature parity with the legacy interface and the versioned interface is actively developed so new consumers should used the versioned interface.

However, the legacy interface is heavily used by ceilometer and other mature OpenStack components so it remains the default.

Note that notifications can be completely disabled by setting `driver=noop` in the `[oslo_messaging_notifications]` group.

The list of versioned notifications is visible in https://docs.openstack.org/nova/latest/reference/notifications.html

Possible values

both Both the legacy unversioned and the new versioned notifications are emitted

versioned Only the new versioned notifications are emitted

unversioned Only the legacy unversioned notifications are emitted

Table 81: Deprecated Variations

Group	Name
DEFAULT	notification_format

`versioned_notifications_topics`

Type list

Default `versioned_notifications`

Specifies the topics for the versioned notifications issued by nova.

The default value is fine for most deployments and rarely needs to be changed. However, if you have a third-party service that consumes versioned notifications, it might be worth getting a topic for that service. Nova will send a message containing a versioned notification payload to each topic queue in this list.

The list of versioned notifications is visible in https://docs.openstack.org/nova/latest/reference/notifications.html

`bdms_in_notifications`

Type boolean

Default `false`

If enabled, include block device information in the versioned notification payload. Sending block device information is disabled by default as providing that information can incur some overhead on the system since the information may need to be loaded from the database.

osapi_v21

`project_id_regex`

Type string

Default `<None>`

This option is a string representing a regular expression (regex) that matches the project_id as contained in URLs. If not set, it will match normal UUIDs created by keystone.

Possible values:

- A string representing any legal regular expression

> **Warning:** This option is deprecated for removal since 13.0.0. Its value may be silently ignored in the future.
>
> **Reason** Recent versions of nova constrain project IDs to hexadecimal characters and dashes. If your installation uses IDs outside of this range, you should use this option to provide your own regex and give you time to migrate offending projects to valid IDs before the next release.

oslo_concurrency

`disable_process_locking`

Type boolean

Default `false`

Enables or disables inter-process locks.

Table 82: Deprecated Variations

Group	Name
DEFAULT	disable_process_locking

`lock_path`

Type string

Default `<None>`

Directory to use for lock files. For security, the specified directory should only be writable by the user running the processes that need locking. Defaults to environment variable OSLO_LOCK_PATH. If external locks are used, a lock path must be set.

Table 83: Deprecated Variations

Group	Name
DEFAULT	lock_path

oslo_messaging_amqp

`container_name`

Type string

Default `<None>`

Name for the AMQP container. must be globally unique. Defaults to a generated UUID

Table 84: Deprecated Variations

Group	Name
amqp1	container_name

idle_timeout

Type integer

Default `0`

Timeout for inactive connections (in seconds)

Table 85: Deprecated Variations

Group	Name
amqp1	idle_timeout

trace

Type boolean

Default `false`

Debug: dump AMQP frames to stdout

Table 86: Deprecated Variations

Group	Name
amqp1	trace

ssl

Type boolean

Default `false`

Attempt to connect via SSL. If no other ssl-related parameters are given, it will use the systems CA-bundle to verify the servers certificate.

ssl_ca_file

Type string

Default `''`

CA certificate PEM file used to verify the servers certificate

Table 87: Deprecated Variations

Group	Name
amqp1	ssl_ca_file

ssl_cert_file

Type string

Default `''`

Self-identifying certificate PEM file for client authentication

Table 88: Deprecated Variations

Group	Name
amqp1	ssl_cert_file

ssl_key_file

Type string

Default ''

Private key PEM file used to sign ssl_cert_file certificate (optional)

Table 89: Deprecated Variations

Group	Name
amqp1	ssl_key_file

ssl_key_password

Type string

Default <None>

Password for decrypting ssl_key_file (if encrypted)

Table 90: Deprecated Variations

Group	Name
amqp1	ssl_key_password

ssl_verify_vhost

Type boolean

Default false

By default SSL checks that the name in the servers certificate matches the hostname in the transport_url. In some configurations it may be preferable to use the virtual hostname instead, for example if the server uses the Server Name Indication TLS extension (rfc6066) to provide a certificate per virtual host. Set ssl_verify_vhost to True if the servers SSL certificate uses the virtual host name instead of the DNS name.

sasl_mechanisms

Type string

Default ''

Space separated list of acceptable SASL mechanisms

Table 91: Deprecated Variations

Group	Name
amqp1	sasl_mechanisms

sasl_config_dir

Type string

Default ''

Path to directory that contains the SASL configuration

Table 92: Deprecated Variations

Group	Name
amqp1	sasl_config_dir

sasl_config_name

Type string

Default ''

Name of configuration file (without .conf suffix)

Table 93: Deprecated Variations

Group	Name
amqp1	sasl_config_name

sasl_default_realm

Type string

Default ''

SASL realm to use if no realm present in username

connection_retry_interval

Type integer

Default 1

Minimum Value 1

Seconds to pause before attempting to re-connect.

connection_retry_backoff

Type integer

Default 2

Minimum Value 0

Increase the connection_retry_interval by this many seconds after each unsuccessful failover attempt.

connection_retry_interval_max

Type integer

Default 30

Minimum Value 1

Maximum limit for connection_retry_interval + connection_retry_backoff

link_retry_delay

Type integer

Default 10

Minimum Value 1

Time to pause between re-connecting an AMQP 1.0 link that failed due to a recoverable error.

default_reply_retry

Type integer

Default 0

Minimum Value -1

The maximum number of attempts to re-send a reply message which failed due to a recoverable error.

default_reply_timeout

Type integer

Default 30

Minimum Value 5

The deadline for an rpc reply message delivery.

default_send_timeout

Type integer

Default 30

Minimum Value 5

The deadline for an rpc cast or call message delivery. Only used when caller does not provide a timeout expiry.

default_notify_timeout

Type integer

Default 30

Minimum Value 5

The deadline for a sent notification message delivery. Only used when caller does not provide a timeout expiry.

default_sender_link_timeout

Type integer

Default 600

Minimum Value 1

The duration to schedule a purge of idle sender links. Detach link after expiry.

addressing_mode

Type string

Default dynamic

Indicates the addressing mode used by the driver. Permitted values: legacy - use legacy non-routable addressing routable - use routable addresses dynamic - use legacy addresses if the message bus does not support routing otherwise use routable addressing

pseudo_vhost

Type boolean

Default true

Enable virtual host support for those message buses that do not natively support virtual hosting (such as qpidd). When set to true the virtual host name will be added to all message bus addresses, effectively creating a private subnet per virtual host. Set to False if the message bus supports virtual hosting using the hostname field in the AMQP 1.0 Open performative as the name of the virtual host.

server_request_prefix

Type string

Default `exclusive`

address prefix used when sending to a specific server

Table 94: Deprecated Variations

Group	Name
amqp1	server_request_prefix

broadcast_prefix

Type string

Default `broadcast`

address prefix used when broadcasting to all servers

Table 95: Deprecated Variations

Group	Name
amqp1	broadcast_prefix

group_request_prefix

Type string

Default `unicast`

address prefix when sending to any server in group

Table 96: Deprecated Variations

Group	Name
amqp1	group_request_prefix

rpc_address_prefix

Type string

Default `openstack.org/om/rpc`

Address prefix for all generated RPC addresses

notify_address_prefix

Type string

Default `openstack.org/om/notify`

Address prefix for all generated Notification addresses

multicast_address

Type string

Default `multicast`

Appended to the address prefix when sending a fanout message. Used by the message bus to identify fanout messages.

unicast_address

Type string

Default `unicast`

Appended to the address prefix when sending to a particular RPC/Notification server. Used by the message bus to identify messages sent to a single destination.

anycast_address

Type string

Default `anycast`

Appended to the address prefix when sending to a group of consumers. Used by the message bus to identify messages that should be delivered in a round-robin fashion across consumers.

default_notification_exchange

Type string

Default `<None>`

Exchange name used in notification addresses. Exchange name resolution precedence: Target.exchange if set else default_notification_exchange if set else control_exchange if set else notify

default_rpc_exchange

Type string

Default `<None>`

Exchange name used in RPC addresses. Exchange name resolution precedence: Target.exchange if set else default_rpc_exchange if set else control_exchange if set else rpc

reply_link_credit

Type integer

Default `200`

Minimum Value 1

Window size for incoming RPC Reply messages.

rpc_server_credit

Type integer

Default `100`

Minimum Value 1

Window size for incoming RPC Request messages

notify_server_credit

Type integer

Default `100`

Minimum Value 1

Window size for incoming Notification messages

pre_settled

Type multi-valued

Default `rpc-cast`

Default `rpc-reply`

Send messages of this type pre-settled. Pre-settled messages will not receive acknowledgement from the peer. Note well: pre-settled messages may be silently discarded if the delivery fails. Permitted values: rpc-call - send RPC Calls pre-settled rpc-reply- send RPC Replies pre-settled rpc-cast - Send RPC Casts pre-settled notify - Send Notifications pre-settled

oslo_messaging_kafka

kafka_max_fetch_bytes

Type integer

Default `1048576`

Max fetch bytes of Kafka consumer

kafka_consumer_timeout

Type floating point

Default `1.0`

Default timeout(s) for Kafka consumers

pool_size

Type integer

Default `10`

Pool Size for Kafka Consumers

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** Driver no longer uses connection pool.

conn_pool_min_size

Type integer

Default `2`

The pool size limit for connections expiration policy

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** Driver no longer uses connection pool.

conn_pool_ttl

Type integer

Default `1200`

The time-to-live in sec of idle connections in the pool

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.

Reason Driver no longer uses connection pool.

consumer_group

Type string

Default `oslo_messaging_consumer`

Group id for Kafka consumer. Consumers in one group will coordinate message consumption

producer_batch_timeout

Type floating point

Default `0.0`

Upper bound on the delay for KafkaProducer batching in seconds

producer_batch_size

Type integer

Default `16384`

Size of batch for the producer async send

compression_codec

Type string

Default `none`

Valid Values none, gzip, snappy, lz4, zstd

The compression codec for all data generated by the producer. If not set, compression will not be used. Note that the allowed values of this depend on the kafka version

enable_auto_commit

Type boolean

Default `false`

Enable asynchronous consumer commits

max_poll_records

Type integer

Default `500`

The maximum number of records returned in a poll call

security_protocol

Type string

Default `PLAINTEXT`

Valid Values PLAINTEXT, SASL_PLAINTEXT, SSL, SASL_SSL

Protocol used to communicate with brokers

sasl_mechanism

Type string

Default `PLAIN`

Mechanism when security protocol is SASL

ssl_cafile

Type string

Default ''

CA certificate PEM file used to verify the server certificate

oslo_messaging_notifications

driver

Type multi-valued

Default ''

The Drivers(s) to handle sending notifications. Possible values are messaging, messagingv2, routing, log, test, noop

Table 97: Deprecated Variations

Group	Name
DEFAULT	notification_driver

transport_url

Type string

Default <None>

A URL representing the messaging driver to use for notifications. If not set, we fall back to the same configuration used for RPC.

Table 98: Deprecated Variations

Group	Name
DEFAULT	notification_transport_url

topics

Type list

Default notifications

AMQP topic used for OpenStack notifications.

Table 99: Deprecated Variations

Group	Name
rpc_notifier2	topics
DEFAULT	notification_topics

retry

Type integer

Default -1

The maximum number of attempts to re-send a notification message which failed to be delivered due to a recoverable error. 0 - No retry, -1 - indefinite

oslo_messaging_rabbit

amqp_durable_queues

Type boolean

Default `false`

Use durable queues in AMQP.

amqp_auto_delete

Type boolean

Default `false`

Auto-delete queues in AMQP.

Table 100: Deprecated Variations

Group	Name
DEFAULT	amqp_auto_delete

ssl

Type boolean

Default `false`

Connect over SSL.

Table 101: Deprecated Variations

Group	Name
oslo_messaging_rabbit	rabbit_use_ssl

ssl_version

Type string

Default `''`

SSL version to use (valid only if SSL enabled). Valid values are TLSv1 and SSLv23. SSLv2, SSLv3, TLSv1_1, and TLSv1_2 may be available on some distributions.

Table 102: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_ssl_version

ssl_key_file

Type string

Default `''`

SSL key file (valid only if SSL enabled).

Table 103: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_ssl_keyfile

ssl_cert_file

Type string

Default ''

SSL cert file (valid only if SSL enabled).

Table 104: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_ssl_certfile

ssl_ca_file

Type string

Default ''

SSL certification authority file (valid only if SSL enabled).

Table 105: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_ssl_ca_certs

heartbeat_in_pthread

Type boolean

Default false

EXPERIMENTAL: Run the health check heartbeat threadthrough a native python thread. By default if thisoption isnt provided the health check heartbeat willinherit the execution model from the parent process. Byexample if the parent process have monkey patched thestdlib by using eventlet/greenlet then the heartbeatwill be run through a green thread.

kombu_reconnect_delay

Type floating point

Default 1.0

How long to wait before reconnecting in response to an AMQP consumer cancel notification.

Table 106: Deprecated Variations

Group	Name
DEFAULT	kombu_reconnect_delay

kombu_compression

Type string

Default <None>

EXPERIMENTAL: Possible values are: gzip, bz2. If not set compression will not be used. This option may not be available in future versions.

kombu_missing_consumer_retry_timeout

Type integer

Default 60

How long to wait a missing client before abandoning to send it its replies. This value should not be longer than rpc_response_timeout.

Table 107: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_reconnect_timeout

kombu_failover_strategy

Type string

Default `round-robin`

Valid Values round-robin, shuffle

Determines how the next RabbitMQ node is chosen in case the one we are currently connected to becomes unavailable. Takes effect only if more than one RabbitMQ node is provided in config.

rabbit_login_method

Type string

Default `AMQPLAIN`

Valid Values PLAIN, AMQPLAIN, RABBIT-CR-DEMO

The RabbitMQ login method.

Table 108: Deprecated Variations

Group	Name
DEFAULT	rabbit_login_method

rabbit_retry_interval

Type integer

Default `1`

How frequently to retry connecting with RabbitMQ.

rabbit_retry_backoff

Type integer

Default `2`

How long to backoff for between retries when connecting to RabbitMQ.

Table 109: Deprecated Variations

Group	Name
DEFAULT	rabbit_retry_backoff

rabbit_interval_max

Type integer

Default `30`

Maximum interval of RabbitMQ connection retries. Default is 30 seconds.

rabbit_ha_queues

Type boolean

Default `false`

Try to use HA queues in RabbitMQ (x-ha-policy: all). If you change this option, you must wipe the RabbitMQ database. In RabbitMQ 3.0, queue mirroring is no longer controlled by the x-ha-policy argument when declaring a queue. If you just want to make sure that all queues (except those with auto-generated names) are mirrored across all nodes, run: rabbitmqctl set_policy HA ^(?!amq.).* {ha-mode: all}

Table 110: Deprecated Variations

Group	Name
DEFAULT	rabbit_ha_queues

`rabbit_transient_queues_ttl`

Type integer

Default `1800`

Minimum Value 1

Positive integer representing duration in seconds for queue TTL (x-expires). Queues which are unused for the duration of the TTL are automatically deleted. The parameter affects only reply and fanout queues.

`rabbit_qos_prefetch_count`

Type integer

Default `0`

Specifies the number of messages to prefetch. Setting to zero allows unlimited messages.

`heartbeat_timeout_threshold`

Type integer

Default `60`

Number of seconds after which the Rabbit broker is considered down if heartbeats keep-alive fails (0 disables heartbeat).

`heartbeat_rate`

Type integer

Default `2`

How often times during the heartbeat_timeout_threshold we check the heartbeat.

`direct_mandatory_flag`

Type boolean

Default `true`

(DEPRECATED) Enable/Disable the RabbitMQ mandatory flag for direct send. The direct send is used as reply, so the MessageUndeliverable exception is raised in case the client queue does not exist.MessageUndeliverable exception will be used to loop for a timeout to lets a chance to sender to recover.This flag is deprecated and it will not be possible to deactivate this functionality anymore

Warning: This option is deprecated for removal. Its value may be silently ignored in the future.

Reason Mandatory flag no longer deactivable.

enable_cancel_on_failover

Type boolean

Default `false`

Enable x-cancel-on-ha-failover flag so that rabbitmq server will cancel and notify consumerswhen queue is down

oslo_middleware

max_request_body_size

Type integer

Default `114688`

The maximum body size for each request, in bytes.

Table 111: Deprecated Variations

Group	Name
DEFAULT	osapi_max_request_body_size
DEFAULT	max_request_body_size

secure_proxy_ssl_header

Type string

Default `X-Forwarded-Proto`

The HTTP Header that will be used to determine what the original request protocol scheme was, even if it was hidden by a SSL termination proxy.

Warning: This option is deprecated for removal. Its value may be silently ignored in the future.

enable_proxy_headers_parsing

Type boolean

Default `false`

Whether the application is behind a proxy or not. This determines if the middleware should parse the headers or not.

oslo_policy

enforce_scope

Type boolean

Default `false`

This option controls whether or not to enforce scope when evaluating policies. If `True`, the scope of the token used in the request is compared to the `scope_types` of the policy being enforced. If the scopes do not match, an `InvalidScope` exception will be raised. If `False`, a message will be logged informing operators that policies are being invoked with mismatching scope.

policy_file

Type string

Default `policy.json`

The relative or absolute path of a file that maps roles to permissions for a given service. Relative paths must be specified in relation to the configuration file setting this option.

Table 112: Deprecated Variations

Group	Name
DEFAULT	policy_file

policy_default_rule

Type string

Default `default`

Default rule. Enforced when a requested rule is not found.

Table 113: Deprecated Variations

Group	Name
DEFAULT	policy_default_rule

policy_dirs

Type multi-valued

Default `policy.d`

Directories where policy configuration files are stored. They can be relative to any directory in the search path defined by the config_dir option, or absolute paths. The file defined by policy_file must exist for these directories to be searched. Missing or empty directories are ignored.

Table 114: Deprecated Variations

Group	Name
DEFAULT	policy_dirs

remote_content_type

Type string

Default `application/x-www-form-urlencoded`

Valid Values application/x-www-form-urlencoded, application/json

Content Type to send and receive data for REST based policy check

remote_ssl_verify_server_crt

Type boolean

Default `false`

server identity verification for REST based policy check

remote_ssl_ca_crt_file

Type string

Default `<None>`

Absolute path to ca cert file for REST based policy check

remote_ssl_client_crt_file

Type string

Default <None>

Absolute path to client cert for REST based policy check

remote_ssl_client_key_file

Type string

Default <None>

Absolute path client key file REST based policy check

pci

alias

Type multi-valued

Default ''

An alias for a PCI passthrough device requirement.

This allows users to specify the alias in the extra specs for a flavor, without needing to repeat all the PCI property requirements.

This should be configured for the `nova-api` service and, assuming you wish to use move operations, for each `nova-compute` service.

Possible Values:

- A dictionary of JSON values which describe the aliases. For example:

```
alias = {
  "name": "QuickAssist",
  "product_id": "0443",
  "vendor_id": "8086",
  "device_type": "type-PCI",
  "numa_policy": "required"
}
```

This defines an alias for the Intel QuickAssist card. (multi valued). Valid key values are :

name Name of the PCI alias.

product_id Product ID of the device in hexadecimal.

vendor_id Vendor ID of the device in hexadecimal.

device_type Type of PCI device. Valid values are: `type-PCI`, `type-PF` and `type-VF`. Note that `"device_type": "type-PF"` **must** be specified if you wish to passthrough a device that supports SR-IOV in its entirety.

numa_policy Required NUMA affinity of device. Valid values are: `legacy`, `preferred` and `required`.

- Supports multiple aliases by repeating the option (not by specifying a list value):

```
alias = {
  "name": "QuickAssist-1",
  "product_id": "0443",
  "vendor_id": "8086",
  "device_type": "type-PCI",
  "numa_policy": "required"
}
alias = {
  "name": "QuickAssist-2",
  "product_id": "0444",
  "vendor_id": "8086",
  "device_type": "type-PCI",
  "numa_policy": "required"
}
```

Table 115: Deprecated Variations

Group	Name
DEFAULT	pci_alias

passthrough_whitelist

Type multi-valued

Default ''

White list of PCI devices available to VMs.

Possible values:

- A JSON dictionary which describe a whitelisted PCI device. It should take the following format:

```
["vendor_id": "<id>",] ["product_id": "<id>",]
["address": "[[[[<domain>]:]<bus>]:][<slot>][.[<function>]]" |
 "devname": "<name>",]
{"<tag>": "<tag_value>",}
```

Where [indicates zero or one occurrences, { indicates zero or multiple occurrences, and | mutually exclusive options. Note that any missing fields are automatically wildcarded.

Valid key values are :

vendor_id Vendor ID of the device in hexadecimal.

product_id Product ID of the device in hexadecimal.

address PCI address of the device. Both traditional glob style and regular expression syntax is supported.

devname Device name of the device (for e.g. interface name). Not all PCI devices have a name.

<tag> Additional <tag> and <tag_value> used for matching PCI devices. Supported <tag> values are :

- physical_network
- trusted

Valid examples are:

```
passthrough_whitelist = {"devname":"eth0",
                         "physical_network":"physnet"}
passthrough_whitelist = {"address":"*:0a:00.*"}
passthrough_whitelist = {"address":":0a:00.",
                         "physical_network":"physnet1"}
passthrough_whitelist = {"vendor_id":"1137",
                         "product_id":"0071"}
passthrough_whitelist = {"vendor_id":"1137",
                         "product_id":"0071",
                         "address": "0000:0a:00.1",
                         "physical_network":"physnet1"}
passthrough_whitelist = {"address":{"domain": ".*",
                                    "bus": "02", "slot": "01",
                                    "function": "[2-7]"},
                         "physical_network":"physnet1"}
passthrough_whitelist = {"address":{"domain": ".*",
                                    "bus": "02", "slot": "0[1-2]",
                                    "function": ".*"},
                         "physical_network":"physnet1"}
passthrough_whitelist = {"devname": "eth0", "physical_network":"physnet1",
                         "trusted": "true"}
```

The following are invalid, as they specify mutually exclusive options:

```
passthrough_whitelist = {"devname":"eth0",
                         "physical_network":"physnet",
                         "address":"*:0a:00.*"}
```

- A JSON list of JSON dictionaries corresponding to the above format. For example:

```
passthrough_whitelist = [{"product_id":"0001", "vendor_id":"8086"},
                         {"product_id":"0002", "vendor_id":"8086"}]
```

Table 116: Deprecated Variations

Group	Name
DEFAULT	pci_passthrough_whitelist

placement

cafile

Type string

Default <None>

PEM encoded Certificate Authority to use when verifying HTTPs connections.

certfile

Type string

Default <None>

PEM encoded client certificate cert file

keyfile

Type string

Default `<None>`

PEM encoded client certificate key file

`insecure`

Type boolean

Default `false`

Verify HTTPS connections.

`timeout`

Type integer

Default `<None>`

Timeout value for http requests

`collect_timing`

Type boolean

Default `false`

Collect per-API call timing information.

`split_loggers`

Type boolean

Default `false`

Log requests to multiple loggers.

`auth_type`

Type unknown type

Default `<None>`

Authentication type to load

Table 117: Deprecated Variations

Group	Name
placement	auth_plugin

`auth_section`

Type unknown type

Default `<None>`

Config Section from which to load plugin specific options

`auth_url`

Type unknown type

Default `<None>`

Authentication URL

`system_scope`

Type unknown type

Default `<None>`

Scope for system operations

domain_id

Type unknown type

Default <None>

Domain ID to scope to

domain_name

Type unknown type

Default <None>

Domain name to scope to

project_id

Type unknown type

Default <None>

Project ID to scope to

project_name

Type unknown type

Default <None>

Project name to scope to

project_domain_id

Type unknown type

Default <None>

Domain ID containing project

project_domain_name

Type unknown type

Default <None>

Domain name containing project

trust_id

Type unknown type

Default <None>

Trust ID

default_domain_id

Type unknown type

Default <None>

Optional domain ID to use with v3 and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

default_domain_name

Type unknown type

Default <None>

Optional domain name to use with v3 API and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

user_id

Type unknown type

Default <None>

User ID

username

Type unknown type

Default <None>

Username

Table 118: Deprecated Variations

Group	Name
placement	user-name
placement	user_name

user_domain_id

Type unknown type

Default <None>

Users domain id

user_domain_name

Type unknown type

Default <None>

Users domain name

password

Type unknown type

Default <None>

Users password

tenant_id

Type unknown type

Default <None>

Tenant ID

tenant_name

Type unknown type

Default <None>

Tenant Name

service_type

Type string

Default `placement`

The default service_type for endpoint URL discovery.

service_name

Type string

Default `<None>`

The default service_name for endpoint URL discovery.

valid_interfaces

Type list

Default `internal,public`

List of interfaces, in order of preference, for endpoint URL.

region_name

Type string

Default `<None>`

The default region_name for endpoint URL discovery.

endpoint_override

Type string

Default `<None>`

Always use this endpoint URL for requests for this client. NOTE: The unversioned endpoint should be specified here; to request a particular API version, use the *version*, *min-version*, and/or *max-version* options.

connect_retries

Type integer

Default `<None>`

The maximum number of retries that should be attempted for connection errors.

connect_retry_delay

Type floating point

Default `<None>`

Delay (in seconds) between two retries for connection errors. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

status_code_retries

Type integer

Default `<None>`

The maximum number of retries that should be attempted for retriable HTTP status codes.

status_code_retry_delay

Type floating point

Default `<None>`

Delay (in seconds) between two retries for retriable status codes. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

powervm

PowerVM options allow cloud administrators to configure how OpenStack will work with the PowerVM hypervisor.

proc_units_factor

Type floating point

Default `0.1`

Minimum Value 0.05

Maximum Value 1

Factor used to calculate the amount of physical processor compute power given to each vCPU. E.g. A value of 1.0 means a whole physical processor, whereas 0.05 means 1/20th of a physical processor.

disk_driver

Type string

Default `localdisk`

Valid Values localdisk, ssp

The disk driver to use for PowerVM disks. PowerVM provides support for localdisk and PowerVM Shared Storage Pool disk drivers.

Related options:

- volume_group_name - required when using localdisk

volume_group_name

Type string

Default `''`

Volume Group to use for block device operations. If disk_driver is localdisk, then this attribute must be specified. It is strongly recommended NOT to use rootvg since that is used by the management partition and filling it will cause failures.

privsep

Configuration options for the oslo.privsep daemon. Note that this group name can be changed by the consuming service. Check the services docs to see if this is the case.

user

Type string

Default `<None>`

User that the privsep daemon should run as.

group

Type string

Default `<None>`

Group that the privsep daemon should run as.

capabilities

Type unknown type

Default `[]`

List of Linux capabilities retained by the privsep daemon.

thread_pool_size

Type integer

Default `multiprocessing.cpu_count()`

Minimum Value 1

This option has a sample default set, which means that its actual default value may vary from the one documented above.

The number of threads available for privsep to concurrently run processes. Defaults to the number of CPU cores in the system.

helper_command

Type string

Default `<None>`

Command to invoke to start the privsep daemon if not using the fork method. If not specified, a default is generated using sudo privsep-helper and arguments designed to recreate the current configuration. This command must accept suitable –privsep_context and –privsep_sock_path arguments.

profiler

enabled

Type boolean

Default `false`

Enable the profiling for all services on this node.

Default value is False (fully disable the profiling feature).

Possible values:

- True: Enables the feature
- False: Disables the feature. The profiling cannot be started via this project operations. If the profiling is triggered by another project, this project part will be empty.

Table 119: Deprecated Variations

Group	Name
profiler	profiler_enabled

trace_sqlalchemy

Type boolean

Default `false`

Enable SQL requests profiling in services.

Default value is False (SQL requests wont be traced).

Possible values:

- True: Enables SQL requests profiling. Each SQL query will be part of the trace and can the be analyzed by how much time was spent for that.
- False: Disables SQL requests profiling. The spent time is only shown on a higher level of operations. Single SQL queries cannot be analyzed this way.

hmac_keys

Type string

Default `SECRET_KEY`

Secret key(s) to use for encrypting context data for performance profiling.

This string value should have the following format: <key1>[,<key2>,<keyn>], where each key is some random string. A user who triggers the profiling via the REST API has to set one of these keys in the headers of the REST API call to include profiling results of this node for this particular project.

Both enabled flag and hmac_keys config options should be set to enable profiling. Also, to generate correct profiling information across all services at least one key needs to be consistent between OpenStack projects. This ensures it can be used from client side to generate the trace, containing information from all possible resources.

connection_string

Type string

Default `messaging://`

Connection string for a notifier backend.

Default value is `messaging://` which sets the notifier to oslo_messaging.

Examples of possible values:

- `messaging://` - use oslo_messaging driver for sending spans.
- `redis://127.0.0.1:6379` - use redis driver for sending spans.
- `mongodb://127.0.0.1:27017` - use mongodb driver for sending spans.
- `elasticsearch://127.0.0.1:9200` - use elasticsearch driver for sending spans.
- `jaeger://127.0.0.1:6831` - use jaeger tracing as driver for sending spans.

es_doc_type

Type string

Default `notification`

Document type for notification indexing in elasticsearch.

es_scroll_time

Type string

Default `2m`

This parameter is a time value parameter (for example: es_scroll_time=2m), indicating for how long the nodes that participate in the search will maintain relevant resources in order to continue and support it.

es_scroll_size

Type integer

Default `10000`

Elasticsearch splits large requests in batches. This parameter defines maximum size of each batch (for example: es_scroll_size=10000).

`socket_timeout`

Type floating point

Default `0.1`

Redissentinel provides a timeout option on the connections. This parameter defines that timeout (for example: socket_timeout=0.1).

`sentinel_service_name`

Type string

Default `mymaster`

Redissentinel uses a service name to identify a master redis service. This parameter defines the name (for example: `sentinal_service_name=mymaster`).

`filter_error_trace`

Type boolean

Default `false`

Enable filter traces that contain error/exception to a separated place.

Default value is set to False.

Possible values:

- True: Enable filter traces that contain error/exception.
- False: Disable the filter.

quota

Quota options allow to manage quotas in openstack deployment.

`instances`

Type integer

Default `10`

Minimum Value -1

The number of instances allowed per project.

Possible Values

- A positive integer or 0.
- -1 to disable the quota.

Table 120: Deprecated Variations

Group	Name
DEFAULT	quota_instances

cores

Type integer

Default `20`

Minimum Value -1

The number of instance cores or vCPUs allowed per project.

Possible values:

- A positive integer or 0.
- -1 to disable the quota.

Table 121: Deprecated Variations

Group	Name
DEFAULT	quota_cores

ram

Type integer

Default `51200`

Minimum Value -1

The number of megabytes of instance RAM allowed per project.

Possible values:

- A positive integer or 0.
- -1 to disable the quota.

Table 122: Deprecated Variations

Group	Name
DEFAULT	quota_ram

floating_ips

Type integer

Default `10`

Minimum Value -1

The number of floating IPs allowed per project.

Floating IPs are not allocated to instances by default. Users need to select them from the pool configured by the OpenStack administrator to attach to their instances.

Possible values:

- A positive integer or 0.
- -1 to disable the quota.

Table 123: Deprecated Variations

Group	Name
DEFAULT	quota_floating_ips

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

fixed_ips

Type integer

Default `-1`

Minimum Value -1

The number of fixed IPs allowed per project.

Unlike floating IPs, fixed IPs are allocated dynamically by the network component when instances boot up. This quota value should be at least the number of instances allowed

Possible values:

- A positive integer or 0.
- -1 to disable the quota.

Table 124: Deprecated Variations

Group	Name
DEFAULT	quota_fixed_ips

> **Warning:** This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.
>
> **Reason** nova-network is deprecated, as are any related configuration options.

metadata_items

Type integer

Default `128`

Minimum Value -1

The number of metadata items allowed per instance.

Users can associate metadata with an instance during instance creation. This metadata takes the form of key-value pairs.

Possible values:

- A positive integer or 0.
- -1 to disable the quota.

Table 125: Deprecated Variations

Group	Name
DEFAULT	quota_metadata_items

injected_files

Type integer

Default `5`

Minimum Value -1

The number of injected files allowed.

File injection allows users to customize the personality of an instance by injecting data into it upon boot. Only text file injection is permitted: binary or ZIP files are not accepted. During file injection, any existing files that match specified files are renamed to include `.bak` extension appended with a timestamp.

Possible values:

- A positive integer or 0.
- -1 to disable the quota.

Table 126: Deprecated Variations

Group	Name
DEFAULT	quota_injected_files

`injected_file_content_bytes`

Type integer

Default `10240`

Minimum Value -1

The number of bytes allowed per injected file.

Possible values:

- A positive integer or 0.
- -1 to disable the quota.

Table 127: Deprecated Variations

Group	Name
DEFAULT	quota_injected_file_content_bytes

`injected_file_path_length`

Type integer

Default `255`

Minimum Value -1

The maximum allowed injected file path length.

Possible values:

- A positive integer or 0.
- -1 to disable the quota.

Table 128: Deprecated Variations

Group	Name
DEFAULT	quota_injected_file_path_length

`security_groups`

Type integer

Default `10`

Minimum Value -1

The number of security groups per project.

Possible values:

- A positive integer or 0.
- -1 to disable the quota.

Table 129: Deprecated Variations

Group	Name
DEFAULT	quota_security_groups

Warning: This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

`security_group_rules`

Type integer

Default `20`

Minimum Value -1

The number of security rules per security group.

The associated rules in each security group control the traffic to instances in the group.

Possible values:

- A positive integer or 0.
- -1 to disable the quota.

Table 130: Deprecated Variations

Group	Name
DEFAULT	quota_security_group_rules

Warning: This option is deprecated for removal since 15.0.0. Its value may be silently ignored in the future.

Reason nova-network is deprecated, as are any related configuration options.

`key_pairs`

Type integer

Default `100`

Minimum Value -1

The maximum number of key pairs allowed per user.

Users can create at least one key pair for each project and use the key pair for multiple instances that belong to that project.

Possible values:

- A positive integer or 0.
- -1 to disable the quota.

Table 131: Deprecated Variations

Group	Name
DEFAULT	quota_key_pairs

server_groups

Type integer

Default `10`

Minimum Value -1

The maxiumum number of server groups per project.

Server groups are used to control the affinity and anti-affinity scheduling policy for a group of servers or instances. Reducing the quota will not affect any existing group, but new servers will not be allowed into groups that have become over quota.

Possible values:

- A positive integer or 0.
- -1 to disable the quota.

Table 132: Deprecated Variations

Group	Name
DEFAULT	quota_server_groups

server_group_members

Type integer

Default `10`

Minimum Value -1

The maximum number of servers per server group.

Possible values:

- A positive integer or 0.
- -1 to disable the quota.

Table 133: Deprecated Variations

Group	Name
DEFAULT	quota_server_group_members

driver

Type string

Default `nova.quota.DbQuotaDriver`

Valid Values nova.quota.DbQuotaDriver, nova.quota.NoopQuotaDriver

Provides abstraction for quota checks. Users can configure a specific driver to use for quota checks.

Possible values

nova.quota.DbQuotaDriver Stores quota limit information in the database and relies on the `quota_*` configuration options for default quota limit values. Counts quota usage on-demand.

nova.quota.NoopQuotaDriver Ignores quota and treats all resources as unlimited.

`recheck_quota`

Type boolean

Default `true`

Recheck quota after resource creation to prevent allowing quota to be exceeded.

This defaults to True (recheck quota after resource creation) but can be set to False to avoid additional load if allowing quota to be exceeded because of racing requests is considered acceptable. For example, when set to False, if a user makes highly parallel REST API requests to create servers, it will be possible for them to create more servers than their allowed quota during the race. If their quota is 10 servers, they might be able to create 50 during the burst. After the burst, they will not be able to create any more servers but they will be able to keep their 50 servers until they delete them.

The initial quota check is done before resources are created, so if multiple parallel requests arrive at the same time, all could pass the quota check and create resources, potentially exceeding quota. When recheck_quota is True, quota will be checked a second time after resources have been created and if the resource is over quota, it will be deleted and OverQuota will be raised, usually resulting in a 403 response to the REST API user. This makes it impossible for a user to exceed their quota with the caveat that it will, however, be possible for a REST API user to be rejected with a 403 response in the event of a collision close to reaching their quota limit, even if the user has enough quota available when they made the request.

`count_usage_from_placement`

Type boolean

Default `false`

Enable the counting of quota usage from the placement service.

Starting in Train, it is possible to count quota usage for cores and ram from the placement service and instances from the API database instead of counting from cell databases.

This works well if there is only one Nova deployment running per placement deployment. However, if an operator is running more than one Nova deployment sharing a placement deployment, they should not set this option to True because currently the placement service has no way to partition resource providers per Nova deployment. When this option is left as the default or set to False, Nova will use the legacy counting method to count quota usage for instances, cores, and ram from its cell databases.

Note that quota usage behavior related to resizes will be affected if this option is set to True. Placement resource allocations are claimed on the destination while holding allocations on the source during a resize, until the resize is confirmed or reverted. During this time, when the server is in VERIFY_RESIZE state, quota usage will reflect resource consumption on both the source and the destination. This can be beneficial as it reserves space for a revert of a downsize, but it also means quota usage will be inflated until a resize is confirmed or reverted.

Behavior will also be different for unscheduled servers in ERROR state. A server in ERROR state that has never been scheduled to a compute host will not have placement allocations, so it will not consume quota usage for cores and ram.

Behavior will be different for servers in SHELVED_OFFLOADED state. A server in SHELVED_OFFLOADED state will not have placement allocations, so it will not consume quota usage for cores and ram. Note that because of this, it will be possible for a request to unshelve a server to be rejected if the user does not have enough quota available to support the cores and ram needed by the server to be unshelved.

The `populate_queued_for_delete` and `populate_user_id` online data migrations must be completed before usage can be counted from placement. Until the data migration is complete, the system will fall back to legacy quota usage counting from cell databases depending on the result of an EXISTS database query during each quota check, if this configuration option is set to True. Operators who want to avoid the performance hit from the EXISTS queries should wait to set this configuration option to True until after they have completed their online data migrations via `nova-manage db online_data_migrations`.

rdp

Options under this group enable and configure Remote Desktop Protocol (RDP) related features.

This group is only relevant to Hyper-V users.

`enabled`

Type boolean

Default `false`

Enable Remote Desktop Protocol (RDP) related features.

Hyper-V, unlike the majority of the hypervisors employed on Nova compute nodes, uses RDP instead of VNC and SPICE as a desktop sharing protocol to provide instance console access. This option enables RDP for graphical console access for virtual machines created by Hyper-V.

Note: RDP should only be enabled on compute nodes that support the Hyper-V virtualization platform.

Related options:

- `compute_driver`: Must be hyperv.

`html5_proxy_base_url`

Type URI

Default `http://127.0.0.1:6083/`

The URL an end user would use to connect to the RDP HTML5 console proxy. The console proxy service is called with this token-embedded URL and establishes the connection to the proper instance.

An RDP HTML5 console proxy service will need to be configured to listen on the address configured here. Typically the console proxy service would be run on a controller node. The localhost address used as default would only work in a single node environment i.e. devstack.

An RDP HTML5 proxy allows a user to access via the web the text or graphical console of any Windows server or workstation using RDP. RDP HTML5 console proxy services include FreeRDP, wsgate. See https://github.com/FreeRDP/FreeRDP-WebConnect

Possible values:

- <scheme>://<ip-address>:<port-number>/

  The scheme must be identical to the scheme configured for the RDP HTML5 console proxy service. It is `http` or `https`.

The IP address must be identical to the address on which the RDP HTML5 console proxy service is listening.

The port must be identical to the port on which the RDP HTML5 console proxy service is listening.

Related options:

- `rdp.enabled`: Must be set to `True` for `html5_proxy_base_url` to be effective.

remote_debug

`host`

Type host address

Default `<None>`

Debug host (IP or name) to connect to. This command line parameter is used when you want to connect to a nova service via a debugger running on a different host.

Note that using the remote debug option changes how Nova uses the eventlet library to support async IO. This could result in failures that do not occur under normal operation. Use at your own risk.

Possible Values:

- IP address of a remote host as a command line parameter to a nova service. For Example:

  /usr/local/bin/nova-compute –config-file /etc/nova/nova.conf –remote_debug-host <IP address where the debugger is running>

`port`

Type port number

Default `<None>`

Minimum Value 0

Maximum Value 65535

Debug port to connect to. This command line parameter allows you to specify the port you want to use to connect to a nova service via a debugger running on different host.

Note that using the remote debug option changes how Nova uses the eventlet library to support async IO. This could result in failures that do not occur under normal operation. Use at your own risk.

Possible Values:

- Port number you want to use as a command line parameter to a nova service. For Example:

  /usr/local/bin/nova-compute –config-file /etc/nova/nova.conf –remote_debug-host <IP address where the debugger is running> –remote_debug-port <port> its listening on>.

scheduler

`driver`

Type string

Default `filter_scheduler`

The class of the driver used by the scheduler. This should be chosen from one of the entrypoints under the namespace nova.scheduler.driver of file setup.cfg. If nothing is specified in this option, the filter_scheduler is used.

Other options are:

- fake_scheduler which is used for testing.

Possible values:

- Any of the drivers included in Nova:
  - filter_scheduler
  - fake_scheduler
- You may also set this to the entry point name of a custom scheduler driver, but you will be responsible for creating and maintaining it in your setup.cfg file.

Related options:

- workers

Table 134: Deprecated Variations

Group	Name
DEFAULT	scheduler_driver

periodic_task_interval

Type integer

Default `60`

Periodic task interval.

This value controls how often (in seconds) to run periodic tasks in the scheduler. The specific tasks that are run for each period are determined by the particular scheduler being used. Currently there are no in-tree scheduler driver that use this option.

If this is larger than the nova-service service_down_time setting, the ComputeFilter (if enabled) may think the compute service is down. As each scheduler can work a little differently than the others, be sure to test this with your selected scheduler.

Possible values:

- An integer, where the integer corresponds to periodic task interval in seconds. 0 uses the default interval (60 seconds). A negative value disables periodic tasks.

Related options:

- `nova-service service_down_time`

max_attempts

Type integer

Default `3`

Minimum Value 1

This is the maximum number of attempts that will be made for a given instance build/move operation. It limits the number of alternate hosts returned by the scheduler. When that list of hosts is exhausted, a MaxRetriesExceeded exception is raised and the instance is set to an error state.

Possible values:

- A positive integer, where the integer corresponds to the max number of attempts that can be made when building or moving an instance.

Table 135: Deprecated Variations

Group	Name
DEFAULT	scheduler_max_attempts

discover_hosts_in_cells_interval

Type integer

Default `-1`

Minimum Value -1

Periodic task interval.

This value controls how often (in seconds) the scheduler should attempt to discover new hosts that have been added to cells. If negative (the default), no automatic discovery will occur.

Deployments where compute nodes come and go frequently may want this enabled, where others may prefer to manually discover hosts when one is added to avoid any overhead from constantly checking. If enabled, every time this runs, we will select any unmapped hosts out of each cell database on every run.

max_placement_results

Type integer

Default `1000`

Minimum Value 1

This setting determines the maximum limit on results received from the placement service during a scheduling operation. It effectively limits the number of hosts that may be considered for scheduling requests that match a large number of candidates.

A value of 1 (the minimum) will effectively defer scheduling to the placement service strictly on will it fit grounds. A higher value will put an upper cap on the number of results the scheduler will consider during the filtering and weighing process. Large deployments may need to set this lower than the total number of hosts available to limit memory consumption, network traffic, etc. of the scheduler.

This option is only used by the FilterScheduler; if you use a different scheduler, this option has no effect.

workers

Type integer

Default `<None>`

Minimum Value 0

Number of workers for the nova-scheduler service. The default will be the number of CPUs available if using the filter_scheduler scheduler driver, otherwise the default will be 1.

limit_tenants_to_placement_aggregate

Type boolean

Default `false`

This setting causes the scheduler to look up a host aggregate with the metadata key of *filter_tenant_id* set to the project of an incoming request, and request results from placement be limited to that aggregate. Multiple tenants may be added to a single aggregate by appending a serial number to the key, such as *filter_tenant_id:123*.

The matching aggregate UUID must be mirrored in placement for proper operation. If no host aggregate with the tenant id is found, or that aggregate does not match one in placement, the result will be the same as not finding any suitable hosts for the request.

See also the placement_aggregate_required_for_tenants option.

placement_aggregate_required_for_tenants

Type boolean

Default `false`

This setting, when limit_tenants_to_placement_aggregate=True, will control whether or not a tenant with no aggregate affinity will be allowed to schedule to any available node. If aggregates are used to limit some tenants but not all, then this should be False. If all tenants should be confined via aggregate, then this should be True to prevent them from receiving unrestricted scheduling to any available node.

See also the limit_tenants_to_placement_aggregate option.

query_placement_for_availability_zone

Type boolean

Default `false`

This setting causes the scheduler to look up a host aggregate with the metadata key of *availability_zone* set to the value provided by an incoming request, and request results from placement be limited to that aggregate.

The matching aggregate UUID must be mirrored in placement for proper operation. If no host aggregate with the *availability_zone* key is found, or that aggregate does not match one in placement, the result will be the same as not finding any suitable hosts.

Note that if you enable this flag, you can disable the (less efficient) AvailabilityZoneFilter in the scheduler.

query_placement_for_image_type_support

Type boolean

Default `false`

This setting causes the scheduler to ask placement only for compute hosts that support the `disk_format` of the image used in the request.

enable_isolated_aggregate_filtering

Type boolean

Default `false`

This setting allows the scheduler to restrict hosts in aggregates based on matching required traits in the aggregate metadata and the instance flavor/image. If an aggregate is configured with a property with key `trait:$TRAIT_NAME` and value `required`, the instance flavor extra_specs and/or image metadata must also contain `trait:$TRAIT_NAME=required` to be eligible to be scheduled to hosts in that aggregate. More technical details at https://docs.openstack.org/nova/latest/reference/isolate-aggregates.html

serial_console

The serial console feature allows you to connect to a guest in case a graphical console like VNC, RDP or SPICE is not available. This is only currently supported for the libvirt, Ironic and hyper-v drivers.

enabled

Type boolean

Default false

Enable the serial console feature.

In order to use this feature, the service nova-serialproxy needs to run. This service is typically executed on the controller node.

port_range

Type string

Default 10000:20000

A range of TCP ports a guest can use for its backend.

Each instance which gets created will use one port out of this range. If the range is not big enough to provide another port for an new instance, this instance wont get launched.

Possible values:

- Each string which passes the regex ^\d+:\d+$ For example 10000:20000. Be sure that the first port number is lower than the second port number and that both are in range from 0 to 65535.

base_url

Type URI

Default ws://127.0.0.1:6083/

The URL an end user would use to connect to the nova-serialproxy service.

The nova-serialproxy service is called with this token enriched URL and establishes the connection to the proper instance.

Related options:

- The IP address must be identical to the address to which the nova-serialproxy service is listening (see option serialproxy_host in this section).
- The port must be the same as in the option serialproxy_port of this section.
- If you choose to use a secured websocket connection, then start this option with wss:// instead of the unsecured ws://. The options cert and key in the [DEFAULT] section have to be set for that.

proxyclient_address

Type string

Default 127.0.0.1

The IP address to which proxy clients (like nova-serialproxy) should connect to get the serial console of an instance.

This is typically the IP address of the host of a nova-compute service.

serialproxy_host

Type string

Default 0.0.0.0

The IP address which is used by the nova-serialproxy service to listen for incoming requests.

The nova-serialproxy service listens on this IP address for incoming connection requests to instances which expose serial console.

Related options:

- Ensure that this is the same IP address which is defined in the option `base_url` of this section or use `0.0.0.0` to listen on all addresses.

`serialproxy_port`

Type port number

Default `6083`

Minimum Value 0

Maximum Value 65535

The port number which is used by the `nova-serialproxy` service to listen for incoming requests.

The `nova-serialproxy` service listens on this port number for incoming connection requests to instances which expose serial console.

Related options:

- Ensure that this is the same port number which is defined in the option `base_url` of this section.

service_user

Configuration options for service to service authentication using a service token. These options allow sending a service token along with the users token when contacting external REST APIs.

`send_service_user_token`

Type boolean

Default `false`

When True, if sending a user token to a REST API, also send a service token.

Nova often reuses the user token provided to the nova-api to talk to other REST APIs, such as Cinder, Glance and Neutron. It is possible that while the user token was valid when the request was made to Nova, the token may expire before it reaches the other service. To avoid any failures, and to make it clear it is Nova calling the service on the users behalf, we include a service token along with the user token. Should the users token have expired, a valid service token ensures the REST API request will still be accepted by the keystone middleware.

`cafile`

Type string

Default `<None>`

PEM encoded Certificate Authority to use when verifying HTTPs connections.

`certfile`

Type string

Default `<None>`

PEM encoded client certificate cert file

`keyfile`

Type string

Default `<None>`

PEM encoded client certificate key file

`insecure`

Type boolean

Default `false`

Verify HTTPS connections.

`timeout`

Type integer

Default `<None>`

Timeout value for http requests

`collect_timing`

Type boolean

Default `false`

Collect per-API call timing information.

`split_loggers`

Type boolean

Default `false`

Log requests to multiple loggers.

`auth_type`

Type unknown type

Default `<None>`

Authentication type to load

Table 136: Deprecated Variations

Group	Name
service_user	auth_plugin

`auth_section`

Type unknown type

Default `<None>`

Config Section from which to load plugin specific options

`auth_url`

Type unknown type

Default `<None>`

Authentication URL

`system_scope`

Type unknown type

Default `<None>`

Scope for system operations

`domain_id`

Type unknown type

Default <None>

Domain ID to scope to

domain_name

Type unknown type

Default <None>

Domain name to scope to

project_id

Type unknown type

Default <None>

Project ID to scope to

project_name

Type unknown type

Default <None>

Project name to scope to

project_domain_id

Type unknown type

Default <None>

Domain ID containing project

project_domain_name

Type unknown type

Default <None>

Domain name containing project

trust_id

Type unknown type

Default <None>

Trust ID

default_domain_id

Type unknown type

Default <None>

Optional domain ID to use with v3 and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

default_domain_name

Type unknown type

Default <None>

Optional domain name to use with v3 API and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

user_id

Type unknown type

Default `<None>`

User ID

username

Type unknown type

Default `<None>`

Username

Table 137: Deprecated Variations

Group	Name
service_user	user-name
service_user	user_name

user_domain_id

Type unknown type

Default `<None>`

Users domain id

user_domain_name

Type unknown type

Default `<None>`

Users domain name

password

Type unknown type

Default `<None>`

Users password

tenant_id

Type unknown type

Default `<None>`

Tenant ID

tenant_name

Type unknown type

Default `<None>`

Tenant Name

spice

SPICE console feature allows you to connect to a guest virtual machine. SPICE is a replacement for fairly limited VNC protocol.

Following requirements must be met in order to use SPICE:

- Virtualization driver must be libvirt
- spice.enabled set to True
- vnc.enabled set to False
- update html5proxy_base_url
- update server_proxyclient_address

enabled

Type boolean

Default `false`

Enable SPICE related features.

Related options:

- VNC must be explicitly disabled to get access to the SPICE console. Set the enabled option to False in the [vnc] section to disable the VNC console.

agent_enabled

Type boolean

Default `true`

Enable the SPICE guest agent support on the instances.

The Spice agent works with the Spice protocol to offer a better guest console experience. However, the Spice console can still be used without the Spice Agent. With the Spice agent installed the following features are enabled:

- Copy & Paste of text and images between the guest and client machine
- Automatic adjustment of resolution when the client screen changes - e.g. if you make the Spice console full screen the guest resolution will adjust to match it rather than letterboxing.
- Better mouse integration - The mouse can be captured and released without needing to click inside the console or press keys to release it. The performance of mouse movement is also improved.

html5proxy_base_url

Type URI

Default `http://127.0.0.1:6082/spice_auto.html`

Location of the SPICE HTML5 console proxy.

End user would use this URL to connect to the *nova-spicehtml5proxy*‘ service. This service will forward request to the console of an instance.

In order to use SPICE console, the service `nova-spicehtml5proxy` should be running. This service is typically launched on the controller node.

Possible values:

- Must be a valid URL of the form: `http://host:port/spice_auto.html` where host is the node running `nova-spicehtml5proxy` and the port is typically 6082. Consider not using default value as it is not well defined for any real deployment.

Related options:

- This option depends on `html5proxy_host` and `html5proxy_port` options. The access URL returned by the compute node must have the host and port where the `nova-spicehtml5proxy` service is listening.

server_listen

Type string

Default `127.0.0.1`

The address where the SPICE server running on the instances should listen.

Typically, the `nova-spicehtml5proxy` proxy client runs on the controller node and connects over the private network to this address on the compute node(s).

Possible values:

- IP address to listen on.

server_proxyclient_address

Type string

Default `127.0.0.1`

The address used by `nova-spicehtml5proxy` client to connect to instance console.

Typically, the `nova-spicehtml5proxy` proxy client runs on the controller node and connects over the private network to this address on the compute node(s).

Possible values:

- Any valid IP address on the compute node.

Related options:

- This option depends on the `server_listen` option. The proxy client must be able to access the address specified in `server_listen` using the value of this option.

keymap

Type string

Default `<None>`

A keyboard layout which is supported by the underlying hypervisor on this node.

Possible values:

- This is usually an IETF language tag (default is en-us). If you use QEMU as hypervisor, you should find the list of supported keyboard layouts at /usr/share/qemu/keymaps.

Warning: This option is deprecated for removal since 18.0.0. Its value may be silently ignored in the future.

Reason Configuring this option forces QEMU to do keymap conversions. These conversions are lossy and can result in significant issues for users of non en-US keyboards. Refer to bug #1682020 for more information.

html5proxy_host

Type host address

Default `0.0.0.0`

IP address or a hostname on which the `nova-spicehtml5proxy` service listens for incoming requests.

Related options:

- This option depends on the `html5proxy_base_url` option. The `nova-spicehtml5proxy` service must be listening on a host that is accessible from the HTML5 client.

`html5proxy_port`

Type port number

Default `6082`

Minimum Value 0

Maximum Value 65535

Port on which the `nova-spicehtml5proxy` service listens for incoming requests.

Related options:

- This option depends on the `html5proxy_base_url` option. The `nova-spicehtml5proxy` service must be listening on a port that is accessible from the HTML5 client.

upgrade_levels

upgrade_levels options are used to set version cap for RPC messages sent between different nova services.

By default all services send messages using the latest version they know about.

The compute upgrade level is an important part of rolling upgrades where old and new nova-compute services run side by side.

The other options can largely be ignored, and are only kept to help with a possible future backport issue.

`compute`

Type string

Default `<None>`

Compute RPC API version cap.

By default, we always send messages using the most recent version the client knows about.

Where you have old and new compute services running, you should set this to the lowest deployed version. This is to guarantee that all services never send messages that one of the compute nodes cant understand. Note that we only support upgrading from release N to release N+1.

Set this option to auto if you want to let the compute RPC module automatically determine what version to use based on the service versions in the deployment.

Possible values:

- By default send the latest version the client knows about
- auto: Automatically determines what version to use based on the service versions in the deployment.
- A string representing a version number in the format N.N; for example, possible values might be 1.12 or 2.0.
- An OpenStack release name, in lower case, such as mitaka or liberty.

`cert`

Type string

Default `<None>`

Cert RPC API version cap.

Possible values:

- By default send the latest version the client knows about
- A string representing a version number in the format N.N; for example, possible values might be 1.12 or 2.0.
- An OpenStack release name, in lower case, such as mitaka or liberty.

Warning: This option is deprecated for removal since 18.0.0. Its value may be silently ignored in the future.

Reason The nova-cert service was removed in 16.0.0 (Pike) so this option is no longer used.

scheduler

Type string

Default `<None>`

Scheduler RPC API version cap.

Possible values:

- By default send the latest version the client knows about
- A string representing a version number in the format N.N; for example, possible values might be 1.12 or 2.0.
- An OpenStack release name, in lower case, such as mitaka or liberty.

conductor

Type string

Default `<None>`

Conductor RPC API version cap.

Possible values:

- By default send the latest version the client knows about
- A string representing a version number in the format N.N; for example, possible values might be 1.12 or 2.0.
- An OpenStack release name, in lower case, such as mitaka or liberty.

console

Type string

Default `<None>`

Console RPC API version cap.

Possible values:

- By default send the latest version the client knows about
- A string representing a version number in the format N.N; for example, possible values might be 1.12 or 2.0.
- An OpenStack release name, in lower case, such as mitaka or liberty.

network

Type string

Default <None>

Network RPC API version cap.

Possible values:

- By default send the latest version the client knows about
- A string representing a version number in the format N.N; for example, possible values might be 1.12 or 2.0.
- An OpenStack release name, in lower case, such as mitaka or liberty.

Warning: This option is deprecated for removal since 18.0.0. Its value may be silently ignored in the future.

Reason The nova-network service was deprecated in 14.0.0 (Newton) and will be removed in an upcoming release.

baseapi

Type string

Default <None>

Base API RPC API version cap.

Possible values:

- By default send the latest version the client knows about
- A string representing a version number in the format N.N; for example, possible values might be 1.12 or 2.0.
- An OpenStack release name, in lower case, such as mitaka or liberty.

vault

root_token_id

Type string

Default <None>

root token for vault

approle_role_id

Type string

Default <None>

AppRole role_id for authentication with vault

approle_secret_id

Type string

Default <None>

AppRole secret_id for authentication with vault

kv_mountpoint

Type string

Default `secret`

Mountpoint of KV store in Vault to use, for example: secret

vault_url

Type string

Default `http://127.0.0.1:8200`

Use this endpoint to connect to Vault, for example: http://127.0.0.1:8200

ssl_ca_crt_file

Type string

Default `<None>`

Absolute path to ca cert file

use_ssl

Type boolean

Default `false`

SSL Enabled/Disabled

vendordata_dynamic_auth

Options within this group control the authentication of the vendordata subsystem of the metadata API server (and config drive) with external systems.

cafile

Type string

Default `<None>`

PEM encoded Certificate Authority to use when verifying HTTPs connections.

certfile

Type string

Default `<None>`

PEM encoded client certificate cert file

keyfile

Type string

Default `<None>`

PEM encoded client certificate key file

insecure

Type boolean

Default `false`

Verify HTTPS connections.

timeout

Type integer

Default `<None>`

Timeout value for http requests

collect_timing

Type boolean

Default `false`

Collect per-API call timing information.

split_loggers

Type boolean

Default `false`

Log requests to multiple loggers.

auth_type

Type unknown type

Default `<None>`

Authentication type to load

Table 138: Deprecated Variations

Group	Name
vendordata_dynamic_auth	auth_plugin

auth_section

Type unknown type

Default `<None>`

Config Section from which to load plugin specific options

auth_url

Type unknown type

Default `<None>`

Authentication URL

system_scope

Type unknown type

Default `<None>`

Scope for system operations

domain_id

Type unknown type

Default `<None>`

Domain ID to scope to

domain_name

Type unknown type

Default <None>

Domain name to scope to

project_id

Type unknown type

Default <None>

Project ID to scope to

project_name

Type unknown type

Default <None>

Project name to scope to

project_domain_id

Type unknown type

Default <None>

Domain ID containing project

project_domain_name

Type unknown type

Default <None>

Domain name containing project

trust_id

Type unknown type

Default <None>

Trust ID

default_domain_id

Type unknown type

Default <None>

Optional domain ID to use with v3 and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

default_domain_name

Type unknown type

Default <None>

Optional domain name to use with v3 API and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

user_id

Type unknown type

Default <None>

User ID

username

Type unknown type

Default `<None>`

Username

Table 139: Deprecated Variations

Group	Name
vendordata_dynamic_auth	user-name
vendordata_dynamic_auth	user_name

user_domain_id

Type unknown type

Default `<None>`

Users domain id

user_domain_name

Type unknown type

Default `<None>`

Users domain name

password

Type unknown type

Default `<None>`

Users password

tenant_id

Type unknown type

Default `<None>`

Tenant ID

tenant_name

Type unknown type

Default `<None>`

Tenant Name

vmware

Related options: Following options must be set in order to launch VMware-based virtual machines.

- compute_driver: Must use vmwareapi.VMwareVCDriver.
- vmware.host_username
- vmware.host_password
- vmware.cluster_name

vlan_interface

Type string

Default `vmnic0`

This option specifies the physical ethernet adapter name for VLAN networking.

Set the vlan_interface configuration option to match the ESX host interface that handles VLAN-tagged VM traffic.

Possible values:

- Any valid string representing VLAN interface name

integration_bridge

Type string

Default `<None>`

This option should be configured only when using the NSX-MH Neutron plugin. This is the name of the integration bridge on the ESXi server or host. This should not be set for any other Neutron plugin. Hence the default value is not set.

Possible values:

- Any valid string representing the name of the integration bridge

console_delay_seconds

Type integer

Default `<None>`

Minimum Value 0

Set this value if affected by an increased network latency causing repeated characters when typing in a remote console.

serial_port_service_uri

Type string

Default `<None>`

Identifies the remote system where the serial port traffic will be sent.

This option adds a virtual serial port which sends console output to a configurable service URI. At the service URI address there will be virtual serial port concentrator that will collect console logs. If this is not set, no serial ports will be added to the created VMs.

Possible values:

- Any valid URI

serial_port_proxy_uri

Type URI

Default `<None>`

Identifies a proxy service that provides network access to the serial_port_service_uri.

Possible values:

- Any valid URI (The scheme is telnet or telnets.)

Related options: This option is ignored if serial_port_service_uri is not specified. * serial_port_service_uri

serial_log_dir

Type string

Default `/opt/vmware/vspc`

Specifies the directory where the Virtual Serial Port Concentrator is storing console log files. It should match the serial_log_dir config value of VSPC.

host_ip

Type host address

Default `<None>`

Hostname or IP address for connection to VMware vCenter host.

host_port

Type port number

Default `443`

Minimum Value 0

Maximum Value 65535

Port for connection to VMware vCenter host.

host_username

Type string

Default `<None>`

Username for connection to VMware vCenter host.

host_password

Type string

Default `<None>`

Password for connection to VMware vCenter host.

ca_file

Type string

Default `<None>`

Specifies the CA bundle file to be used in verifying the vCenter server certificate.

insecure

Type boolean

Default `false`

If true, the vCenter server certificate is not verified. If false, then the default CA truststore is used for verification.

Related options: * ca_file: This option is ignored if ca_file is set.

cluster_name

Type string

Default `<None>`

Name of a VMware Cluster ComputeResource.

datastore_regex

Type string

Default `<None>`

Regular expression pattern to match the name of datastore.

The datastore_regex setting specifies the datastores to use with Compute. For example, datastore_regex=nas.* selects all the data stores that have a name starting with nas.

NOTE: If no regex is given, it just picks the datastore with the most freespace.

Possible values:

- Any matching regular expression to a datastore must be given

task_poll_interval

Type floating point

Default `0.5`

Time interval in seconds to poll remote tasks invoked on VMware VC server.

api_retry_count

Type integer

Default `10`

Minimum Value 0

Number of times VMware vCenter server API must be retried on connection failures, e.g. socket error, etc.

vnc_port

Type port number

Default `5900`

Minimum Value 0

Maximum Value 65535

This option specifies VNC starting port.

Every VM created by ESX host has an option of enabling VNC client for remote connection. Above option vnc_port helps you to set default starting port for the VNC client.

Possible values:

- Any valid port number within 5900 -(5900 + vnc_port_total)

Related options: Below options should be set to enable VNC client. * vnc.enabled = True * vnc_port_total

vnc_port_total

Type integer

Default `10000`

Minimum Value 0

Total number of VNC ports.

vnc_keymap

Type string

Default `en-us`

Keymap for VNC.

The keyboard mapping (keymap) determines which keyboard layout a VNC session should use by default.

Possible values:

- A keyboard layout which is supported by the underlying hypervisor on this node. This is usually an IETF language tag (for example en-us).

use_linked_clone

Type boolean

Default `true`

This option enables/disables the use of linked clone.

The ESX hypervisor requires a copy of the VMDK file in order to boot up a virtual machine. The compute driver must download the VMDK via HTTP from the OpenStack Image service to a datastore that is visible to the hypervisor and cache it. Subsequent virtual machines that need the VMDK use the cached version and dont have to copy the file again from the OpenStack Image service.

If set to false, even with a cached VMDK, there is still a copy operation from the cache location to the hypervisor file directory in the shared datastore. If set to true, the above copy operation is avoided as it creates copy of the virtual machine that shares virtual disks with its parent VM.

connection_pool_size

Type integer

Default `10`

Minimum Value 10

This option sets the http connection pool size

The connection pool size is the maximum number of connections from nova to vSphere. It should only be increased if there are warnings indicating that the connection pool is full, otherwise, the default should suffice.

pbm_enabled

Type boolean

Default `false`

This option enables or disables storage policy based placement of instances.

Related options:

- pbm_default_policy

pbm_wsdl_location

Type string

Default `<None>`

This option specifies the PBM service WSDL file location URL.

Setting this will disable storage policy based placement of instances.

Possible values:

- Any valid file path e.g file:///opt/SDK/spbm/wsdl/pbmService.wsdl

pbm_default_policy

Type string

Default `<None>`

This option specifies the default policy to be used.

If pbm_enabled is set and there is no defined storage policy for the specific request, then this policy will be used.

Possible values:

- Any valid storage policy such as VSAN default storage policy

Related options:

- pbm_enabled

`maximum_objects`

Type integer

Default `100`

Minimum Value 0

This option specifies the limit on the maximum number of objects to return in a single result.

A positive value will cause the operation to suspend the retrieval when the count of objects reaches the specified limit. The server may still limit the count to something less than the configured value. Any remaining objects may be retrieved with additional requests.

`cache_prefix`

Type string

Default `<None>`

This option adds a prefix to the folder where cached images are stored

This is not the full path - just a folder prefix. This should only be used when a datastore cache is shared between compute nodes.

Note: This should only be used when the compute nodes are running on same host or they have a shared file system.

Possible values:

- Any string representing the cache prefix to the folder

vnc

Virtual Network Computer (VNC) can be used to provide remote desktop console access to instances for tenants and/or administrators.

`enabled`

Type boolean

Default `true`

Enable VNC related features.

Guests will get created with graphical devices to support this. Clients (for example Horizon) can then establish a VNC connection to the guest.

Table 140: Deprecated Variations

Group	Name
DEFAULT	vnc_enabled

keymap

Type string

Default <None>

Keymap for VNC.

The keyboard mapping (keymap) determines which keyboard layout a VNC session should use by default.

Possible values:

- A keyboard layout which is supported by the underlying hypervisor on this node. This is usually an IETF language tag (for example en-us). If you use QEMU as hypervisor, you should find the list of supported keyboard layouts at /usr/share/qemu/keymaps.

Table 141: Deprecated Variations

Group	Name
DEFAULT	vnc_keymap

Warning: This option is deprecated for removal since 18.0.0. Its value may be silently ignored in the future.

Reason Configuring this option forces QEMU to do keymap conversions. These conversions are lossy and can result in significant issues for users of non en-US keyboards. You should instead use a VNC client that supports Extended Key Event messages, such as noVNC 1.0.0. Refer to bug #1682020 for more information.

server_listen

Type host address

Default 127.0.0.1

The IP address or hostname on which an instance should listen to for incoming VNC connection requests on this node.

Table 142: Deprecated Variations

Group	Name
DEFAULT	vncserver_listen
vnc	vncserver_listen

server_proxyclient_address

Type host address

Default 127.0.0.1

Private, internal IP address or hostname of VNC console proxy.

The VNC proxy is an OpenStack component that enables compute service users to access their instances through VNC clients.

This option sets the private address to which proxy clients, such as nova-novncproxy, should connect to.

Table 143: Deprecated Variations

Group	Name
DEFAULT	vncserver_proxyclient_address
vnc	vncserver_proxyclient_address

novncproxy_base_url

Type URI

Default `http://127.0.0.1:6080/vnc_auto.html`

Public address of noVNC VNC console proxy.

The VNC proxy is an OpenStack component that enables compute service users to access their instances through VNC clients. noVNC provides VNC support through a websocket-based client.

This option sets the public base URL to which client systems will connect. noVNC clients can use this address to connect to the noVNC instance and, by extension, the VNC sessions.

If using noVNC >= 1.0.0, you should use `vnc_lite.html` instead of `vnc_auto.html`.

Related options:

- novncproxy_host
- novncproxy_port

Table 144: Deprecated Variations

Group	Name
DEFAULT	novncproxy_base_url

xvpvncproxy_host

Type host address

Default `0.0.0.0`

IP address or hostname that the XVP VNC console proxy should bind to.

The VNC proxy is an OpenStack component that enables compute service users to access their instances through VNC clients. Xen provides the Xenserver VNC Proxy, or XVP, as an alternative to the websocket-based noVNC proxy used by Libvirt. In contrast to noVNC, XVP clients are Java-based.

This option sets the private address to which the XVP VNC console proxy service should bind to.

Related options:

- xvpvncproxy_port
- xvpvncproxy_base_url

Table 145: Deprecated Variations

Group	Name
DEFAULT	xvpvncproxy_host

Warning: This option is deprecated for removal since 19.0.0. Its value may be silently ignored in the future.

Reason The `nova-xvpvnxproxy` service is deprecated and will be removed in an upcoming release.

xvpvncproxy_port

Type port number

Default `6081`

Minimum Value 0

Maximum Value 65535

Port that the XVP VNC console proxy should bind to.

The VNC proxy is an OpenStack component that enables compute service users to access their instances through VNC clients. Xen provides the Xenserver VNC Proxy, or XVP, as an alternative to the websocket-based noVNC proxy used by Libvirt. In contrast to noVNC, XVP clients are Java-based.

This option sets the private port to which the XVP VNC console proxy service should bind to.

Related options:

- xvpvncproxy_host
- xvpvncproxy_base_url

Table 146: Deprecated Variations

Group	Name
DEFAULT	xvpvncproxy_port

Warning: This option is deprecated for removal since 19.0.0. Its value may be silently ignored in the future.

Reason The `nova-xvpvnxproxy` service is deprecated and will be removed in an upcoming release.

xvpvncproxy_base_url

Type URI

Default `http://127.0.0.1:6081/console`

Public URL address of XVP VNC console proxy.

The VNC proxy is an OpenStack component that enables compute service users to access their instances through VNC clients. Xen provides the Xenserver VNC Proxy, or XVP, as an alternative to the websocket-based noVNC proxy used by Libvirt. In contrast to noVNC, XVP clients are Java-based.

This option sets the public base URL to which client systems will connect. XVP clients can use this address to connect to the XVP instance and, by extension, the VNC sessions.

Related options:

- xvpvncproxy_host
- xvpvncproxy_port

Table 147: Deprecated Variations

Group	Name
DEFAULT	xvpvncproxy_base_url

Warning: This option is deprecated for removal since 19.0.0. Its value may be silently ignored in the future.

Reason The `nova-xvpvnxproxy` service is deprecated and will be removed in an upcoming release.

`novncproxy_host`

Type string

Default `0.0.0.0`

IP address that the noVNC console proxy should bind to.

The VNC proxy is an OpenStack component that enables compute service users to access their instances through VNC clients. noVNC provides VNC support through a websocket-based client.

This option sets the private address to which the noVNC console proxy service should bind to.

Related options:

- novncproxy_port
- novncproxy_base_url

Table 148: Deprecated Variations

Group	Name
DEFAULT	novncproxy_host

`novncproxy_port`

Type port number

Default `6080`

Minimum Value 0

Maximum Value 65535

Port that the noVNC console proxy should bind to.

The VNC proxy is an OpenStack component that enables compute service users to access their instances through VNC clients. noVNC provides VNC support through a websocket-based client.

This option sets the private port to which the noVNC console proxy service should bind to.

Related options:

- novncproxy_host
- novncproxy_base_url

Table 149: Deprecated Variations

Group	Name
DEFAULT	novncproxy_port

auth_schemes

Type list

Default none

The authentication schemes to use with the compute node.

Control what RFB authentication schemes are permitted for connections between the proxy and the compute host. If multiple schemes are enabled, the first matching scheme will be used, thus the strongest schemes should be listed first.

Related options:

- [vnc]vencrypt_client_key, [vnc]vencrypt_client_cert: must also be set

vencrypt_client_key

Type string

Default <None>

The path to the client certificate PEM file (for x509)

The fully qualified path to a PEM file containing the private key which the VNC proxy server presents to the compute node during VNC authentication.

Related options:

- vnc.auth_schemes: must include vencrypt
- vnc.vencrypt_client_cert: must also be set

vencrypt_client_cert

Type string

Default <None>

The path to the client key file (for x509)

The fully qualified path to a PEM file containing the x509 certificate which the VNC proxy server presents to the compute node during VNC authentication.

Realted options:

- vnc.auth_schemes: must include vencrypt
- vnc.vencrypt_client_key: must also be set

vencrypt_ca_certs

Type string

Default <None>

The path to the CA certificate PEM file

The fully qualified path to a PEM file containing one or more x509 certificates for the certificate authorities used by the compute node VNC server.

Related options:

- vnc.auth_schemes: must include vencrypt

workarounds

A collection of workarounds used to mitigate bugs or issues found in system tools (e.g. Libvirt or QEMU) or Nova itself under certain conditions. These should only be enabled in exceptional circumstances. All options are linked against bug IDs, where more information on the issue can be found.

`disable_rootwrap`

Type boolean

Default `false`

Use sudo instead of rootwrap.

Allow fallback to sudo for performance reasons.

For more information, refer to the bug report:

https://bugs.launchpad.net/nova/+bug/1415106

Possible values:

- True: Use sudo instead of rootwrap
- False: Use rootwrap as usual

Interdependencies to other options:

- Any options that affect rootwrap will be ignored.

`disable_libvirt_livesnapshot`

Type boolean

Default `false`

Disable live snapshots when using the libvirt driver.

Live snapshots allow the snapshot of the disk to happen without an interruption to the guest, using coordination with a guest agent to quiesce the filesystem.

When using libvirt 1.2.2 live snapshots fail intermittently under load (likely related to concurrent libvirt/qemu operations). This config option provides a mechanism to disable live snapshot, in favor of cold snapshot, while this is resolved. Cold snapshot causes an instance outage while the guest is going through the snapshotting process.

For more information, refer to the bug report:

https://bugs.launchpad.net/nova/+bug/1334398

Possible values:

- True: Live snapshot is disabled when using libvirt
- False: Live snapshots are always used when snapshotting (as long as there is a new enough libvirt and the backend storage supports it)

Warning: This option is deprecated for removal since 19.0.0. Its value may be silently ignored in the future.

Reason This option was added to work around issues with libvirt 1.2.2. We no longer support this version of libvirt, which means this workaround is no longer necessary. It will be removed in a future release.

handle_virt_lifecycle_events

Type boolean

Default `true`

Enable handling of events emitted from compute drivers.

Many compute drivers emit lifecycle events, which are events that occur when, for example, an instance is starting or stopping. If the instance is going through task state changes due to an API operation, like resize, the events are ignored.

This is an advanced feature which allows the hypervisor to signal to the compute service that an unexpected state change has occurred in an instance and that the instance can be shutdown automatically. Unfortunately, this can race in some conditions, for example in reboot operations or when the compute service or when host is rebooted (planned or due to an outage). If such races are common, then it is advisable to disable this feature.

Care should be taken when this feature is disabled and sync_power_state_interval is set to a negative value. In this case, any instances that get out of sync between the hypervisor and the Nova database will have to be synchronized manually.

For more information, refer to the bug report: https://bugs.launchpad.net/bugs/1444630

Interdependencies to other options:

- If `sync_power_state_interval` is negative and this feature is disabled, then instances that get out of sync between the hypervisor and the Nova database will have to be synchronized manually.

disable_group_policy_check_upcall

Type boolean

Default `false`

Disable the server group policy check upcall in compute.

In order to detect races with server group affinity policy, the compute service attempts to validate that the policy was not violated by the scheduler. It does this by making an upcall to the API database to list the instances in the server group for one that it is booting, which violates our api/cell isolation goals. Eventually this will be solved by proper affinity guarantees in the scheduler and placement service, but until then, this late check is needed to ensure proper affinity policy.

Operators that desire api/cell isolation over this check should enable this flag, which will avoid making that upcall from compute.

Related options:

- [filter_scheduler]/track_instance_changes also relies on upcalls from the compute service to the scheduler service.

enable_numa_live_migration

Type boolean

Default `false`

Enable live migration of instances with NUMA topologies.

Live migration of instances with NUMA topologies when using the libvirt driver is only supported in deployments that have been fully upgraded to Train. In previous versions, or in mixed Stein/Train deployments with a rolling upgrade in progress, live migration of instances with NUMA topologies is disabled by default when using the libvirt driver. This includes live migration of instances with CPU pinning or hugepages. CPU pinning and huge page information for such instances is not currently re-calculated, as noted in bug #1289064. This means that if instances were already present on the destination host, the migrated instance could be placed on the same dedicated cores as these instances or use hugepages allocated for another instance. Alternately, if the

host platforms were not homogeneous, the instance could be assigned to non-existent cores or be inadvertently split across host NUMA nodes.

Despite these known issues, there may be cases where live migration is necessary. By enabling this option, operators that are aware of the issues and are willing to manually work around them can enable live migration support for these instances.

Related options:

- `compute_driver`: Only the libvirt driver is affected.

> **Warning:** This option is deprecated for removal since 20.0.0. Its value may be silently ignored in the future.
>
> **Reason** This option was added to mitigate known issues when live migrating instances with a NUMA topology with the libvirt driver. Those issues are resolved in Train. Clouds using the libvirt driver and fully upgraded to Train support NUMA-aware live migration. This option will be removed in a future release.

`ensure_libvirt_rbd_instance_dir_cleanup`

Type boolean

Default `false`

Ensure the instance directory is removed during clean up when using rbd.

When enabled this workaround will ensure that the instance directory is always removed during cleanup on hosts using `[libvirt]/images_type=rbd`. This avoids the following bugs with evacuation and revert resize clean up that lead to the instance directory remaining on the host:

https://bugs.launchpad.net/nova/+bug/1414895

https://bugs.launchpad.net/nova/+bug/1761062

Both of these bugs can then result in `DestinationDiskExists` errors being raised if the instances ever attempt to return to the host.

> **Warning:** Operators will need to ensure that the instance directory itself, specified by `[DEFAULT]/instances_path`, is not shared between computes before enabling this workaround otherwise the console.log, kernels, ramdisks and any additional files being used by the running instance will be lost.

Related options:

- `compute_driver` (libvirt)
- `[libvirt]/images_type` (rbd)
- `instances_path`

`disable_fallback_pcpu_query`

Type boolean

Default `false`

Disable fallback request for VCPU allocations when using pinned instances.

Starting in Train, compute nodes using the libvirt virt driver can report `PCPU` inventory and will use this for pinned instances. The scheduler will automatically translate requests using the legacy CPU pinning-related

flavor extra specs, `hw:cpu_policy` and `hw:cpu_thread_policy`, their image metadata property equivalents, and the emulator threads pinning flavor extra spec, `hw:emulator_threads_policy`, to new placement requests. However, compute nodes require additional configuration in order to report `PCPU` inventory and this configuration may not be present immediately after an upgrade. To ensure pinned instances can be created without this additional configuration, the scheduler will make a second request to placement for old-style `VCPU`-based allocations and fallback to these allocation candidates if necessary. This has a slight performance impact and is not necessary on new or upgraded deployments where the new configuration has been set on all hosts. By setting this option, the second lookup is disabled and the scheduler will only request `PCPU`-based allocations.

Warning: This option is deprecated for removal since 20.0.0. Its value may be silently ignored in the future.

reserve_disk_resource_for_image_cache

Type boolean

Default `false`

If it is set to True then the libvirt driver will reserve DISK_GB resource for the images stored in the image cache. If the *DEFAULT.instances_path* is on different disk partition than the image cache directory then the driver will not reserve resource for the cache.

Such disk reservation is done by a periodic task in the resource tracker that runs every *update_resources_interval* seconds. So the reservation is not updated immediately when an image is cached.

Related options:

- *DEFAULT.instances_path*
- *image_cache_subdirectory_name*
- *update_resources_interval*

never_download_image_if_on_rbd

Type boolean

Default `false`

When booting from an image on a ceph-backed compute node, if the image does not already reside on the ceph cluster (as would be the case if glance is also using the same cluster), nova will download the image from glance and upload it to ceph itself. If using multiple ceph clusters, this may cause nova to unintentionally duplicate the image in a non-COW-able way in the local ceph deployment, wasting space.

For more information, refer to the bug report:

https://bugs.launchpad.net/nova/+bug/1858877

Enabling this option will cause nova to *refuse* to boot an instance if it would require downloading the image from glance and uploading it to ceph itself.

Related options:

- `compute_driver` (libvirt)
- `[libvirt]/images_type` (rbd)

wsgi

Options under this group are used to configure WSGI (Web Server Gateway Interface). WSGI is used to serve API requests.

api_paste_config

Type string

Default `api-paste.ini`

This option represents a file name for the paste.deploy config for nova-api.

Possible values:

- A string representing file name for the paste.deploy config.

Table 150: Deprecated Variations

Group	Name
DEFAULT	api_paste_config

wsgi_log_format

Type string

Default `%(client_ip)s "%(request_line)s" status: %(status_code)s len: %(body_length)s time: %(wall_seconds).7f`

It represents a python format string that is used as the template to generate log lines. The following values can be formatted into it: client_ip, date_time, request_line, status_code, body_length, wall_seconds.

This option is used for building custom request loglines when running nova-api under eventlet. If used under uwsgi or apache, this option has no effect.

Possible values:

- %(client_ip)s %(request_line)s status: %(status_code)s len: %(body_length)s time: %(wall_seconds).7f (default)
- Any formatted string formed by specific values.

Table 151: Deprecated Variations

Group	Name
DEFAULT	wsgi_log_format

Warning: This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason This option only works when running nova-api under eventlet, and encodes very eventlet specific pieces of information. Starting in Pike the preferred model for running nova-api is under uwsgi or apache mod_wsgi.

secure_proxy_ssl_header

Type string

Default `<None>`

This option specifies the HTTP header used to determine the protocol scheme for the original request, even if it was removed by a SSL terminating proxy.

Possible values:

- None (default) - the request scheme is not influenced by any HTTP headers
- Valid HTTP header, like `HTTP_X_FORWARDED_PROTO`

WARNING: Do not set this unless you know what you are doing.

Make sure ALL of the following are true before setting this (assuming the values from the example above):

- Your API is behind a proxy.
- Your proxy strips the X-Forwarded-Proto header from all incoming requests. In other words, if end users include that header in their requests, the proxy will discard it.
- Your proxy sets the X-Forwarded-Proto header and sends it to API, but only for requests that originally come in via HTTPS.

If any of those are not true, you should keep this setting set to None.

Table 152: Deprecated Variations

Group	Name
DEFAULT	secure_proxy_ssl_header

ssl_ca_file

Type string

Default `<None>`

This option allows setting path to the CA certificate file that should be used to verify connecting clients.

Possible values:

- String representing path to the CA certificate file.

Related options:

- enabled_ssl_apis

Table 153: Deprecated Variations

Group	Name
DEFAULT	ssl_ca_file

ssl_cert_file

Type string

Default `<None>`

This option allows setting path to the SSL certificate of API server.

Possible values:

- String representing path to the SSL certificate.

Related options:

- enabled_ssl_apis

Table 154: Deprecated Variations

Group	Name
DEFAULT	ssl_cert_file

ssl_key_file

Type string

Default <None>

This option specifies the path to the file where SSL private key of API server is stored when SSL is in effect.

Possible values:

- String representing path to the SSL private key.

Related options:

- enabled_ssl_apis

Table 155: Deprecated Variations

Group	Name
DEFAULT	ssl_key_file

tcp_keepidle

Type integer

Default 600

Minimum Value 0

This option sets the value of TCP_KEEPIDLE in seconds for each server socket. It specifies the duration of time to keep connection active. TCP generates a KEEPALIVE transmission for an application that requests to keep connection active. Not supported on OS X.

Related options:

- keep_alive

Table 156: Deprecated Variations

Group	Name
DEFAULT	tcp_keepidle

default_pool_size

Type integer

Default 1000

Minimum Value 0

This option specifies the size of the pool of greenthreads used by wsgi. It is possible to limit the number of concurrent connections using this option.

Table 157: Deprecated Variations

Group	Name
DEFAULT	wsgi_default_pool_size

max_header_line

Type integer

Default `16384`

Minimum Value 0

This option specifies the maximum line size of message headers to be accepted. max_header_line may need to be increased when using large tokens (typically those generated by the Keystone v3 API with big service catalogs).

Since TCP is a stream based protocol, in order to reuse a connection, the HTTP has to have a way to indicate the end of the previous response and beginning of the next. Hence, in a keep_alive case, all messages must have a self-defined message length.

Table 158: Deprecated Variations

Group	Name
DEFAULT	max_header_line

keep_alive

Type boolean

Default `true`

This option allows using the same TCP connection to send and receive multiple HTTP requests/responses, as opposed to opening a new one for every single request/response pair. HTTP keep-alive indicates HTTP connection reuse.

Possible values:

- True : reuse HTTP connection.
- False : closes the client socket connection explicitly.

Related options:

- tcp_keepidle

Table 159: Deprecated Variations

Group	Name
DEFAULT	wsgi_keep_alive

client_socket_timeout

Type integer

Default `900`

Minimum Value 0

This option specifies the timeout for client connections socket operations. If an incoming connection is idle for this number of seconds it will be closed. It indicates timeout on individual read/writes on the socket connection. To wait forever set to 0.

Table 160: Deprecated Variations

Group	Name
DEFAULT	client_socket_timeout

xenserver

Warning: The xenapi driver is deprecated and may be removed in a future release. The driver is not tested by the OpenStack project nor does it have clear maintainer(s) and thus its quality can not be ensured. If you are using the driver in production please let us know in freenode IRC and/or the openstack-discuss mailing list.

XenServer options are used when the compute_driver is set to use XenServer (compute_driver=xenapi.XenAPIDriver).

Must specify connection_url, connection_password and ovs_integration_bridge to use compute_driver=xenapi.XenAPIDriver.

`agent_timeout`

Type integer

Default `30`

Minimum Value 0

Number of seconds to wait for agents reply to a request.

Nova configures/performs certain administrative actions on a server with the help of an agent thats installed on the server. The communication between Nova and the agent is achieved via sharing messages, called records, over xenstore, a shared storage across all the domains on a Xenserver host. Operations performed by the agent on behalf of nova are: version, key_init, password,resetnetwork,inject_file, and agentupdate.

To perform one of the above operations, the xapi agent plugin writes the command and its associated parameters to a certain location known to the domain and awaits response. On being notified of the message, the agent performs appropriate actions on the server and writes the result back to xenstore. This result is then read by the xapi agent plugin to determine the success/failure of the operation.

This config option determines how long the xapi agent plugin shall wait to read the response off of xenstore for a given request/command. If the agent on the instance fails to write the result in this time period, the operation is considered to have timed out.

Related options:

- `agent_version_timeout`
- `agent_resetnetwork_timeout`

`agent_version_timeout`

Type integer

Default `300`

Minimum Value 0

Number of seconds to wait for agentt reply to version request.

This indicates the amount of time xapi agent plugin waits for the agent to respond to the version request specifically. The generic timeout for agent communication `agent_timeout` is ignored in this case.

During the build process the version request is used to determine if the agent is available/operational to perform other requests such as resetnetwork, password, key_init and inject_file. If the version call fails, the other configuration is skipped. So, this configuration option can also be interpreted as time in which agent is expected to be fully operational.

`agent_resetnetwork_timeout`

Type integer

Default `60`

Minimum Value 0

Number of seconds to wait for agents reply to resetnetwork request.

This indicates the amount of time xapi agent plugin waits for the agent to respond to the resetnetwork request specifically. The generic timeout for agent communication `agent_timeout` is ignored in this case.

`agent_path`

Type string

Default `usr/sbin/xe-update-networking`

Path to locate guest agent on the server.

Specifies the path in which the XenAPI guest agent should be located. If the agent is present, network configuration is not injected into the image.

Related options:

For this option to have an effect: * `flat_injected` should be set to `True` * `compute_driver` should be set to `xenapi.XenAPIDriver`

`disable_agent`

Type boolean

Default `false`

Disables the use of XenAPI agent.

This configuration option suggests whether the use of agent should be enabled or not regardless of what image properties are present. Image properties have an effect only when this is set to `True`. Read description of config option `use_agent_default` for more information.

Related options:

- `use_agent_default`

`use_agent_default`

Type boolean

Default `false`

Whether or not to use the agent by default when its usage is enabled but not indicated by the image.

The use of XenAPI agent can be disabled altogether using the configuration option `disable_agent`. However, if it is not disabled, the use of an agent can still be controlled by the image in use through one of its properties, `xenapi_use_agent`. If this property is either not present or specified incorrectly on the image, the use of agent is determined by this configuration option.

Note that if this configuration is set to `True` when the agent is not present, the boot times will increase significantly.

Related options:

- `disable_agent`

`login_timeout`

Type integer

Default `10`

Minimum Value 0

Timeout in seconds for XenAPI login.

connection_concurrent

Type integer

Default `5`

Minimum Value 1

Maximum number of concurrent XenAPI connections.

In nova, multiple XenAPI requests can happen at a time. Configuring this option will parallelize access to the XenAPI session, which allows you to make concurrent XenAPI connections.

cache_images

Type string

Default `all`

Valid Values all, some, none

Cache glance images locally.

The value for this option must be chosen from the choices listed here. Configuring a value other than these will default to all.

Note: There is nothing that deletes these images.

Possible values

all Will cache all images

some Will only cache images that have the image_property `cache_in_nova=True`

none Turns off caching entirely

image_compression_level

Type integer

Default `<None>`

Minimum Value 1

Maximum Value 9

Compression level for images.

By setting this option we can configure the gzip compression level. This option sets GZIP environment variable before spawning tar -cz to force the compression level. It defaults to none, which means the GZIP environment variable is not set and the default (usually -6) is used.

Possible values:

- Range is 1-9, e.g., 9 for gzip -9, 9 being most compressed but most CPU intensive on dom0.
- Any values out of this range will default to None.

default_os_type

Type string

Default `linux`

Default OS type used when uploading an image to glance

block_device_creation_timeout

Type integer

Default `10`

Minimum Value 1

Time in secs to wait for a block device to be created

max_kernel_ramdisk_size

Type integer

Default `16777216`

Maximum size in bytes of kernel or ramdisk images.

Specifying the maximum size of kernel or ramdisk will avoid copying large files to dom0 and fill up /boot/guest.

sr_matching_filter

Type string

Default `default-sr:true`

Filter for finding the SR to be used to install guest instances on.

Possible values:

- To use the Local Storage in default XenServer/XCP installations set this flag to other-config:i18n-key=local-storage.
- To select an SR with a different matching criteria, you could set it to other-config:my_favorite_sr=true.
- To fall back on the Default SR, as displayed by XenCenter, set this flag to: default-sr:true.

sparse_copy

Type boolean

Default `true`

Whether to use sparse_copy for copying data on a resize down. (False will use standard dd). This speeds up resizes down considerably since large runs of zeros wont have to be rsynced.

num_vbd_unplug_retries

Type integer

Default `10`

Minimum Value 0

Maximum number of retries to unplug VBD. If set to 0, should try once, no retries.

ipxe_network_name

Type string

Default `<None>`

Name of network to use for booting iPXE ISOs.

An iPXE ISO is a specially crafted ISO which supports iPXE booting. This feature gives a means to roll your own image.

By default this option is not set. Enable this option to boot an iPXE ISO.

Related Options:

- *ipxe_boot_menu_url*
- *ipxe_mkisofs_cmd*

ipxe_boot_menu_url

Type string

Default `<None>`

URL to the iPXE boot menu.

An iPXE ISO is a specially crafted ISO which supports iPXE booting. This feature gives a means to roll your own image.

By default this option is not set. Enable this option to boot an iPXE ISO.

Related Options:

- *ipxe_network_name*
- *ipxe_mkisofs_cmd*

ipxe_mkisofs_cmd

Type string

Default `mkisofs`

Name and optionally path of the tool used for ISO image creation.

An iPXE ISO is a specially crafted ISO which supports iPXE booting. This feature gives a means to roll your own image.

Note: By default *mkisofs* is not present in the Dom0, so the package can either be manually added to Dom0 or include the *mkisofs* binary in the image itself.

Related Options:

- *ipxe_network_name*
- *ipxe_boot_menu_url*

connection_url

Type string

Default `<None>`

URL for connection to XenServer/Xen Cloud Platform. A special value of unix://local can be used to connect to the local unix socket.

Possible values:

- Any string that represents a URL. The connection_url is generally the management network IP address of the XenServer.
- This option must be set if you chose the XenServer driver.

connection_username

Type string

Default `root`

Username for connection to XenServer/Xen Cloud Platform

connection_password

Type string

Default `<None>`

Password for connection to XenServer/Xen Cloud Platform

vhd_coalesce_poll_interval

Type floating point

Default `5.0`

Minimum Value 0

The interval used for polling of coalescing vhds.

This is the interval after which the task of coalesce VHD is performed, until it reaches the max attempts that is set by vhd_coalesce_max_attempts.

Related options:

- *vhd_coalesce_max_attempts*

check_host

Type boolean

Default `true`

Ensure compute service is running on host XenAPI connects to. This option must be set to false if the independent_compute option is set to true.

Possible values:

- Setting this option to true will make sure that compute service is running on the same host that is specified by connection_url.
- Setting this option to false, doesnt perform the check.

Related options:

- *independent_compute*

vhd_coalesce_max_attempts

Type integer

Default `20`

Minimum Value 0

Max number of times to poll for VHD to coalesce.

This option determines the maximum number of attempts that can be made for coalescing the VHD before giving up.

Related opitons:

- *vhd_coalesce_poll_interval*

sr_base_path

Type string

Default `/var/run/sr-mount`

Base path to the storage repository on the XenServer host.

target_host

Type host address

Default `<None>`

The iSCSI Target Host.

This option represents the hostname or ip of the iSCSI Target. If the target host is not present in the connection information from the volume provider then the value from this option is taken.

Possible values:

- Any string that represents hostname/ip of Target.

`target_port`

Type port number

Default `3260`

Minimum Value 0

Maximum Value 65535

The iSCSI Target Port.

This option represents the port of the iSCSI Target. If the target port is not present in the connection information from the volume provider then the value from this option is taken.

`independent_compute`

Type boolean

Default `false`

Used to prevent attempts to attach VBDs locally, so Nova can be run in a VM on a different host.

Related options:

- `CONF.flat_injected` (Must be False)
- `CONF.xenserver.check_host` (Must be False)
- `CONF.default_ephemeral_format` (Must be unset or ext3)
- Joining host aggregates (will error if attempted)
- Swap disks for Windows VMs (will error if attempted)
- Nova-based auto_configure_disk (will error if attempted)

`running_timeout`

Type integer

Default `60`

Minimum Value 0

Wait time for instances to go to running state.

Provide an integer value representing time in seconds to set the wait time for an instance to go to running state.

When a request to create an instance is received by nova-api and communicated to nova-compute, the creation of the instance occurs through interaction with Xen via XenAPI in the compute node. Once the node on which the instance(s) are to be launched is decided by nova-schedule and the launch is triggered, a certain amount of wait time is involved until the instance(s) can become available and running. This wait time is defined by running_timeout. If the instances do not go to running state within this specified wait time, the launch expires and the instance(s) are set to error state.

`image_upload_handler`

Type string

Default `''`

Dom0 plugin driver used to handle image uploads.

Provide a string value representing a plugin driver required to handle the image uploading to GlanceStore.

Images, and snapshots from XenServer need to be uploaded to the data store for use. image_upload_handler takes in a value for the Dom0 plugin driver. This driver is then called to uplaod images to the GlanceStore.

> **Warning:** This option is deprecated for removal since 18.0.0. Its value may be silently ignored in the future.
>
> **Reason** Instead of setting the class path here, we will use short names to represent image handlers. The download and upload handlers must also be matching. So another new option image_handler will be used to set the short name for a specific image handler for both image download and upload.

`image_handler`

Type string

Default `direct_vhd`

Valid Values direct_vhd, vdi_local_dev, vdi_remote_stream

The plugin used to handle image uploads and downloads.

Provide a short name representing an image driver required to handle the image between compute host and glance.

Possible values

direct_vhd This plugin directly processes the VHD files in XenServer SR(Storage Repository). So this plugin only works when the hosts SR type is file system based e.g. ext, nfs.

vdi_local_dev This plugin implements an image handler which attaches the instances VDI as a local disk to the VM where the OpenStack Compute service runs. It uploads the raw disk to glance when creating image; when booting an instance from a glance image, it downloads the image and streams it into the disk which is attached to the compute VM.

vdi_remote_stream This plugin implements an image handler which works as a proxy between glance and XenServer. The VHD streams to XenServer via a remote import API supplied by XAPI for image download; and for image upload, the VHD streams from XenServer via a remote export API supplied by XAPI. This plugin works for all SR types supported by XenServer.

`introduce_vdi_retry_wait`

Type integer

Default `20`

Minimum Value 0

Number of seconds to wait for SR to settle if the VDI does not exist when first introduced.

Some SRs, particularly iSCSI connections are slow to see the VDIs right after they got introduced. Setting this option to a time interval will make the SR to wait for that time period before raising VDI not found exception.

`ovs_integration_bridge`

Type string

Default `<None>`

The name of the integration Bridge that is used with xenapi when connecting with Open vSwitch.

Note: The value of this config option is dependent on the environment, therefore this configuration value must be set accordingly if you are using XenAPI.

Possible values:

- Any string that represents a bridge name.

`use_join_force`

Type boolean

Default `true`

When adding new host to a pool, this will append a –force flag to the command, forcing hosts to join a pool, even if they have different CPUs.

Since XenServer version 5.6 it is possible to create a pool of hosts that have different CPU capabilities. To accommodate CPU differences, XenServer limited features it uses to determine CPU compatibility to only the ones that are exposed by CPU and support for CPU masking was added. Despite this effort to level differences between CPUs, it is still possible that adding new host will fail, thus option to force join was introduced.

`console_public_hostname`

Type string

Default `<current_hostname>`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

Publicly visible name for this console host.

Possible values:

- Current hostname (default) or any string representing hostname.

Table 161: Deprecated Variations

Group	Name
DEFAULT	console_public_hostname

xvp

Configuration options for XVP.

xvp (Xen VNC Proxy) is a proxy server providing password-protected VNC-based access to the consoles of virtual machines hosted on Citrix XenServer.

`console_xvp_conf_template`

Type string

Default `$pybasedir/nova/console/xvp.conf.template`

XVP conf template

Table 162: Deprecated Variations

Group	Name
DEFAULT	console_xvp_conf_template

console_xvp_conf

Type string

Default `/etc/xvp.conf`

Generated XVP conf file

Table 163: Deprecated Variations

Group	Name
DEFAULT	console_xvp_conf

console_xvp_pid

Type string

Default `/var/run/xvp.pid`

XVP master process pid file

Table 164: Deprecated Variations

Group	Name
DEFAULT	console_xvp_pid

console_xvp_log

Type string

Default `/var/log/xvp.log`

XVP log file

Table 165: Deprecated Variations

Group	Name
DEFAULT	console_xvp_log

console_xvp_multiplex_port

Type port number

Default `5900`

Minimum Value 0

Maximum Value 65535

Port for XVP to multiplex VNC connections on

Table 166: Deprecated Variations

Group	Name
DEFAULT	console_xvp_multiplex_port

zvm

zvm options allows cloud administrator to configure related z/VM hypervisor driver to be used within an OpenStack deployment.

zVM options are used when the compute_driver is set to use zVM (compute_driver=zvm.ZVMDriver)

cloud_connector_url

Type URI

Default `http://zvm.example.org:8080/`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

URL to be used to communicate with z/VM Cloud Connector.

ca_file

Type string

Default `<None>`

CA certificate file to be verified in httpd server with TLS enabled

A string, it must be a path to a CA bundle to use.

image_tmp_path

Type string

Default `$state_path/images`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

The path at which images will be stored (snapshot, deploy, etc).

Images used for deploy and images captured via snapshot need to be stored on the local disk of the compute host. This configuration identifies the directory location.

Possible values: A file system path on the host running the compute service.

reachable_timeout

Type integer

Default `300`

Timeout (seconds) to wait for an instance to start.

The z/VM driver relies on communication between the instance and cloud connector. After an instance is created, it must have enough time to wait for all the network info to be written into the user directory. The driver will keep rechecking network status to the instance with the timeout value, If setting network failed, it will notify the user that starting the instance failed and put the instance in ERROR state. The underlying z/VM guest will then be deleted.

Possible Values: Any positive integer. Recommended to be at least 300 seconds (5 minutes), but it will vary depending on instance and system load. A value of 0 is used for debug. In this case the underlying z/VM guest will not be deleted when the instance is marked in ERROR state.

Nova Policies

The following is an overview of all available policies in Nova.

nova

context_is_admin

Default `role:admin`

Decides what is required for the is_admin:True check to succeed.

admin_or_owner

Default `is_admin:True or project_id:%(project_id)s`

Default rule for most non-Admin APIs.

admin_api

Default `is_admin:True`

Default rule for most Admin APIs.

os_compute_api:os-admin-actions:reset_state

Default `rule:admin_api`

Operations

- **POST** `/servers/{server_id}/action (os-resetState)`

Reset the state of a given server

os_compute_api:os-admin-actions:inject_network_info

Default `rule:admin_api`

Operations

- **POST** `/servers/{server_id}/action (injectNetworkInfo)`

Inject network information into the server

os_compute_api:os-admin-actions:reset_network

Default `rule:admin_api`

Operations

- **POST** `/servers/{server_id}/action (resetNetwork)`

Reset networking on a server

os_compute_api:os-admin-password

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (changePassword)`

Change the administrative password for a server

os_compute_api:os-agents

Default `rule:admin_api`

Operations

- **GET** `/os-agents`
- **POST** `/os-agents`
- **PUT** `/os-agents/{agent_build_id}`

- **DELETE** `/os-agents/{agent_build_id}`

Create, list, update, and delete guest agent builds

This is XenAPI driver specific. It is used to force the upgrade of the XenAPI guest agent on instance boot.

`os_compute_api:os-aggregates:set_metadata`

Default `rule:admin_api`

Operations

- **POST** `/os-aggregates/{aggregate_id}/action (set_metadata)`

Create or replace metadata for an aggregate

`os_compute_api:os-aggregates:add_host`

Default `rule:admin_api`

Operations

- **POST** `/os-aggregates/{aggregate_id}/action (add_host)`

Add a host to an aggregate

`os_compute_api:os-aggregates:create`

Default `rule:admin_api`

Operations

- **POST** `/os-aggregates`

Create an aggregate

`os_compute_api:os-aggregates:remove_host`

Default `rule:admin_api`

Operations

- **POST** `/os-aggregates/{aggregate_id}/action (remove_host)`

Remove a host from an aggregate

`os_compute_api:os-aggregates:update`

Default `rule:admin_api`

Operations

- **PUT** `/os-aggregates/{aggregate_id}`

Update name and/or availability zone for an aggregate

`os_compute_api:os-aggregates:index`

Default `rule:admin_api`

Operations

- **GET** `/os-aggregates`

List all aggregates

`os_compute_api:os-aggregates:delete`

Default `rule:admin_api`

Operations

- **DELETE** `/os-aggregates/{aggregate_id}`

Delete an aggregate

`os_compute_api:os-aggregates:show`

Default `rule:admin_api`

Operations

- **GET** `/os-aggregates/{aggregate_id}`

Show details for an aggregate

`os_compute_api:os-assisted-volume-snapshots:create`

Default `rule:admin_api`

Operations

- **POST** `/os-assisted-volume-snapshots`

Create an assisted volume snapshot

`os_compute_api:os-assisted-volume-snapshots:delete`

Default `rule:admin_api`

Operations

- **DELETE** `/os-assisted-volume-snapshots/{snapshot_id}`

Delete an assisted volume snapshot

`os_compute_api:os-attach-interfaces`

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}/os-interface`
- **GET** `/servers/{server_id}/os-interface/{port_id}`

List port interfaces or show details of a port interface attached to a server

`os_compute_api:os-attach-interfaces:create`

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/os-interface`

Attach an interface to a server

`os_compute_api:os-attach-interfaces:delete`

Default `rule:admin_or_owner`

Operations

- **DELETE** `/servers/{server_id}/os-interface/{port_id}`

Detach an interface from a server

`os_compute_api:os-availability-zone:list`

Default `rule:admin_or_owner`

Operations

- **GET** `/os-availability-zone`

List availability zone information without host information

`os_compute_api:os-availability-zone:detail`

Default `rule:admin_api`

Operations

- **GET** `/os-availability-zone/detail`

List detailed availability zone information with host information

`os_compute_api:os-baremetal-nodes`

Default `rule:admin_api`

Operations

- **GET** `/os-baremetal-nodes`
- **GET** `/os-baremetal-nodes/{node_id}`

List and show details of bare metal nodes.

These APIs are proxy calls to the Ironic service and are deprecated.

`os_compute_api:os-console-auth-tokens`

Default `rule:admin_api`

Operations

- **GET** `/os-console-auth-tokens/{console_token}`

Show console connection information for a given console authentication token

`os_compute_api:os-console-output`

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (os-getConsoleOutput)`

Show console output for a server

`os_compute_api:os-consoles:create`

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/consoles`

Create a console for a server instance

`os_compute_api:os-consoles:show`

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}/consoles/{console_id}`

Show console details for a server instance

`os_compute_api:os-consoles:delete`

Default `rule:admin_or_owner`

Operations

- **DELETE** `/servers/{server_id}/consoles/{console_id}`

Delete a console for a server instance

`os_compute_api:os-consoles:index`

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}/consoles`

List all consoles for a server instance

`os_compute_api:os-create-backup`

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (createBackup)`

Create a back up of a server

`os_compute_api:os-deferred-delete`

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (restore)`
- **POST** `/servers/{server_id}/action (forceDelete)`

Restore a soft deleted server or force delete a server before deferred cleanup

`os_compute_api:os-evacuate`

Default `rule:admin_api`

Operations

- **POST** `/servers/{server_id}/action (evacuate)`

Evacuate a server from a failed host to a new host

`os_compute_api:os-extended-server-attributes`

Default `rule:admin_api`

Operations

- **GET** `/servers/{id}`
- **GET** `/servers/detail`

Return extended attributes for server.

This rule will control the visibility for a set of servers attributes:

- `OS-EXT-SRV-ATTR:host`
- `OS-EXT-SRV-ATTR:instance_name`
- `OS-EXT-SRV-ATTR:reservation_id` (since microversion 2.3)
- `OS-EXT-SRV-ATTR:launch_index` (since microversion 2.3)
- `OS-EXT-SRV-ATTR:hostname` (since microversion 2.3)

- `OS-EXT-SRV-ATTR:kernel_id` (since microversion 2.3)
- `OS-EXT-SRV-ATTR:ramdisk_id` (since microversion 2.3)
- `OS-EXT-SRV-ATTR:root_device_name` (since microversion 2.3)
- `OS-EXT-SRV-ATTR:user_data` (since microversion 2.3)

`os_compute_api:extensions`

Default `rule:admin_or_owner`

Operations

- **GET** `/extensions`
- **GET** `/extensions/{alias}`

List available extensions and show information for an extension by alias

`os_compute_api:os-flavor-access:add_tenant_access`

Default `rule:admin_api`

Operations

- **POST** `/flavors/{flavor_id}/action (addTenantAccess)`

Add flavor access to a tenant

`os_compute_api:os-flavor-access:remove_tenant_access`

Default `rule:admin_api`

Operations

- **POST** `/flavors/{flavor_id}/action (removeTenantAccess)`

Remove flavor access from a tenant

`os_compute_api:os-flavor-access`

Default `rule:admin_or_owner`

Operations

- **GET** `/flavors/{flavor_id}/os-flavor-access`

List flavor access information

Allows access to the full list of tenants that have access to a flavor via an os-flavor-access API.

`os_compute_api:os-flavor-extra-specs:show`

Default `rule:admin_or_owner`

Operations

- **GET** `/flavors/{flavor_id}/os-extra_specs/{flavor_extra_spec_key}`

Show an extra spec for a flavor

`os_compute_api:os-flavor-extra-specs:create`

Default `rule:admin_api`

Operations

- **POST** `/flavors/{flavor_id}/os-extra_specs/`

Create extra specs for a flavor

os_compute_api:os-flavor-extra-specs:update

Default `rule:admin_api`

Operations

- **PUT** `/flavors/{flavor_id}/os-extra_specs/{flavor_extra_spec_key}`

Update an extra spec for a flavor

os_compute_api:os-flavor-extra-specs:delete

Default `rule:admin_api`

Operations

- **DELETE** `/flavors/{flavor_id}/os-extra_specs/{flavor_extra_spec_key}`

Delete an extra spec for a flavor

os_compute_api:os-flavor-extra-specs:index

Default `rule:admin_or_owner`

Operations

- **GET** `/flavors/{flavor_id}/os-extra_specs/`
- **GET** `/servers/detail`
- **GET** `/servers/{server_id}`
- **PUT** `/servers/{server_id}`
- **POST** `/servers/{server_id}/action (rebuild)`
- **POST** `/flavors`
- **GET** `/flavors/detail`
- **GET** `/flavors/{flavor_id}`
- **PUT** `/flavors/{flavor_id}`

List extra specs for a flavor. Starting with microversion 2.47, the flavor used for a server is also returned in the response when showing server details, updating a server or rebuilding a server. Starting with microversion 2.61, extra specs may be returned in responses for the flavor resource.

os_compute_api:os-flavor-manage:create

Default `rule:admin_api`

Operations

- **POST** `/flavors`

Create a flavor

os_compute_api:os-flavor-manage:update

Default `rule:admin_api`

Operations

- **PUT** `/flavors/{flavor_id}`

Update a flavor

os_compute_api:os-flavor-manage:delete

Default `rule:admin_api`

Operations

- **DELETE** `/flavors/{flavor_id}`

Delete a flavor

os_compute_api:os-floating-ip-pools

Default `rule:admin_or_owner`

Operations

- **GET** `/os-floating-ip-pools`

List floating IP pools. This API is deprecated.

os_compute_api:os-floating-ips

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (addFloatingIp)`
- **POST** `/servers/{server_id}/action (removeFloatingIp)`
- **GET** `/os-floating-ips`
- **POST** `/os-floating-ips`
- **GET** `/os-floating-ips/{floating_ip_id}`
- **DELETE** `/os-floating-ips/{floating_ip_id}`

Manage a projects floating IPs. These APIs are all deprecated.

os_compute_api:os-hosts

Default `rule:admin_api`

Operations

- **GET** `/os-hosts`
- **GET** `/os-hosts/{host_name}`
- **PUT** `/os-hosts/{host_name}`
- **GET** `/os-hosts/{host_name}/reboot`
- **GET** `/os-hosts/{host_name}/shutdown`
- **GET** `/os-hosts/{host_name}/startup`

List, show and manage physical hosts.

These APIs are all deprecated in favor of os-hypervisors and os-services.

os_compute_api:os-hypervisors

Default `rule:admin_api`

Operations

- **GET** `/os-hypervisors`

- **GET** `/os-hypervisors/details`
- **GET** `/os-hypervisors/statistics`
- **GET** `/os-hypervisors/{hypervisor_id}`
- **GET** `/os-hypervisors/{hypervisor_id}/uptime`
- **GET** `/os-hypervisors/{hypervisor_hostname_pattern}/search`
- **GET** `/os-hypervisors/{hypervisor_hostname_pattern}/servers`

Policy rule for hypervisor related APIs.

This rule will be checked for the following APIs:

List all hypervisors, list all hypervisors with details, show summary statistics for all hypervisors over all compute nodes, show details for a hypervisor, show the uptime of a hypervisor, search hypervisor by hypervisor_hostname pattern and list all servers on hypervisors that can match the provided hypervisor_hostname pattern.

`os_compute_api:os-instance-actions:events`

Default `rule:admin_api`

Operations

- **GET** `/servers/{server_id}/os-instance-actions/{request_id}`

Add events details in action details for a server.

This check is performed only after the check os_compute_api:os-instance-actions passes. Beginning with Microversion 2.51, events details are always included; traceback information is provided per event if policy enforcement passes. Beginning with Microversion 2.62, each event includes a hashed host identifier and, if policy enforcement passes, the name of the host.

`os_compute_api:os-instance-actions`

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}/os-instance-actions`
- **GET** `/servers/{server_id}/os-instance-actions/{request_id}`

List actions and show action details for a server.

`os_compute_api:os-instance-usage-audit-log`

Default `rule:admin_api`

Operations

- **GET** `/os-instance_usage_audit_log`
- **GET** `/os-instance_usage_audit_log/{before_timestamp}`

List all usage audits and that occurred before a specified time for all servers on all compute hosts where usage auditing is configured

`os_compute_api:ips:show`

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}/ips/{network_label}`

Show IP addresses details for a network label of a server

os_compute_api:ips:index

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}/ips`

List IP addresses that are assigned to a server

os_compute_api:os-keypairs:index

Default `rule:admin_api or user_id:%(user_id)s`

Operations

- **GET** `/os-keypairs`

List all keypairs

os_compute_api:os-keypairs:create

Default `rule:admin_api or user_id:%(user_id)s`

Operations

- **POST** `/os-keypairs`

Create a keypair

os_compute_api:os-keypairs:delete

Default `rule:admin_api or user_id:%(user_id)s`

Operations

- **DELETE** `/os-keypairs/{keypair_name}`

Delete a keypair

os_compute_api:os-keypairs:show

Default `rule:admin_api or user_id:%(user_id)s`

Operations

- **GET** `/os-keypairs/{keypair_name}`

Show details of a keypair

os_compute_api:limits

Default `rule:admin_or_owner`

Operations

- **GET** `/limits`

Show rate and absolute limits for the project

os_compute_api:os-lock-server:lock

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (lock)`

Lock a server

os_compute_api:os-lock-server:unlock

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (unlock)`

Unlock a server

os_compute_api:os-lock-server:unlock:unlock_override

Default `rule:admin_api`

Operations

- **POST** `/servers/{server_id}/action (unlock)`

Unlock a server, regardless who locked the server.

This check is performed only after the check os_compute_api:os-lock-server:unlock passes

os_compute_api:os-migrate-server:migrate

Default `rule:admin_api`

Operations

- **POST** `/servers/{server_id}/action (migrate)`

Cold migrate a server to a host

os_compute_api:os-migrate-server:migrate_live

Default `rule:admin_api`

Operations

- **POST** `/servers/{server_id}/action (os-migrateLive)`

Live migrate a server to a new host without a reboot

os_compute_api:os-migrations:index

Default `rule:admin_api`

Operations

- **GET** `/os-migrations`

List migrations

os_compute_api:os-multinic

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (addFixedIp)`
- **POST** `/servers/{server_id}/action (removeFixedIp)`

Add or remove a fixed IP address from a server.

These APIs are proxy calls to the Network service. These are all deprecated.

os_compute_api:os-networks

Default `rule:admin_api`

Operations

- **POST** `/os-networks`
- **POST** `/os-networks/add`
- **DELETE** `/os-networks/{network_id}`
- **POST** `/os-networks/{network_id}/action (disassociate)`

Create and delete a network, add and disassociate a network from a project.

These APIs are only available with nova-network which is deprecated.

`os_compute_api:os-networks:view`

Default `rule:admin_or_owner`

Operations

- **GET** `/os-networks`
- **GET** `/os-networks/{network_id}`

List networks for the project and show details for a network.

These APIs are proxy calls to the Network service. These are all deprecated.

`os_compute_api:os-networks-associate`

Default `rule:admin_api`

Operations

- **POST** `/os-networks/{network_id}/action (disassociate_host)`
- **POST** `/os-networks/{network_id}/action (disassociate_project)`
- **POST** `/os-networks/{network_id}/action (associate_host)`

Associate or disassociate a network from a host or project.

These APIs are only available with nova-network which is deprecated.

`os_compute_api:os-pause-server:pause`

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (pause)`

Pause a server

`os_compute_api:os-pause-server:unpause`

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (unpause)`

Unpause a paused server

`os_compute_api:os-quota-class-sets:show`

Default `is_admin:True or quota_class:%(quota_class)s`

Operations

- **GET** `/os-quota-class-sets/{quota_class}`

List quotas for specific quota classs

os_compute_api:os-quota-class-sets:update

Default `rule:admin_api`

Operations

- **PUT** `/os-quota-class-sets/{quota_class}`

Update quotas for specific quota class

os_compute_api:os-quota-sets:update

Default `rule:admin_api`

Operations

- **PUT** `/os-quota-sets/{tenant_id}`

Update the quotas

os_compute_api:os-quota-sets:defaults

Default `@`

Operations

- **GET** `/os-quota-sets/{tenant_id}/defaults`

List default quotas

os_compute_api:os-quota-sets:show

Default `rule:admin_or_owner`

Operations

- **GET** `/os-quota-sets/{tenant_id}`

Show a quota

os_compute_api:os-quota-sets:delete

Default `rule:admin_api`

Operations

- **DELETE** `/os-quota-sets/{tenant_id}`

Revert quotas to defaults

os_compute_api:os-quota-sets:detail

Default `rule:admin_or_owner`

Operations

- **GET** `/os-quota-sets/{tenant_id}/detail`

Show the detail of quota

os_compute_api:os-remote-consoles

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (os-getRDPConsole)`
- **POST** `/servers/{server_id}/action (os-getSerialConsole)`
- **POST** `/servers/{server_id}/action (os-getSPICEConsole)`

- **POST** `/servers/{server_id}/action (os-getVNCConsole)`
- **POST** `/servers/{server_id}/remote-consoles`

Generate a URL to access remove server console

`os_compute_api:os-rescue`

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (rescue)`
- **POST** `/servers/{server_id}/action (unrescue)`

Rescue/unrescue a server

`os_compute_api:os-security-group-default-rules`

Default `rule:admin_api`

Operations

- **GET** `/os-security-group-default-rules`
- **GET** `/os-security-group-default-rules/{security_group_default_rule_id}`
- **POST** `/os-security-group-default-rules`
- **DELETE** `/os-security-group-default-rules/{security_group_default_rule_id}`

List, show information for, create, or delete default security group rules.

These APIs are only available with nova-network which is now deprecated.

`os_compute_api:os-security-groups`

Default `rule:admin_or_owner`

Operations

- **GET** `/os-security-groups`
- **GET** `/os-security-groups/{security_group_id}`
- **POST** `/os-security-groups`
- **PUT** `/os-security-groups/{security_group_id}`
- **DELETE** `/os-security-groups/{security_group_id}`
- **GET** `/servers/{server_id}/os-security-groups`
- **POST** `/servers/{server_id}/action (addSecurityGroup)`
- **POST** `/servers/{server_id}/action (removeSecurityGroup)`

List, show, add, or remove security groups.

APIs which are directly related to security groups resource are deprecated: Lists, shows information for, creates, updates and deletes security groups. Creates and deletes security group rules. All these APIs are deprecated.

APIs which are related to server resource are not deprecated: Lists Security Groups for a server. Add Security Group to a server and remove security group from a server.

`os_compute_api:os-server-diagnostics`

Default `rule:admin_api`

Operations

- **GET** `/servers/{server_id}/diagnostics`

Show the usage data for a server

`os_compute_api:os-server-external-events:create`

Default `rule:admin_api`

Operations

- **POST** `/os-server-external-events`

Create one or more external events

`os_compute_api:os-server-groups:create`

Default `rule:admin_or_owner`

Operations

- **POST** `/os-server-groups`

Create a new server group

`os_compute_api:os-server-groups:delete`

Default `rule:admin_or_owner`

Operations

- **DELETE** `/os-server-groups/{server_group_id}`

Delete a server group

`os_compute_api:os-server-groups:index`

Default `rule:admin_or_owner`

Operations

- **GET** `/os-server-groups`

List all server groups

`os_compute_api:os-server-groups:show`

Default `rule:admin_or_owner`

Operations

- **GET** `/os-server-groups/{server_group_id}`

Show details of a server group

`os_compute_api:server-metadata:index`

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}/metadata`

List all metadata of a server

`os_compute_api:server-metadata:show`

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}/metadata/{key}`

Show metadata for a server

`os_compute_api:server-metadata:create`

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/metadata`

Create metadata for a server

`os_compute_api:server-metadata:update_all`

Default `rule:admin_or_owner`

Operations

- **PUT** `/servers/{server_id}/metadata`

Replace metadata for a server

`os_compute_api:server-metadata:update`

Default `rule:admin_or_owner`

Operations

- **PUT** `/servers/{server_id}/metadata/{key}`

Update metadata from a server

`os_compute_api:server-metadata:delete`

Default `rule:admin_or_owner`

Operations

- **DELETE** `/servers/{server_id}/metadata/{key}`

Delete metadata from a server

`os_compute_api:os-server-password`

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}/os-server-password`
- **DELETE** `/servers/{server_id}/os-server-password`

Show and clear the encrypted administrative password of a server

`os_compute_api:os-server-tags:delete_all`

Default `rule:admin_or_owner`

Operations

- **DELETE** `/servers/{server_id}/tags`

Delete all the server tags

`os_compute_api:os-server-tags:index`

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}/tags`

List all tags for given server

`os_compute_api:os-server-tags:update_all`

Default `rule:admin_or_owner`

Operations

- **PUT** `/servers/{server_id}/tags`

Replace all tags on specified server with the new set of tags.

`os_compute_api:os-server-tags:delete`

Default `rule:admin_or_owner`

Operations

- **DELETE** `/servers/{server_id}/tags/{tag}`

Delete a single tag from the specified server

`os_compute_api:os-server-tags:update`

Default `rule:admin_or_owner`

Operations

- **PUT** `/servers/{server_id}/tags/{tag}`

Add a single tag to the server if server has no specified tag

`os_compute_api:os-server-tags:show`

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}/tags/{tag}`

Check tag existence on the server.

`compute:server:topology:index`

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}/topology`

Show the NUMA topology data for a server

`compute:server:topology:host:index`

Default `rule:admin_api`

Operations

- **GET** `/servers/{server_id}/topology`

Show the NUMA topology data for a server with host NUMA ID and CPU pinning information

`os_compute_api:servers:index`

Default `rule:admin_or_owner`

Operations

- **GET** `/servers`

List all servers

os_compute_api:servers:detail

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/detail`

List all servers with detailed information

os_compute_api:servers:index:get_all_tenants

Default `rule:admin_api`

Operations

- **GET** `/servers`

List all servers for all projects

os_compute_api:servers:detail:get_all_tenants

Default `rule:admin_api`

Operations

- **GET** `/servers/detail`

List all servers with detailed information for all projects

os_compute_api:servers:allow_all_filters

Default `rule:admin_api`

Operations

- **GET** `/servers`
- **GET** `/servers/detail`

Allow all filters when listing servers

os_compute_api:servers:show

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}`

Show a server

os_compute_api:servers:show:host_status

Default `rule:admin_api`

Operations

- **GET** `/servers/{server_id}`
- **GET** `/servers/detail`

Show a server with additional host status information

os_compute_api:servers:create

Default `rule:admin_or_owner`

Operations

- **POST** `/servers`

Create a server

`os_compute_api:servers:create:forced_host`

Default `rule:admin_api`

Operations

- **POST** `/servers`

Create a server on the specified host and/or node.

In this case, the server is forced to launch on the specified host and/or node by bypassing the scheduler filters unlike the `compute:servers:create:requested_destination` rule.

`compute:servers:create:requested_destination`

Default `rule:admin_api`

Operations

- **POST** `/servers`

Create a server on the requested compute service host and/or hypervisor_hostname.

In this case, the requested host and/or hypervisor_hostname is validated by the scheduler filters unlike the `os_compute_api:servers:create:forced_host` rule.

`os_compute_api:servers:create:attach_volume`

Default `rule:admin_or_owner`

Operations

- **POST** `/servers`

Create a server with the requested volume attached to it

`os_compute_api:servers:create:attach_network`

Default `rule:admin_or_owner`

Operations

- **POST** `/servers`

Create a server with the requested network attached to it

`os_compute_api:servers:create:trusted_certs`

Default `rule:admin_or_owner`

Operations

- **POST** `/servers`

Create a server with trusted image certificate IDs

`os_compute_api:servers:create:zero_disk_flavor`

Default `rule:admin_api`

Operations

- **POST** `/servers`

This rule controls the compute API validation behavior of creating a server with a flavor that has 0 disk, indicating the server should be volume-backed.

For a flavor with disk=0, the root disk will be set to exactly the size of the image used to deploy the instance. However, in this case the filter_scheduler cannot select the compute host based on the virtual image size. Therefore, 0 should only be used for volume booted instances or for testing purposes.

WARNING: It is a potential security exposure to enable this policy rule if users can upload their own images since repeated attempts to create a disk=0 flavor instance with a large image can exhaust the local disk of the compute (or shared storage cluster). See bug https://bugs.launchpad.net/nova/+bug/1739646 for details.

network:attach_external_network

Default `is_admin:True`

Operations

- **POST** `/servers`
- **POST** `/servers/{server_id}/os-interface`

Attach an unshared external network to a server

os_compute_api:servers:delete

Default `rule:admin_or_owner`

Operations

- **DELETE** `/servers/{server_id}`

Delete a server

os_compute_api:servers:update

Default `rule:admin_or_owner`

Operations

- **PUT** `/servers/{server_id}`

Update a server

os_compute_api:servers:confirm_resize

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (confirmResize)`

Confirm a server resize

os_compute_api:servers:revert_resize

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (revertResize)`

Revert a server resize

os_compute_api:servers:reboot

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (reboot)`

Reboot a server

os_compute_api:servers:resize

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (resize)`

Resize a server

os_compute_api:servers:rebuild

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (rebuild)`

Rebuild a server

os_compute_api:servers:rebuild:trusted_certs

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (rebuild)`

Rebuild a server with trusted image certificate IDs

os_compute_api:servers:create_image

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (createImage)`

Create an image from a server

os_compute_api:servers:create_image:allow_volume_backed

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (createImage)`

Create an image from a volume backed server

os_compute_api:servers:start

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (os-start)`

Start a server

os_compute_api:servers:stop

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (os-stop)`

Stop a server

os_compute_api:servers:trigger_crash_dump

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (trigger_crash_dump)`

Trigger crash dump in a server

os_compute_api:servers:migrations:show

Default `rule:admin_api`

Operations

- **GET** `/servers/{server_id}/migrations/{migration_id}`

Show details for an in-progress live migration for a given server

os_compute_api:servers:migrations:force_complete

Default `rule:admin_api`

Operations

- **POST** `/servers/{server_id}/migrations/{migration_id}/action (force_complete)`

Force an in-progress live migration for a given server to complete

os_compute_api:servers:migrations:delete

Default `rule:admin_api`

Operations

- **DELETE** `/servers/{server_id}/migrations/{migration_id}`

Delete(Abort) an in-progress live migration

os_compute_api:servers:migrations:index

Default `rule:admin_api`

Operations

- **GET** `/servers/{server_id}/migrations`

Lists in-progress live migrations for a given server

os_compute_api:os-services

Default `rule:admin_api`

Operations

- **GET** `/os-services`
- **PUT** `/os-services/enable`
- **PUT** `/os-services/disable`
- **PUT** `/os-services/disable-log-reason`
- **PUT** `/os-services/force-down`
- **PUT** `/os-services/{service_id}`
- **DELETE** `/os-services/{service_id}`

List all running Compute services in a region, enables or disable scheduling for a Compute service, logs disabled Compute service information, set or unset forced_down flag for the compute service and delete a Compute service

os_compute_api:os-shelve:shelve

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (shelve)`

Shelve server

os_compute_api:os-shelve:unshelve

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (unshelve)`

Unshelve (restore) shelved server

os_compute_api:os-shelve:shelve_offload

Default `rule:admin_api`

Operations

- **POST** `/servers/{server_id}/action (shelveOffload)`

Shelf-offload (remove) server

os_compute_api:os-simple-tenant-usage:show

Default `rule:admin_or_owner`

Operations

- **GET** `/os-simple-tenant-usage/{tenant_id}`

Show usage statistics for a specific tenant

os_compute_api:os-simple-tenant-usage:list

Default `rule:admin_api`

Operations

- **GET** `/os-simple-tenant-usage`

List per tenant usage statistics for all tenants

os_compute_api:os-suspend-server:resume

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (resume)`

Resume suspended server

os_compute_api:os-suspend-server:suspend

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/action (suspend)`

Suspend server

os_compute_api:os-tenant-networks

Default `rule:admin_or_owner`

Operations

- **GET** `/os-tenant-networks`
- **POST** `/os-tenant-networks`
- **GET** `/os-tenant-networks/{network_id}`
- **DELETE** `/os-tenant-networks/{network_id}`

Create, list, show information for, and delete project networks.

These APIs are proxy calls to the Network service. These are all deprecated.

os_compute_api:os-used-limits

Default `rule:admin_api`

Operations

- **GET** `/limits`

Show rate and absolute limits for the project.

This policy only checks if the user has access to the requested project limits. And this check is performed only after the check os_compute_api:limits passes

os_compute_api:os-volumes

Default `rule:admin_or_owner`

Operations

- **GET** `/os-volumes`
- **POST** `/os-volumes`
- **GET** `/os-volumes/detail`
- **GET** `/os-volumes/{volume_id}`
- **DELETE** `/os-volumes/{volume_id}`
- **GET** `/os-snapshots`
- **POST** `/os-snapshots`
- **GET** `/os-snapshots/detail`
- **GET** `/os-snapshots/{snapshot_id}`
- **DELETE** `/os-snapshots/{snapshot_id}`

Manage volumes for use with the Compute API.

Lists, shows details, creates, and deletes volumes and snapshots. These APIs are proxy calls to the Volume service. These are all deprecated.

os_compute_api:os-volumes-attachments:index

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}/os-volume_attachments`

List volume attachments for an instance

os_compute_api:os-volumes-attachments:create

Default `rule:admin_or_owner`

Operations

- **POST** `/servers/{server_id}/os-volume_attachments`

Attach a volume to an instance

os_compute_api:os-volumes-attachments:show

Default `rule:admin_or_owner`

Operations

- **GET** `/servers/{server_id}/os-volume_attachments/{volume_id}`

Show details of a volume attachment

os_compute_api:os-volumes-attachments:update

Default `rule:admin_api`

Operations

- **PUT** `/servers/{server_id}/os-volume_attachments/{volume_id}`

Update a volume attachment

os_compute_api:os-volumes-attachments:delete

Default `rule:admin_or_owner`

Operations

- **DELETE** `/servers/{server_id}/os-volume_attachments/{volume_id}`

Detach a volume from an instance

4.6 Development Quickstart

This page describes how to setup and use a working Python development environment that can be used in developing nova on Ubuntu, Fedora or Mac OS X. These instructions assume youre already familiar with git.

Following these instructions will allow you to build the documentation and run the nova unit tests. If you want to be able to run nova (i.e., launch VM instances), you will also need to either manually or by letting DevStack do it for you install libvirt and at least one of the supported hypervisors. Running nova is currently only supported on Linux, although you can run the unit tests on Mac OS X.

Note: For how to contribute to Nova, see HowToContribute. Nova uses the Gerrit code review system, GerritWorkflow.

4.6.1 Setup

There are two ways to create a development environment: using DevStack, or explicitly installing and cloning just what you need.

Using DevStack

See Devstack Documentation. If you would like to use Vagrant, there is a Vagrant for DevStack.

Explicit Install/Clone

DevStack installs a complete OpenStack environment. Alternatively, you can explicitly install and clone just what you need for Nova development.

Getting the code

Grab the code from git:

```
git clone https://opendev.org/openstack/nova
cd nova
```

Linux Systems

The first step of this process is to install the system (not Python) packages that are required. Following are instructions on how to do this on Linux and on the Mac.

Note: This section is tested for Nova on Ubuntu (14.04-64) and Fedora-based (RHEL 6.1) distributions. Feel free to add notes and change according to your experiences or operating system.

Install the prerequisite packages listed in the `bindep.txt` file.

On Debian-based distributions (e.g., Debian/Mint/Ubuntu):

```
sudo apt-get install python-pip
sudo pip install tox
tox -e bindep
sudo apt-get install <indicated missing package names>
```

On Fedora-based distributions (e.g., Fedora/RHEL/CentOS/Scientific Linux):

```
sudo yum install python-pip
sudo pip install tox
tox -e bindep
sudo yum install <indicated missing package names>
```

On openSUSE-based distributions (SLES, openSUSE Leap / Tumbleweed):

```
sudo zypper in python-pip
sudo pip install tox
tox -e bindep
sudo zypper in <indicated missing package names>
```

Mac OS X Systems

Install virtualenv:

```
sudo easy_install virtualenv
```

Check the version of OpenSSL you have installed:

```
openssl version
```

The stock version of OpenSSL that ships with Mac OS X 10.6 (OpenSSL 0.9.8l) or Mac OS X 10.7 (OpenSSL 0.9.8r) or Mac OS X 10.10.3 (OpenSSL 0.9.8zc) works fine with nova. OpenSSL versions from brew like OpenSSL 1.0.1k work fine as well.

Brew is very useful for installing dependencies. As a minimum for running tests, install the following:

```
brew install python3 postgres
python3 -mpip install tox
```

4.6.2 Building the Documentation

Install the prerequisite packages: graphviz

To do a full documentation build, issue the following command while the nova directory is current.

```
tox -edocs
```

That will create a Python virtual environment, install the needed Python prerequisites in that environment, and build all the documentation in that environment.

4.6.3 Running unit tests

See Running Python Unit Tests.

Note that some unit and functional tests use a database. See the file `tools/test-setup.sh` on how the databases are set up in the OpenStack CI environment and replicate it in your test environment.

4.6.4 Using a remote debugger

Some modern IDE such as pycharm (commercial) or Eclipse (open source) support remote debugging. In order to run nova with remote debugging, start the nova process with the following parameters:

```
--remote_debug-host <host IP where the debugger is running>
--remote_debug-port <port it is listening on>
```

Before you start your nova process, start the remote debugger using the instructions for that debugger:

- For pycharm - http://blog.jetbrains.com/pycharm/2010/12/python-remote-debug-with-pycharm/
- For Eclipse - http://pydev.org/manual_adv_remote_debugger.html

More detailed instructions are located here - https://wiki.openstack.org/wiki/Nova/RemoteDebugging

4.6.5 Using fake computes for tests

The number of instances supported by fake computes is not limited by physical constraints. It allows you to perform stress tests on a deployment with few resources (typically a laptop). Take care to avoid using scheduler filters that will limit the number of instances per compute, such as `AggregateCoreFilter`.

Fake computes can also be used in multi hypervisor-type deployments in order to take advantage of fake and real computes during tests:

- create many fake instances for stress tests
- create some real instances for functional tests

Fake computes can be used for testing Nova itself but also applications on top of it.

4.7 Extending the API

4.7.1 Background

Nova has v2.1 API frameworks which supports microversions.

This document covers how to add API for the v2.1 API framework. A *microversions specific document* covers the details around what is required for the microversions part.

The v2.1 API framework is under `nova/api` and each API is implemented in `nova/api/openstack/compute`.

Note that any change to the Nova API to be merged will first require a spec be approved first. See here for the appropriate repository. For guidance on the design of the API please refer to the OpenStack API WG

4.7.2 Basic API Controller

API controller includes the implementation of API methods for a resource.

A very basic controller of a v2.1 API:

```
"""Basic Controller"""

from nova.api.openstack.compute.schemas import xyz
from nova.api.openstack import extensions
from nova.api.openstack import wsgi
from nova.api import validation

class BasicController(wsgi.Controller):

    # Define support for GET on a collection
    def index(self, req):
        data = {'param': 'val'}
        return data

    # Define support for POST on a collection
    @extensions.expected_errors((400, 409))
    @validation.schema(xyz.create)
    @wsgi.response(201)
    def create(self, req, body):
        write_body_here = ok
        return response_body

    # Defining support for other RESTFul methods based on resource.
```

See servers.py for ref.

All of the controller modules should live in the `nova/api/openstack/compute` directory.

URL Mapping to API

The URL mapping is based on the plain list which routes the API request to appropriate controller and method. Each API needs to add its route information in `nova/api/openstack/compute/routes.py`.

A basic skeleton of URL mapping in routers.py:

```
"""URL Mapping Router List"""

import functools

import nova.api.openstack
from nova.api.openstack.compute import basic_api

# Create a controller object
basic_controller = functools.partial(
    _create_controller, basic_api.BasicController, [], [])

# Routing list structure:
# (
#     ('Route path': {
#         'HTTP method: [
#             'Controller',
#             'The method of controller is used to handle this route'
#         ],
#         ...
#     }),
#     ...
# )
ROUTE_LIST = (
    .
    .
    .
    ('/basic', {
        'GET': [basic_controller, 'index'],
        'POST': [basic_controller, 'create']
    }),
    .
    .
    .
)
```

Complete routing list can be found in routes.py.

Policy

For more info about policy, see *policies*, Also look at the `context.can(...)` call in existing API controllers.

Modularity

The Nova REST API is separated into different controllers in the directory nova/api/openstack/compute/

Because microversions are supported in the Nova REST API, the API can be extended without any new controller. But for code readability, the Nova REST API code still needs modularity. Here are rules for how to separate modules:

- You are adding a new resource The new resource should be in standalone module. There isnt any reason to put different resources in a single module.

- Add sub-resource for existing resource To prevent an existing resource module becoming over-inflated, the sub-resource should be implemented in a separate module.
- Add extended attributes for existing resource In normally, the extended attributes is part of existing resources data model too. So this can be added into existing resource module directly and lightly. To avoid namespace complexity, we should avoid to add extended attributes in existing extended models. New extended attributes neednt any namespace prefix anymore.

JSON-Schema

The v2.1 API validates a REST request body with JSON-Schema library. Valid body formats are defined with JSON-Schema in the directory nova/api/openstack/compute/schemas. Each definition is used at the corresponding method with the `validation.schema` decorator like:

```
@validation.schema(schema.update_something)
def update(self, req, id, body):
    ....
```

Similarly to controller modularity, JSON-Schema definitions can be added in same or separate JSON-Schema module.

The following are the combinations of extensible API and method name which returns additional JSON-Schema parameters:

- Create a server API - get_server_create_schema()

For example, keypairs extension(Keypairs class) contains the method get_server_create_schema() which returns:

```
{
    'key_name': parameter_types.name,
}
```

then the parameter key_name is allowed on Create a server API.

Note: Currently only create schema are implemented in modular way. Final goal is to merge them all and define the concluded process in this doc.

These are essentially hooks into the servers controller which allow other controller to modify behaviour without having to modify servers.py. In the past not having this capability led to very large chunks of unrelated code being added to servers.py which was difficult to maintain.

4.7.3 Unit Tests

Unit tests for the API can be found under path `nova/tests/unit/api/openstack/compute/`. Unit tests for the API are generally negative scenario tests, because the positive scenarios are tested with functional API samples tests.

Negative tests would include such things as:

- Request schema validation failures, for both the request body and query parameters
- HTTPNotFound or other >=400 response code failures

4.7.4 Functional tests and API Samples

All functional API changes, including new microversions - especially if there are new request or response parameters, should have new functional API samples tests.

The API samples tests are made of two parts:

- The API sample for the reference docs. These are found under path `doc/api_samples/`. There is typically one directory per API controller with subdirectories per microversion for that API controller. The unversioned samples are used for the base v2.0 / v2.1 APIs.
- Corresponding API sample templates found under path `nova/tests/functional/api_sample_tests/api_samples`. These have a similar structure to the API reference docs samples, except the format of the sample can include substitution variables filled in by the tests where necessary, for example, to substitute things that change per test run, like a server UUID.

The actual functional tests are found under path `nova/tests/functional/api_sample_tests/`. Most, if not all, API samples tests extend the `ApiSampleTestBaseV21` class which extends `ApiSampleTestBase`. These base classes provide the framework for making a request using an API reference doc sample and validating the response using the corresponding template file, along with any variable substitutions that need to be made.

Note that it is possible to automatically generate the API reference doc samples using the templates by simply running the tests using `tox -r -e api-samples`. This relies, of course, upon the test and templates being correct for the test to pass, which may take some iteration.

In general, if you are adding a new microversion to an existing API controller, it is easiest to simply copy an existing test and modify it for the new microversion and the new samples/templates.

The functional API samples tests are not the simplest thing in the world to get used to, and can be very frustrating at times when they fail in not obvious ways. If you need help debugging a functional API sample test failure, feel free to post your work-in-progress change for review and ask for help in the `openstack-nova` freenode IRC channel.

4.7.5 Documentation

All API changes must also include updates to the compute API reference, which can be found under path `api-ref/source/`.

Things to consider here include:

- Adding new request and/or response parameters with a new microversion
- Marking existing parameters as deprecated in a new microversion

More information on the compute API reference format and conventions can be found in the *API reference guideline*.

For more detailed documentation of certain aspects of the API, consider writing something into the compute API guide found under path `api-guide/source/`.

4.7.6 Deprecating APIs

Compute REST API routes may be deprecated by capping a method or functionality using microversions. For example, the *2.36 microversion* deprecated several compute REST API routes which only work when using the `nova-network` service, which itself was deprecated, or are proxies to other external services like Cinder, Neutron, etc.

The point of deprecating with microversions is users can still get the same functionality at a lower microversion but there is at least some way to signal to users that they should stop using the REST API.

The general steps for deprecating a REST API are:

- Set a maximum allowed microversion for the route. Requests beyond that microversion on that route will result in a `404 HTTPNotFound` error.
- Update the Compute API reference documentation to indicate the route is deprecated and move it to the bottom of the list with the other deprecated APIs.
- Deprecate, and eventually remove, related CLI / SDK functionality in other projects like python-novaclient.

4.7.7 Removing deprecated APIs

Nova tries to maintain backward compatibility with all REST APIs as much as possible, but when enough time has lapsed, there are few (if any) users or there are supported alternatives, the underlying service code that supports a deprecated REST API, like in the case of `nova-network`, is removed and the REST API must also be effectively removed.

The general steps for removing support for a deprecated REST API are:

- The route mapping will remain but all methods will return a `410 HTTPGone` error response. This is slightly different then the `404 HTTPNotFound` error response a user will get for trying to use a microversion that does not support a deprecated API. 410 means the resource is gone and not coming back, which is more appropriate when the API is fully removed and will not work at any microversion.
- Related configuration options, policy rules, and schema validation are removed.
- The API reference documentation should be updated to move the documentation for the removed API to the Obsolete APIs section and mention in which release the API was removed.
- Unit tests can be removed.
- API sample functional tests can be changed to assert the 410 response behavior, but can otherwise be mostly gutted. Related *.tpl files for the API sample functional tests can be deleted since they will not be used.
- An upgrade *release note* should be added to mention the REST API routes that were removed along with any related configuration options that were also removed.

Here is an example of the above steps: https://review.opendev.org/567682/

Todo: This should be merged into contributor/api

4.8 Adding a Method to the OpenStack API

The interface is a mostly RESTful API. REST stands for Representational State Transfer and provides an architecture style for distributed systems using HTTP for transport. Figure out a way to express your request and response in terms of resources that are being created, modified, read, or destroyed.

4.8.1 Routing

To map URLs to controllers+actions, OpenStack uses the Routes package, a clone of Rails routes for Python implementations. See http://routes.readthedocs.io/en/latest/ for more information.

URLs are mapped to action methods on controller classes in `nova/api/openstack/__init__/ApiRouter.__init__`.

See http://routes.readthedocs.io/en/latest/modules/mapper.html for all syntax, but youll probably just need these two:

- mapper.connect() lets you map a single URL to a single action on a controller.
- mapper.resource() connects many standard URLs to actions on a controller.

4.8.2 Controllers and actions

Controllers live in `nova/api/openstack`, and inherit from nova.wsgi.Controller.

See `nova/api/openstack/compute/servers.py` for an example.

Action methods take parameters that are sucked out of the URL by mapper.connect() or .resource(). The first two parameters are self and the WebOb request, from which you can get the req.environ, req.body, req.headers, etc.

4.8.3 Serialization

Actions return a dictionary, and wsgi.Controller serializes that to JSON.

4.8.4 Faults

If you need to return a non-200, you should return faults.Fault(webob.exc.HTTPNotFound()) replacing the exception as appropriate.

4.9 API reference guideline

The API reference should be updated when compute APIs are modified (microversion is bumped, etc.). This page describes the guideline for updating the API reference.

4.9.1 API reference

- Compute API reference

4.9.2 The guideline to write the API reference

The API reference consists of the following files.

Compute API reference

- API reference text: `api-ref/source/*.inc`
- Parameter definition: `api-ref/source/parameters.yaml`
- JSON request/response samples: `doc/api_samples/*`

Structure of inc file

Each REST API is described in the text file (*.inc). The structure of inc file is as follows:

- Title (Resource name)
  - Introductory text and context

    The introductory text and the context for the resource in question should be added. This might include links to the API Concept guide, or building other supporting documents to explain a concept (like versioning).
  - API Name
    - REST Method
      - URL
      - Description

        See the *Description* section for more details.
      - Response codes
      - Request
        - Parameters
        - JSON request body example (if exists)
      - Response
        - Parameters
        - JSON response body example (if exists)
  - API Name (Next)
    -

REST Method

The guideline for describing HTTP methods is described in this section. All supported methods by resource should be listed in the API reference.

The order of methods

Methods have to be sorted by each URI in the following order:

1. GET
2. POST
3. PUT
4. PATCH (unused by Nova)
5. DELETE

And sorted from broadest to narrowest. So for /severs it would be:

1. GET /servers
2. POST /servers
3. GET /servers/details

4. GET /servers/{server_id}
5. PUT /servers/{server_id}
6. DELETE /servers/{server_id}

Method titles spelling and case

The spelling and the case of method names in the title have to match what is in the code. For instance, the title for the section on method Get Rdp Console should be Get Rdp Console (os-getRDPConsole Action) NOT Get Rdp Console (Os-Getrdpconsole Action)

Description

The following items should be described in each API. Or links to the pages describing them should be added.

- The purpose of the API(s)
  - e.g. Lists, creates, shows details for, updates, and deletes servers.
  - e.g. Creates a server.
- Microversion
  - Deprecated
    - Warning
    - Microversion to start deprecation
    - Alternatives (superseded ways) and their links (if document is available)
  - Added
    - Microversion in which the API has been added
  - Changed behavior
    - Microversion to change behavior
    - Explanation of the behavior
  - Changed HTTP response codes
    - Microversion to change the response code
    - Explanation of the response code
- Warning if direct use is not recommended
  - e.g. This is an admin level service API only designed to be used by other OpenStack services. The point of this API is to coordinate between Nova and Neutron, Nova and Cinder (and potentially future services) on activities they both need to be involved in, such as network hotplugging. Unless you are writing Neutron or Cinder code you should not be using this API.
- Explanation about statuses of resource in question
  - e.g. The server status.
    - `ACTIVE`. The server is active.
- Supplementary explanation for parameters
  - Examples of query parameters

  - Parameters that are not specified at the same time
  - Values that cannot be specified.
    - e.g. A destination host is the same host.
- Behavior
  - Config options to change the behavior and the effect
  - Effect to resource status
    - Ephemeral disks, attached volumes, attached network ports and others
    - Data loss or preserve contents
  - Scheduler
    - Whether the scheduler choose a destination host or not
- Sort order of response results
  - Describe sorting order of response results if the API implements the order (e.g. The response is sorted by `created_at` and `id` in descending order by default)
- Policy
  - Default policy (the admin only, the admin or the owner)
  - How to change the policy
- Preconditions
  - Server status
  - Task state
  - Policy for locked servers
  - Quota
  - Limited support
    - e.g. Only qcow2 is supported
  - Compute driver support
    - If very few compute drivers support the operation, add a warning and a link to the support matrix of virt driver.
  - Cases that are not supported
    - e.g. A volume-backed server
- Postconditions
  - If the operation is asynchronous, it should be Asynchronous postconditions.
  - Describe what status/state resource in question becomes by the operation
    - Server status
    - Task state
    - Path of output file
- Troubleshooting
  - e.g. If the server status remains `BUILDING` or shows another error status, the request failed. Ensure you meet the preconditions then investigate the compute node.

- Related operations
  - Operations to be paired
    - e.g. Start and stop
  - Subsequent operation
    - e.g. Confirm resize after Resize operation
- Performance
  - e.g. The progress of this operation depends on the location of the requested image, network I/O, host load, selected flavor, and other factors.
- Progress
  - How to get progress of the operation if the operation is asynchronous.
- Restrictions
  - Range that ID is unique
    - e.g. HostId is unique per account and is not globally unique.
- How to avoid errors
  - e.g. The server to get console log from should set `export LC_ALL=en_US.UTF-8` in order to avoid incorrect unicode error.
- Reference
  - Links to the API Concept guide, or building other supporting documents to explain a concept (like versioning).
- Other notices

Response codes

The normal response codes (20x) and error response codes have to be listed. The order of response codes should be in ascending order. The description of typical error response codes are as follows:

Table 167: Error response codes

Response codes	Description
400	badRequest(400)
401	unauthorized(401)
403	forbidden(403)
404	itemNotFound(404)
409	conflict(409)
410	gone(410)
501	notImplemented(501)
503	serviceUnavailable(503)

In addition, the following explanations should be described.

- Conditions under which each normal response code is returned (If there are multiple normal response codes.)
- Conditions under which each error response code is returned

Parameters

Parameters need to be defined by 2 subsections. The one is in the Request subsection, the other is in the Response subsection. The queries, request headers and attributes go in the Request subsection and response headers and attributes go in the Response subsection.

The order of parameters in each API

The request and response parameters have to be listed in the following order in each API in the text file.

1. Header
2. Path
3. Query
4. Body
    a. Top level object (i.e. server)
    b. Required fields
    c. Optional fields
    d. Parameters added in microversions (by the microversion they were added)

Parameter type

The parameters are defined in the parameter file (`parameters.yaml`). The type of parameters have to be one of followings:

- `array`

  It is a list.

- `boolean`
- `float`
- `integer`
- `none`

  The value is always `null` in a response or should be `null` in a request.

- `object`

  The value is dict.

- `string`

  If the value can be specified by multiple types, specify one type in the file and mention the other types in the description.

Required or optional

In the parameter file, define the `required` field in each parameter.

`true`	The parameter must be specified in the request, or the parameter always appears in the response.
`false`	It is not always necessary to specify the parameter in the request, or the parameter does not appear in the response in some cases. e.g. A config option defines whether the parameter appears in the response or not. A parameter appears when administrators call but does not appear when non-admin users call.

If a parameter must be specified in the request or always appears in the response in the micoversion added or later, the parameter must be defined as required (`true`).

Microversion

If a parameter is available starting from a microversion, the microversion must be described by `min_version` in the parameter file. However, if the minimum microversion is the same as a microversion that the API itself is added, it is not necessary to describe the microversion.

For example:

```
aggregate_uuid:
  description: |
    The UUID of the host aggregate.
  in: body
  required: true
  type: string
  min_version: 2.41
```

This example describes that `aggregate_uuid` is available starting from microversion 2.41.

If a parameter is available up to a microversion, the microversion must be described by `max_version` in the parameter file.

For example:

```
security_group_rules:
  description: |
    The number of allowed rules for each security group.
  in: body
  required: false
  type: integer
  max_version: 2.35
```

This example describes that `security_group_rules` is available up to microversion 2.35 (and has been removed since microversion 2.36).

The order of parameters in the parameter file

The order of parameters in the parameter file has to be kept as follows:

1. By in type
    a. Header
    b. Path
    c. Query
    d. Body

2. Then alphabetical by name

Example

One or more examples should be provided for operations whose request and/or response contains a payload. The example should describe what the operation is attempting to do and provide a sample payload for the request and/or response as appropriate. Sample files should be created in the `doc/api_samples` directory and inlined by inclusion.

When an operation has no payload in the response, a suitable message should be included. For example:

```
There is no body content for the response of a successful DELETE query.
```

Examples for multiple microversions should be included in ascending microversion order.

4.9.3 Reference

- Verifying the Nova API Ref
- The description for Parameters whose values are null
- The definition of Optional parameter
- How to document your OpenStack API service

4.10 Blueprints, Specs and Priorities

Like most OpenStack projects, Nova uses blueprints and specifications (specs) to track new features, but not all blueprints require a spec. This document covers when a spec is needed.

Note: Novas specs live at: specs.openstack.org

4.10.1 Specs

A spec is needed for any feature that requires a design discussion. All features need a blueprint but not all blueprints require a spec.

If a new feature is straightforward enough that it doesnt need any design discussion, then no spec is required. In order to provide the sort of documentation that would otherwise be provided via a spec, the commit message should include a `DocImpact` flag and a thorough description of the feature from a user/operator perspective.

Guidelines for when a feature doesnt need a spec.

- Is the feature a single self contained change?
  - If the feature touches code all over the place, it probably should have a design discussion.
  - If the feature is big enough that it needs more than one commit, it probably should have a design discussion.
- Not an API change.
  - API changes always require a design discussion.

When a blueprint does not require a spec it still needs to be approved before the code which implements the blueprint is merged. Specless blueprints are discussed and potentially approved during the *Open Discussion* portion of the weekly nova IRC meeting. See trivial specifications for more details.

4.10.2 Project Priorities

- Pick several project priority themes, in the form of use cases, to help us prioritize work
  - Generate list of improvement blueprints based on the themes
  - Produce rough draft of list going into summit and finalize the list at the summit
  - Publish list of project priorities and look for volunteers to work on them
- Update spec template to include
  - Specific use cases
  - State if the spec is project priority or not
- Keep an up to date list of project priority blueprints that need code review in an etherpad.
- Consumers of project priority and project priority blueprint lists:
  - Reviewers looking for direction of where to spend their blueprint review time. If a large subset of nova-core doesnt use the project priorities it means the core team is not aligned properly and should revisit the list of project priorities
  - The blueprint approval team, to help find the right balance of blueprints
  - Contributors looking for something to work on
  - People looking for what they can expect in the next release

4.11 Code Review Guide for Nova

This is a very terse set of points for reviewers to consider when looking at nova code. These are things that are important for the continued smooth operation of Nova, but that tend to be carried as tribal knowledge instead of being written down. It is an attempt to boil down some of those things into nearly checklist format. Further explanation about why some of these things are important belongs elsewhere and should be linked from here.

4.11.1 Upgrade-Related Concerns

RPC API Versions

- If an RPC method is modified, the following needs to happen:
  - The manager-side (example: compute/manager) needs a version bump
  - **The manager-side method needs to tolerate older calls as well as** newer calls
  - Arguments can be added as long as they are optional. Arguments cannot be removed or changed in an incompatible way.
  - The RPC client code (example: compute/rpcapi.py) needs to be able to honor a pin for the older version (see self.client.can_send_version() calls). If we are pinned at 1.5, but the version requirement for a method is 1.7, we need to be able to formulate the call at version 1.5.
  - Methods can drop compatibility with older versions when we bump a major version.

- RPC methods can be deprecated by removing the client (example: compute/rpcapi.py) implementation. However, the manager method must continue to exist until the major version of the API is bumped.

Object Versions

- If a tracked attribute (i.e. listed in fields) or remotable method is added, or a method is changed, the object version must be bumped. Changes for methods follow the same rules as above for regular RPC methods. We have tests to try to catch these changes, which remind you to bump the version and then correct the version-hash in the tests.
- Field types cannot be changed. If absolutely required, create a new attribute and deprecate the old one. Ideally, support converting the old attribute to the new one with an obj_load_attr() handler. There are some exceptional cases where changing the type can be allowed, but care must be taken to ensure it does not affect the wireline API.
- New attributes should be removed from the primitive in obj_make_compatible() if the attribute was added after the target version.
- Remotable methods should not return unversioned structures wherever possible. They should return objects or simple values as the return types are not (and cannot) be checked by the hash tests.
- Remotable methods should not take complex structures as arguments. These cannot be verified by the hash tests, and thus are subject to drift. Either construct an object and pass that, or pass all the simple values required to make the call.
- Changes to an object as described above will cause a hash to change in TestObjectVersions. This is a reminder to the developer and the reviewer that the version needs to be bumped. There are times when we need to make a change to an object without bumping its version, but those cases are only where the hash logic detects a change that is not actually a compatibility issue and must be handled carefully.

Database Schema

- Changes to the database schema must generally be additive-only. This means you can add columns, but you cant drop or alter a column. We have some hacky tests to try to catch these things, but they are fragile. Extreme reviewer attention to non-online alterations to the DB schema will help us avoid disaster.
- Dropping things from the schema is a thing we need to be extremely careful about, making sure that the column has not been used (even present in one of our models) for at least a release.
- Data migrations must not be present in schema migrations. If data needs to be converted to another format, or moved from one place to another, then that must be done while the database server remains online. Generally, this can and should be hidden within the object layer so that an object can load from either the old or new location, and save to the new one.
- Multiple Cells v2 cells are supported started in the Pike release. As such, any online data migrations that move data from a cell database to the API database must be multi-cell aware.

4.11.2 REST API

When making a change to the nova API, we should always follow the API WG guidelines rather than going for local consistency. Developers and reviewers should read all of the guidelines, but they are very long. So here are some key points:

- Terms
  - `project` should be used in the REST API instead of `tenant`.

  - `server` should be used in the REST API instead of `instance`.
  - `compute` should be used in the REST API instead of `nova`.
- Naming Conventions
  - URL should not include underscores; use hyphens (-) instead.
  - The field names contained in a request/response body should use snake_case style, not CamelCase or Mixed_Case style.
- HTTP Response Codes
  - Synchronous resource creation: `201 Created`
  - Asynchronous resource creation: `202 Accepted`
  - Synchronous resource deletion: `204 No Content`
  - For all other successful operations: `200 OK`

4.11.3 Config Options

Location

The central place where all config options should reside is the `/nova/conf/` package. Options that are in named sections of `nova.conf`, such as `[serial_console]`, should be in their own module. Options that are in the `[DEFAULT]` section should be placed in modules that represent a natural grouping. For example, all of the options that affect the scheduler would be in the `scheduler.py` file, and all the networking options would be moved to `network.py`.

Implementation

A config option should be checked for:

- A short description which explains what it does. If it is a unit (e.g. timeouts or so) describe the unit which is used (seconds, megabyte, mebibyte,).
- A long description which explains the impact and scope. The operators should know the expected change in the behavior of Nova if they tweak this.
- Descriptions/Validations for the possible values.
  - If this is an option with numeric values (int, float), describe the edge cases (like the min value, max value, 0, -1).
  - If this is a DictOpt, describe the allowed keys.
  - If this is a StrOpt, list any possible regex validations, or provide a list of acceptable and/or prohibited values.

Previously used sections which explained which services consume a specific config option and which options are related to each other got dropped because they are too hard to maintain: http://lists.openstack.org/pipermail/openstack-dev/2016-May/095538.html

4.11.4 Third Party Tests

Any change that is not tested well by the Jenkins check jobs must have a recent +1 vote from an appropriate third party test (or tests) on the latest patchset, before a core reviewer is allowed to make a +2 vote.

Virt drivers

At a minimum, we must ensure that any technology specific code has a +1 from the relevant third party test, on the latest patchset, before a +2 vote can be applied. Specifically, changes to nova/virt/driver/<NNNN> need a +1 vote from the respective third party CI. For example, if you change something in the XenAPI virt driver, you must wait for a +1 from the XenServer CI on the latest patchset, before you can give that patch set a +2 vote.

This is important to ensure:

- We keep those drivers stable
- We dont break that third party CI

Notes

Please note:

- Long term, we should ensure that any patch a third party CI is allowed to vote on, can be blocked from merging by that third party CI. But we need a lot more work to make something like that feasible, hence the proposed compromise.
- While its possible to break a virt driver CI system by changing code that is outside the virt drivers, this policy is not focusing on fixing that. A third party test failure should always be investigated, but the failure of a third party test to report in a timely manner should not block others.
- We are only talking about the testing of in-tree code. Please note the only public API is our REST API, see: *Development policies*

4.11.5 Should I run the experimental queue jobs on this change?

Because we cant run all CI jobs in the check and gate pipelines, some jobs can be executed on demand, thanks to the experimental pipeline. To run the experimental jobs, you need to comment your Gerrit review with check experimental.

The experimental jobs aim to test specific features, such as LXC containers or DVR with multiple nodes. Also, it might be useful to run them when we want to test backward compatibility with tools that deploy OpenStack outside Devstack (e.g. TripleO, etc). They can produce a non-voting feedback of whether the system continues to work when we deprecate or remove some options or features in Nova.

The experimental queue can also be used to test that new CI jobs are correct before making them voting.

4.11.6 Database Schema

- Use the `utf8` charset only where necessary. Some string fields, such as hex-stringified UUID values, MD5 fingerprints, SHA1 hashes or base64-encoded data, are always interpreted using ASCII encoding. A hex-stringified UUID value in `latin1` is 1/3 the size of the same field in `utf8`, impacting performance without bringing any benefit. If there are no string type columns in the table, or the string type columns contain **only** the data described above, then stick with `latin1`.

4.11.7 Microversion API

If a new microversion API is added, the following needs to happen:

- A new patch for the microversion API change in python-novaclient side should be submitted before the microversion change in Nova is merged. See Adding support for a new microversion in python-novaclient for more details.

- If the microversion changes the response schema, a new schema and test for the microversion must be added to Tempest. The microversion change in Nova should not be merged until the Tempest test is submitted and at least passing; it does not need to be merged yet as long as it is testing the Nova change via Depends-On. The Nova microversion change commit message should reference the Change-Id of the Tempest test for reviewers to identify it.

4.11.8 Notifications

- Every new notification type shall use the new versioned notification infrastructure documented in *Notifications in Nova*

4.11.9 Release Notes

A release note is required on changes that have upgrade impact, security impact, introduce a new feature, fix Critical bugs, or fix long-standing bugs with high importance. See *Release Notes* for details on how to create a release note, each available section and the type of content required.

4.12 Documentation Guidelines

These are some basic guidelines for contributing to documentation in nova.

4.12.1 Review Guidelines

Documentation-only patches differ from code patches in a few ways.

- They are often written by users / operators that arent plugged into daily cycles of nova or on IRC
- Outstanding patches are far more likely to trigger merge conflict in Git than code patches
- There may be wide variation on points of view of what the best or clearest way is to say a thing

This all can lead to a large number of practical documentation improvements stalling out because the author submitted the fix, and does not have the time to merge conflict chase or is used to the Gerrit follow up model.

As such, documentation patches should be evaluated in the basic context of does this make things better than the current tree. Patches are cheap, it can always be further enhanced in future patches.

Typo / trivial documentation only fixes should get approved with a single +2.

4.12.2 How users consume docs

The current primary target for all documentation in nova is the web. While it is theoretically possible to generate PDF versions of the content, the tree is not currently well structured for that, and its not clear there is an audience for that.

The main nova docs tree `doc/source` is published per release, so there will be copies of all of this as both the `latest` URL (which is master), and for every stable release (e.g. `pike`).

Note: This raises an interesting and unexplored question about whether we want all of `doc/source` published with stable branches that will be stale and unimproved as we address content in `latest`.

The `api-ref` and `api-guide` publish only from master to a single site on *docs.openstack.org*. As such, they are effectively branchless.

4.12.3 Guidelines for consumable docs

- Give users context before following a link

  Most users exploring our documentation will be trying to learn about our software. Entry and subpages that provide links to in depth topics need to provide some basic context about why someone would need to know about a `filter scheduler` before following the link named filter scheduler.

  Providing these summaries helps the new consumer come up to speed more quickly.

- Doc moves require `.htaccess` redirects

  If a file is moved in a documentation source tree, we should be aware that it might be linked from external sources, and is now a `404 Not Found` error for real users.

  All doc moves should include an entry in `doc/source/_extra/.htaccess` to redirect from the old page to the new page.

- Images are good, please provide source

  An image is worth a 1000 words, but can go stale pretty quickly. We ideally want `png` files for display on the web, but thats a non modifiable format. For any new diagram contributions we should also get some kind of source format (`svg` is ideal as it can be modified with open tools) along with `png` formats.

4.12.4 Long Term TODOs

- Sort out our toctree / sidebar navigation

  During the bulk import of the install, admin, config guides we started with a unified toctree, which was a ton of entries, and made the navigation sidebar in Nova incredibly confusing. The short term fix was to just make that almost entirely hidden and rely on structured landing and sub pages.

  Long term it would be good to reconcile the toctree / sidebar into something that feels coherent.

4.13 Evacuate vs Rebuild

The evacuate API and rebuild API are commonly confused in nova because the internal conductor code and compute code use the same methods called `rebuild_instance`. This document explains some of the differences in what happens between an evacuate and rebuild operation.

4.13.1 High level

Evacuate is an operation performed by an administrator when a compute service or host is encountering some problem, goes down and needs to be fenced from the network. The servers that were running on that compute host can be rebuilt on a **different** host using the **same** image. If the source and destination hosts are running on shared storage then the root disk image of the servers can be retained otherwise the root disk image (if not using a volume-backed server) will be lost. This is one example of why it is important to attach data volumes to a server to store application data and leave the root disk for the operating system since data volumes will be re-attached to the server as part of the evacuate process.

Rebuild is an operation which can be performed by a non-administrative owner of the server (the user) performed on the **same** compute host to change certain aspects of the server, most notably using a **different** image. Note that the image does not *have* to change and in the case of volume-backed servers the image currently cannot change. Other attributes of the server can be changed as well such as `key_name` and `user_data`. See the rebuild API reference for full usage details. When a user rebuilds a server they want to change it which requires re-spawning the guest in the

hypervisor but retain the UUID, volumes and ports attached to the server. For a non-volume-backed server the root disk image is rebuilt.

4.13.2 Scheduling

Evacuate always schedules the server to another host and rebuild always occurs on the same host.

Note that when rebuilding with a different image, the request is run through the scheduler to ensure the new image is still valid for the current compute host.

4.13.3 Image

As noted above, the image that the server uses during an evacuate operation does not change. The image used to rebuild a server *may* change but does not have to and in the case of volume-backed servers *cannot* change.

4.13.4 Resource claims

The compute service `ResourceTracker` has a claims operation which is used to ensure resources are available before building a server on the host. The scheduler performs the initial filtering of hosts to ensure a server can be built on a given host and the compute claim is essentially meant as a secondary check to prevent races when the scheduler has out of date information or when there are concurrent build requests going to the same host.

During an evacuate operation there is a rebuild claim since the server is being re-built on a different host.

During a rebuild operation, since the flavor does not change, there is no claim made since the host does not change.

4.13.5 Allocations

Since the 16.0.0 (Pike) release, the scheduler uses the placement service to filter compute nodes (resource providers) based on information in the flavor and image used to build the server. Once the scheduler runs through its filters and weighers and picks a host, resource class allocations are atomically consumed in placement with the server as the consumer.

During an evacuate operation, the allocations held by the server consumer against the source compute node resource provider are left intact since the source compute service is down. Note that migration-based allocations, which were introduced in the 17.0.0 (Queens) release, do not apply to evacuate operations but only resize, cold migrate and live migrate. So once a server is successfully evacuated to a different host, the placement service will track allocations for that server against both the source and destination compute node resource providers. If the source compute service is restarted after being evacuated and fixed, the compute service will delete the old allocations held by the evacuated servers.

During a rebuild operation, since neither the host nor flavor changes, the server allocations remain intact.

4.14 API Microversions

4.14.1 Background

Nova uses a framework we call API Microversions for allowing changes to the API while preserving backward compatibility. The basic idea is that a user has to explicitly ask for their request to be treated with a particular version of the API. So breaking changes can be added to the API without breaking users who dont specifically ask for it. This is done with an HTTP header `OpenStack-API-Version` which has as its value a string containing the name of the

service, `compute`, and a monotonically increasing semantic version number starting from `2.1`. The full form of the header takes the form:

```
OpenStack-API-Version: compute 2.1
```

If a user makes a request without specifying a version, they will get the `DEFAULT_API_VERSION` as defined in `nova/api/openstack/wsgi.py`. This value is currently `2.1` and is expected to remain so for quite a long time.

There is a special value `latest` which can be specified, which will allow a client to always receive the most recent version of API responses from the server.

Warning: The `latest` value is mostly meant for integration testing and would be dangerous to rely on in client code since Nova microversions are not following semver and therefore backward compatibility is not guaranteed. Clients, like python-novaclient, should always require a specific microversion but limit what is acceptable to the version range that it understands at the time.

Warning: To maintain compatibility, an earlier form of the microversion header is acceptable. It takes the form:

```
X-OpenStack-Nova-API-Version: 2.1
```

This form will continue to be supported until the `DEFAULT_API_VERSION` is raised to version `2.27` or higher.

Clients accessing deployments of the Nova API which are not yet providing microversion `2.27` must use the older form.

For full details please read the Kilo spec for microversions and Microversion Specification.

4.14.2 When do I need a new Microversion?

A microversion is needed when the contract to the user is changed. The user contract covers many kinds of information such as:

- the Request
  - the list of resource urls which exist on the server

    Example: adding a new servers/{ID}/foo which didnt exist in a previous version of the code
  - the list of query parameters that are valid on urls

    Example: adding a new parameter `is_yellow` servers/{ID}?is_yellow=True
  - the list of query parameter values for non free form fields

    Example: parameter filter_by takes a small set of constants/enums A, B, C. Adding support for new enum D.
  - new headers accepted on a request
  - the list of attributes and data structures accepted.

    Example: adding a new attribute locked: True/False to the request body

    However, the attribute `os.scheduler_hints` of the create a server API is an exception to this. A new scheduler which adds a new attribute to `os:scheduler_hints` doesnt require a new microversion, because available schedulers depend on cloud environments, and we accept customized schedulers as a rule.

- the Response
  - the list of attributes and data structures returned

    Example: adding a new attribute locked: True/False to the output of servers/{ID}
  - the allowed values of non free form fields

    Example: adding a new allowed `status` to servers/{ID}
  - the list of status codes allowed for a particular request

    Example: an API previously could return 200, 400, 403, 404 and the change would make the API now also be allowed to return 409.

    See[2] for the 400, 403, 404 and 415 cases.
  - changing a status code on a particular response

    Example: changing the return code of an API from 501 to 400.

    Note: Fixing a bug so that a 400+ code is returned rather than a 500 or 503 does not require a microversion change. Its assumed that clients are not expected to handle a 500 or 503 response and therefore should not need to opt-in to microversion changes that fixes a 500 or 503 response from happening. According to the OpenStack API Working Group, a **500 Internal Server Error** should **not** be returned to the user for failures due to user error that can be fixed by changing the request on the client side. See[1].

  - new headers returned on a response

The following flow chart attempts to walk through the process of do we need a microversion.

[2] The exception to not needing a microversion when returning a previously unspecified error code is the 400, 403, 404 and 415 cases. This is considered OK to return even if previously unspecified in the code since its implied given keystone authentication can fail with a 403 and API validation can fail with a 400 for invalid json request body. Request to url/resource that does not exist always fails with 404. Invalid content types are handled before API methods are called which results in a 415.

Note: When in doubt about whether or not a microversion is required for changing an error response code, consult the Nova API subteam.

---

[1] When fixing 500 errors that previously caused stack traces, try to map the new error into the existing set of errors that API call could previously return (400 if nothing else is appropriate). Changing the set of allowed status codes from a request is changing the contract, and should be part of a microversion (except in[2]).

The reason why we are so strict on contract is that wed like application writers to be able to know, for sure, what the contract is at every microversion in Nova. If they do not, they will need to write conditional code in their application to handle ambiguities.

When in doubt, consider application authors. If it would work with no client side changes on both Nova versions, you probably dont need a microversion. If, on the other hand, there is any ambiguity, a microversion is probably needed.

Footnotes

4.14.3 When a microversion is not needed

A microversion is not needed in the following situation:

- the response
  - Changing the error message without changing the response code does not require a new microversion.
  - Removing an inapplicable HTTP header, for example, suppose the Retry-After HTTP header is being returned with a 4xx code. This header should only be returned with a 503 or 3xx response, so it may be removed without bumping the microversion.
  - An obvious regression bug in an admin-only API where the bug can still be fixed upstream on active

stable branches. Admin-only APIs are less of a concern for interoperability and generally a regression in behavior can be dealt with as a bug fix when the documentation clearly shows the API behavior was unexpectedly regressed. See[3] for an example. Intentional behavior changes to an admin-only API *do* require a microversion, like the *2.53 microversion* for example.

Footnotes

4.14.4 In Code

In `nova/api/openstack/wsgi.py` we define an `@api_version` decorator which is intended to be used on top-level Controller methods. It is not appropriate for lower-level methods. Some examples:

Adding a new API method

In the controller class:

```
@wsgi.Controller.api_version("2.4")
def my_api_method(self, req, id):
    ....
```

This method would only be available if the caller had specified an `OpenStack-API-Version` of >= `2.4`. If they had specified a lower version (or not specified it and received the default of `2.1`) the server would respond with `HTTP/404`.

Removing an API method

In the controller class:

```
@wsgi.Controller.api_version("2.1", "2.4")
def my_api_method(self, req, id):
    ....
```

This method would only be available if the caller had specified an `OpenStack-API-Version` of <= `2.4`. If `2.5` or later is specified the server will respond with `HTTP/404`.

Changing a methods behavior

In the controller class:

```
@wsgi.Controller.api_version("2.1", "2.3")
def my_api_method(self, req, id):
    .... method_1 ...

@wsgi.Controller.api_version("2.4")  # noqa
def my_api_method(self, req, id):
    .... method_2 ...
```

If a caller specified `2.1`, `2.2` or `2.3` (or received the default of `2.1`) they would see the result from `method_1`, `2.4` or later `method_2`.

It is vital that the two methods have the same name, so the second of them will need `# noqa` to avoid failing flake8s `F811` rule. The two methods may be different in any kind of semantics (schema validation, return values, response codes, etc)

[3] https://review.opendev.org/#/c/523194/

A change in schema only

If there is no change to the method, only to the schema that is used for validation, you can add a version range to the `validation.schema` decorator:

```
@wsgi.Controller.api_version("2.1")
@validation.schema(dummy_schema.dummy, "2.3", "2.8")
@validation.schema(dummy_schema.dummy2, "2.9")
def update(self, req, id, body):
    ....
```

This method will be available from version `2.1`, validated according to `dummy_schema.dummy` from `2.3` to `2.8`, and validated according to `dummy_schema.dummy2` from `2.9` onward.

When not using decorators

When you dont want to use the `@api_version` decorator on a method or you want to change behavior within a method (say it leads to simpler or simply a lot less code) you can directly test for the requested version with a method as long as you have access to the api request object (commonly called `req`). Every API method has an api_version_request object attached to the req object and that can be used to modify behavior based on its value:

```
def index(self, req):
    <common code>

    req_version = req.api_version_request
    req1_min = api_version_request.APIVersionRequest("2.1")
    req1_max = api_version_request.APIVersionRequest("2.5")
    req2_min = api_version_request.APIVersionRequest("2.6")
    req2_max = api_version_request.APIVersionRequest("2.10")

    if req_version.matches(req1_min, req1_max):
        ....stuff....
    elif req_version.matches(req2min, req2_max):
        ....other stuff....
    elif req_version > api_version_request.APIVersionRequest("2.10"):
        ....more stuff.....

    <common code>
```

The first argument to the matches method is the minimum acceptable version and the second is maximum acceptable version. A specified version can be null:

```
null_version = APIVersionRequest()
```

If the minimum version specified is null then there is no restriction on the minimum version, and likewise if the maximum version is null there is no restriction the maximum version. Alternatively a one sided comparison can be used as in the example above.

4.14.5 Other necessary changes

If you are adding a patch which adds a new microversion, it is necessary to add changes to other places which describe your change:

- Update `REST_API_VERSION_HISTORY` in `nova/api/openstack/api_version_request.py`
- Update `_MAX_API_VERSION` in `nova/api/openstack/api_version_request.py`

- Add a verbose description to `nova/api/openstack/compute/rest_api_version_history.rst`.
- Add a *release note* with a `features` section announcing the new or changed feature and the microversion.
- Update the expected versions in affected tests, for example in `nova/tests/unit/api/openstack/compute/test_versions.py`.
- Update the get versions api sample file: `doc/api_samples/versions/versions-get-resp.json` and `doc/api_samples/versions/v21-version-get-resp.json`.
- Make a new commit to python-novaclient and update corresponding files to enable the newly added microversion API. See Adding support for a new microversion in python-novaclient for more details.
- If the microversion changes the response schema, a new schema and test for the microversion must be added to Tempest.
- Update the API Reference documentation as appropriate. The source is located under *api-ref/source/*.

4.14.6 Allocating a microversion

If you are adding a patch which adds a new microversion, it is necessary to allocate the next microversion number. Except under extremely unusual circumstances and this would have been mentioned in the nova spec for the change, the minor number of `_MAX_API_VERSION` will be incremented. This will also be the new microversion number for the API change.

It is possible that multiple microversion patches would be proposed in parallel and the microversions would conflict between patches. This will cause a merge conflict. We dont reserve a microversion for each patch in advance as we dont know the final merge order. Developers may need over time to rebase their patch calculating a new version number as above based on the updated value of `_MAX_API_VERSION`.

4.14.7 Testing Microversioned API Methods

Testing a microversioned API method is very similar to a normal controller method test, you just need to add the `OpenStack-API-Version` header, for example:

```
req = fakes.HTTPRequest.blank('/testable/url/endpoint')
req.headers = {'OpenStack-API-Version': 'compute 2.28'}
req.api_version_request = api_version.APIVersionRequest('2.6')

controller = controller.TestableController()

res = controller.index(req)
... assertions about the response ...
```

For many examples of testing, the canonical examples are in `nova/tests/unit/api/openstack/compute/test_microversions.py`.

4.15 Development policies

4.15.1 Out Of Tree Support

While nova has many entrypoints and other places in the code that allow for wiring in out of tree code, upstream doesnt actively make any guarantees about these extensibility points; we dont support them, make any guarantees about compatibility, stability, etc.

Furthermore, hooks and extension points in the code impede efforts in Nova to support interoperability between OpenStack clouds. Therefore an effort is being made to systematically deprecate and remove hooks, extension points, and classloading of managers and other services.

4.15.2 Public Contractual APIs

Although nova has many internal APIs, they are not all public contractual APIs. Below is a link of our public contractual APIs:

- https://docs.openstack.org/api-ref/compute/

Anything not in this list is considered private, not to be used outside of nova, and should not be considered stable.

4.15.3 REST APIs

Follow the guidelines set in: https://wiki.openstack.org/wiki/APIChangeGuidelines

The canonical source for REST API behavior is the code *not* documentation. Documentation is manually generated after the code by folks looking at the code and writing up what they think it does, and it is very easy to get this wrong.

This policy is in place to prevent us from making backwards incompatible changes to REST APIs.

4.15.4 Patches and Reviews

Merging a patch requires a non-trivial amount of reviewer resources. As a patch author, you should try to offset the reviewer resources spent on your patch by reviewing other patches. If no one does this, the review team (cores and otherwise) become spread too thin.

For review guidelines see: https://docs.openstack.org/infra/manual/developers.html#peer-review

4.15.5 Reverts for Retrospective Vetos

Sometimes our simple 2 +2s approval policy will result in errors. These errors might be a bug that was missed, or equally importantly, it might be that other cores feel that there is a need for more discussion on the implementation of a given piece of code.

Rather than an enforced time-based solution - for example, a patch couldnt be merged until it has been up for review for 3 days - we have chosen an honor-based system where core reviewers would not approve potentially contentious patches until the proposal had been sufficiently socialized and everyone had a chance to raise any concerns.

Recognising that mistakes can happen, we also have a policy where contentious patches which were quickly approved should be reverted so that the discussion around the proposal can continue as if the patch had never been merged in the first place. In such a situation, the procedure is:

0. The commit to be reverted must not have been released.
1. The core team member who has a -2 worthy objection should propose a revert, stating the specific concerns that they feel need addressing.
2. Any subsequent patches depending on the to-be-reverted patch may need to be reverted also.
3. Other core team members should quickly approve the revert. No detailed debate should be needed at this point. A -2 vote on a revert is strongly discouraged, because it effectively blocks the right of cores approving the revert from -2 voting on the original patch.
4. The original patch submitter should re-submit the change, with a reference to the original patch and the revert.

5. The original reviewers of the patch should restore their votes and attempt to summarize their previous reasons for their votes.
6. The patch should not be re-approved until the concerns of the people proposing the revert are worked through. A mailing list discussion or design spec might be the best way to achieve this.

4.15.6 Metrics Gathering

Nova currently has a monitor plugin to gather CPU metrics on compute nodes. This feeds into the MetricsFilter and MetricsWeigher in the scheduler. The CPU metrics monitor is only implemented for the libvirt compute driver. External projects like Ceilometer and Watcher consume these metrics.

Over time people have tried to add new monitor plugins for things like memory bandwidth. There have also been attempts to expose these monitors over CLI, the REST API, and notifications.

At the Newton midcycle it was decided that Nova does a poor job as a metrics gathering tool, especially as its incomplete, not tested, and there are numerous other tools available to get this information as their primary function.

Therefore, there is a freeze on adding new metrics monitoring plugins which also includes exposing existing monitored metrics outside of Nova, like with the nova-manage CLI, the REST API, or the notification bus. Long-term, metrics gathering will likely be deprecated within Nova. Since there is not yet a clear replacement, the deprecation is open-ended, but serves as a signal that new deployments should not rely on the metrics that Nova gathers and should instead focus their efforts on alternative solutions for placement.

4.16 Release Notes

4.16.1 What is reno ?

Nova uses reno for providing release notes in-tree. That means that a patch can include a *reno file* or a series can have a follow-on change containing that file explaining what the impact is.

A *reno file* is a YAML file written in the `releasenotes/notes` tree which is generated using the *reno* tool this way:

```
$ tox -e venv -- reno new <name-your-file>
```

where usually `<name-your-file>` can be `bp-<blueprint_name>` for a blueprint or `bug-XXXXXX` for a bugfix.

Refer to the reno documentation for more information.

4.16.2 When a release note is needed

A release note is required anytime a reno section is needed. Below are some examples for each section. Any sections that would be blank should be left out of the note file entirely. If no section is needed, then you know you dont need to provide a release note :-)

- **`upgrade`**
  - The patch has an UpgradeImpact tag
  - A DB change needs some deployer modification (like a migration)
  - A configuration option change (deprecation, removal or modified default)

  - some specific changes that have a DocImpact tag but require further action from an deployer perspective
  - any patch that requires an action from the deployer in general
- **security**
  - If the patch fixes a known vulnerability
- **features**
  - If the patch has an APIImpact tag
  - For nova-manage and python-novaclient changes, if it adds or changes a new command, including adding new options to existing commands
  - not all blueprints in general, just the ones impacting a *Development policies*
  - a new virt driver is provided or an existing driver impacts the *HypervisorSupportMatrix*
- **critical**
  - Bugfixes categorized as Critical in Launchpad *impacting users*
- **fixes**
  - No clear definition of such bugfixes. Hairy long-standing bugs with high importance that have been fixed are good candidates though.

Three sections are left intentionally unexplained (`prelude`, `issues` and `other`). Those are targeted to be filled in close to the release time for providing details about the soon-ish release. Dont use them unless you know exactly what you are doing.

4.17 Test Strategy

A key part of the four opens is ensuring the OpenStack delivers well-tested and usable software. For more details see: http://docs.openstack.org/project-team-guide/introduction.html#the-four-opens

Experience has shown that untested features are frequently broken, in part due to the velocity of upstream changes. As we aim to ensure we keep all features working across upgrades, we must aim to test all features.

4.17.1 Reporting Test Coverage

For details on plans to report the current test coverage, refer to *Feature Classification*.

4.17.2 Running tests and reporting results

Voting in Gerrit

On every review in gerrit, check tests are run on very patch set, and are able to report a +1 or -1 vote. For more details, please see: http://docs.openstack.org/infra/manual/developers.html#automated-testing

Before merging any code, there is an integrate gate test queue, to ensure master is always passing all tests. For more details, please see: http://docs.openstack.org/infra/zuul/user/gating.html

Infra vs Third-Party

Tests that use fully open source components are generally run by the OpenStack Infra teams. Test setups that use non-open technology must be run outside of that infrastructure, but should still report their results upstream.

For more details, please see: http://docs.openstack.org/infra/system-config/third_party.html

Ad-hoc testing

It is particularly common for people to run ad-hoc tests on each released milestone, such as RC1, to stop regressions. While these efforts can help stabilize the release, as a community we have a much stronger preference for continuous integration testing. Partly this is because we encourage users to deploy master, and we generally have to assume that any upstream commit may already been deployed in production.

4.17.3 Types of tests

Unit tests

Unit tests help document and enforce the contract for each component. Without good unit test coverage it is hard to continue to quickly evolve the codebase. The correct level of unit test coverage is very subjective, and as such we are not aiming for a particular percentage of coverage, rather we are aiming for good coverage. Generally, every code change should have a related unit test: https://opendev.org/openstack/nova/src/branch/master/HACKING.rst#creating-unit-tests

Integration tests

Today, our integration tests involve running the Tempest test suite on a variety of Nova deployment scenarios. The integration job setup is defined in the `.zuul.yaml` file in the root of the nova repository. Jobs are restricted by queue:

- `check`: jobs in this queue automatically run on all proposed changes even with non-voting jobs
- `gate`: jobs in this queue automatically run on all approved changes (voting jobs only)
- `experimental`: jobs in this queue are non-voting and run on-demand by leaving a review comment on the change of check experimental

In addition, we have third parties running the tests on their preferred Nova deployment scenario.

Functional tests

Nova has a set of in-tree functional tests that focus on things that are out of scope for tempest testing and unit testing. Tempest tests run against a full live OpenStack deployment, generally deployed using devstack. At the other extreme, unit tests typically use mock to test a unit of code in isolation. Functional tests dont run an entire stack, they are isolated to nova code, and have no reliance on external services. They do have a WSGI app, nova services and a database, with minimal stubbing of nova internals.

Interoperability tests

The DefCore committee maintains a list that contains a subset of Tempest tests. These are used to verify if a particular Nova deployments API responds as expected. For more details, see: https://opendev.org/openstack/interop

4.18 Testing NUMA related hardware setup with libvirt

This page describes how to test the libvirt drivers handling of the NUMA placement, large page allocation and CPU pinning features. It relies on setting up a virtual machine as the test environment and requires support for nested virtualization since plain QEMU is not sufficiently functional. The virtual machine will itself be given NUMA topology, so it can then act as a virtual host for testing purposes.

4.18.1 Provisioning a virtual machine for testing

The entire test process will take place inside a large virtual machine running Fedora 24. The instructions should work for any other Linux distribution which includes libvirt >= 1.2.9 and QEMU >= 2.1.2

The tests will require support for nested KVM, which is not enabled by default on hypervisor hosts. It must be explicitly turned on in the host when loading the kvm-intel/kvm-amd kernel modules.

On Intel hosts verify it with

```
# cat /sys/module/kvm_intel/parameters/nested
N

# rmmod kvm-intel
# echo "options kvm-intel nested=y" > /etc/modprobe.d/dist.conf
# modprobe kvm-intel

# cat /sys/module/kvm_intel/parameters/nested
Y
```

While on AMD hosts verify it with

```
# cat /sys/module/kvm_amd/parameters/nested
0

# rmmod kvm-amd
# echo "options kvm-amd nested=1" > /etc/modprobe.d/dist.conf
# modprobe kvm-amd

# cat /sys/module/kvm_amd/parameters/nested
1
```

The virt-install command below shows how to provision a basic Fedora 24 x86_64 guest with 8 virtual CPUs, 8 GB of RAM and 20 GB of disk space:

```
# cd /var/lib/libvirt/images
# wget https://download.fedoraproject.org/pub/fedora/linux/releases/24/Server/x86_64/
↪iso/Fedora-Server-netinst-x86_64-24-1.2.iso

# virt-install \
   --name f24x86_64 \
   --ram 8000 \
   --vcpus 8 \
   --file /var/lib/libvirt/images/f24x86_64.img \
   --file-size 20
   --cdrom /var/lib/libvirt/images/Fedora-Server-netinst-x86_64-24-1.2.iso \
   --os-variant fedora23
```

When the virt-viewer application displays the installer, follow the defaults for the installation with a couple of exceptions:

- The automatic disk partition setup can be optionally tweaked to reduce the swap space allocated. No more than 500MB is required, freeing up an extra 1.5 GB for the root disk
- Select Minimal install when asked for the installation type since a desktop environment is not required
- When creating a user account be sure to select the option Make this user administrator so it gets sudo rights

Once the installation process has completed, the virtual machine will reboot into the final operating system. It is now ready to deploy an OpenStack development environment.

4.18.2 Setting up a devstack environment

For later ease of use, copy your SSH public key into the virtual machine:

```
# ssh-copy-id  <IP of VM>
```

Now login to the virtual machine:

```
# ssh <IP of VM>
```

The Fedora minimal install does not contain git. Install git and clone the devstack repo:

```
$ sudo dnf install git
$ git clone https://opendev.org/openstack/devstack
$ cd devstack
```

At this point a fairly standard devstack setup can be done with one exception: we should enable the `NUMATopologyFilter` filter, which we will use later. For example:

```
$ cat >>local.conf <<EOF
[[local|localrc]]
DATA_DIR=$DEST/data
SERVICE_DIR=$DEST/status

LOGFILE=$DATA_DIR/logs/stack.log
VERBOSE=True

disable_service n-net
enable_service neutron q-svc q-dhcp q-l3 q-meta q-agt

MYSQL_PASSWORD=123456
DATABASE_PASSWORD=123456
SERVICE_TOKEN=123456
SERVICE_PASSWORD=123456
ADMIN_PASSWORD=123456
RABBIT_PASSWORD=123456

[[post-config|$NOVA_CONF]]
[DEFAULT]
firewall_driver=nova.virt.firewall.NoopFirewallDriver

[filter_scheduler]
enabled_filters=ComputeFilter,AvailabilityZoneFilter,ComputeCapabilitiesFilter,
↪ImagePropertiesFilter,PciPassthroughFilter,NUMATopologyFilter
EOF
$ FORCE=yes ./stack.sh
```

Unfortunately while devstack starts various system services and changes various system settings it doesnt make the changes persistent. Fix that now to avoid later surprises after reboots:

```
$ sudo systemctl enable mariadb.service
$ sudo systemctl enable rabbitmq-server.service
$ sudo systemctl enable httpd.service

$ sudo vi /etc/sysconfig/selinux
SELINUX=permissive
```

4.18.3 Testing basis non-NUMA usage

First to confirm weve not done anything unusual to the traditional operation of nova libvirt guests boot a tiny instance:

```
$ . openrc admin
$ openstack server create --image cirros-0.3.4-x86_64-uec --flavor m1.tiny \
    cirros1
```

The host will be reporting NUMA topology, but there should only be a single NUMA cell this point. We can validate this by querying the nova database. For example (with object versioning fields removed):

```
$ mysql -u root -p123456 nova
MariaDB [nova]> select numa_topology from compute_nodes;
+--------------------------------------------------------------------------------+
| numa_topology                                                                  |
+--------------------------------------------------------------------------------+
| {
|     "nova_object.name": "NUMATopology",
|     "nova_object.data": {
|         "cells": [{
|                 "nova_object.name": "NUMACell",
|                 "nova_object.data": {
|                     "cpu_usage": 0,
|                     "memory_usage": 0,
|                     "cpuset": [0, 1, 2, 3, 4, 5, 6, 7],
|                     "pinned_cpus": [],
|                     "siblings": [],
|                     "memory": 7793,
|                     "mempages": [
|                         {
|                             "nova_object.name": "NUMAPagesTopology",
|                             "nova_object.data": {
|                                 "used": 0,
|                                 "total": 987430,
|                                 "reserved":0,
|                                 "size_kb": 4
|                             },
|                         },
|                         {
|                             "nova_object.name": "NUMAPagesTopology",
|                             "nova_object.data": {
|                                 "used": 0,
|                                 "total": 0,
|                                 "reserved":0,
|                                 "size_kb": 2048
|                             },
```

(continues on next page)

(continued from previous page)

```
|                             },
|                             {
|                                 "nova_object.name": "NUMAPagesTopology",
|                                 "nova_object.data": {
|                                     "used": 0,
|                                     "total": 0,
|                                     "reserved": 0,
|                                     "size_kb": 1048576
|                                 },
|                             }
|                         ],
|                         "id": 0
|                     },
|                 },
|             ]
|         },
| }
+--------------------------------------------------------------------------------+
```

Meanwhile, the guest instance should not have any NUMA configuration recorded:

```
MariaDB [nova]> select numa_topology from instance_extra;
+---------------+
| numa_topology |
+---------------+
| NULL          |
+---------------+
```

4.18.4 Reconfiguring the test instance to have NUMA topology

Now that devstack is proved operational, it is time to configure some NUMA topology for the test VM, so that it can be used to verify the OpenStack NUMA support. To do the changes, the VM instance that is running devstack must be shut down:

```
$ sudo shutdown -h now
```

And now back on the physical host edit the guest config as root:

```
$ sudo virsh edit f21x86_64
```

The first thing is to change the *<cpu>* block to do passthrough of the host CPU. In particular this exposes the SVM or VMX feature bits to the guest so that Nested KVM can work. At the same time we want to define the NUMA topology of the guest. To make things interesting were going to give the guest an asymmetric topology with 4 CPUS and 4 GBs of RAM in the first NUMA node and 2 CPUs and 2 GB of RAM in the second and third NUMA nodes. So modify the guest XML to include the following CPU XML:

```
<cpu mode='host-passthrough'>
  <numa>
    <cell id='0' cpus='0-3' memory='4096000'/>
    <cell id='1' cpus='4-5' memory='2048000'/>
    <cell id='2' cpus='6-7' memory='2048000'/>
  </numa>
</cpu>
```

Now start the guest again:

```
# virsh start f24x86_64
```

and login back in:

```
# ssh <IP of VM>
```

Before starting OpenStack services again, it is necessary to explicitly set the libvirt virtualization type to KVM, so that guests can take advantage of nested KVM:

```
$ sudo sed -i 's/virt_type = qemu/virt_type = kvm/g' /etc/nova/nova.conf
```

With that done, OpenStack can be started again:

```
$ cd devstack
$ ./stack.sh
```

The first thing is to check that the compute node picked up the new NUMA topology setup for the guest:

```
$ mysql -u root -p123456 nova
MariaDB [nova]> select numa_topology from compute_nodes;
+--------------------------------------------------------------------------------+
| numa_topology                                                                  |
+--------------------------------------------------------------------------------+
| {
|     "nova_object.name": "NUMATopology",
|     "nova_object.data": {
|         "cells": [
|             {
|                 "nova_object.name": "NUMACell",
|                 "nova_object.data": {
|                     "cpu_usage": 0,
|                     "memory_usage": 0,
|                     "cpuset": [0, 1, 2, 3],
|                     "pinned_cpus": [],
|                     "siblings": [],
|                     "memory": 3856,
|                     "mempages": [
|                         {
|                             "nova_object.name": "NUMAPagesTopology",
|                             "nova_object.data": {
|                                 "used": 0,
|                                 "total": 987231,
|                                 "reserved": 0,
|                                 "size_kb": 4
|                             },
|                         },
|                         {
|                             "nova_object.name": "NUMAPagesTopology",
|                             "nova_object.data": {
|                                 "used": 0,
|                                 "total": 0,
|                                 "reserved": 0,
|                                 "size_kb": 2048
|                             },
|                         },
|                         {
|                             "nova_object.name": "NUMAPagesTopology",
```

(continues on next page)

(continued from previous page)

```
|                                 "nova_object.data": {
|                                     "used": 0,
|                                     "total": 0,
|                                     "reserved": 0,
|                                     "size_kb": 1048576
|                                 },
|                             }
|                         ],
|                         "id": 0
|                     },
|                 },
|                 {
|                     "nova_object.name": "NUMACell",
|                     "nova_object.data": {
|                         "cpu_usage": 0,
|                         "memory_usage": 0,
|                         "cpuset": [4, 5],
|                         "pinned_cpus": [],
|                         "siblings": [],
|                         "memory": 1969,
|                         "mempages": [
|                             {
|                                 "nova_object.name": "NUMAPagesTopology",
|                                 "nova_object.data": {
|                                     "used": 0,
|                                     "total": 504202,
|                                     "reserved": 0,
|                                     "size_kb": 4
|                                 },
|                             },
|                             {
|                                 "nova_object.name": "NUMAPagesTopology",
|                                 "nova_object.data": {
|                                     "used": 0,
|                                     "total": 0,
|                                     "reserved": 0,
|                                     "size_kb": 2048
|                                 },
|                             },
|                             {
|                                 "nova_object.name": "NUMAPagesTopology",
|                                 "nova_object.data": {
|                                     "used": 0,
|                                     "total": 0,
|                                     "reserved": 0,
|                                     "size_kb": 1048576
|                                 },
|                             }
|                         ],
|                         "id": 1
|                     },
|                 },
|                 {
|                     "nova_object.name": "NUMACell",
|                     "nova_object.data": {
|                         "cpu_usage": 0,
|                         "memory_usage": 0,
```

(continues on next page)

(continued from previous page)

```
|                         "cpuset": [6, 7],
|                         "pinned_cpus": [],
|                         "siblings": [],
|                         "memory": 1967,
|                         "mempages": [
|                             {
|                                 "nova_object.name": "NUMAPagesTopology",
|                                 "nova_object.data": {
|                                     "used": 0,
|                                     "total": 503565,
|                                     "reserved": 0,
|                                     "size_kb": 4
|                                 },
|                             },
|                             {
|                                 "nova_object.name": "NUMAPagesTopology",
|                                 "nova_object.data": {
|                                     "used": 0,
|                                     "total": 0,
|                                     "reserved": 0,
|                                     "size_kb": 2048
|                                 },
|                             },
|                             {
|                                 "nova_object.name": "NUMAPagesTopology",
|                                 "nova_object.data": {
|                                     "used": 0,
|                                     "total": 0,
|                                     "reserved": 0,
|                                     "size_kb": 1048576
|                                 },
|                             }
|                         ],
|                         "id": 2
|                     },
|                 }
|             ]
|         },
| }
+--------------------------------------------------------------------------------+
```

This indeed shows that there are now 3 NUMA nodes for the host machine, the first with 4 GB of RAM and 4 CPUs, and others with 2 GB of RAM and 2 CPUs each.

4.18.5 Testing instance boot with no NUMA topology requested

For the sake of backwards compatibility, if the NUMA filter is enabled, but the flavor/image does not have any NUMA settings requested, it should be assumed that the guest will have a single NUMA node. The guest should be locked to a single host NUMA node too. Boot a guest with the *m1.tiny* flavor to test this condition:

```
$ . openrc admin admin
$ openstack server create --image cirros-0.3.4-x86_64-uec --flavor m1.tiny \
    cirros1
```

Now look at the libvirt guest XML:

```
$ sudo virsh list
 Id    Name                           State
----------------------------------------------------
 1     instance-00000001              running
$ sudo virsh dumpxml instance-00000001
...
<vcpu placement='static'>1</vcpu>
...
```

This example shows that there is no explicit NUMA topology listed in the guest XML.

4.18.6 Testing instance boot with 1 NUMA cell requested

Moving forward a little, explicitly tell nova that the NUMA topology for the guest should have a single NUMA node. This should operate in an identical manner to the default behavior where no NUMA policy is set. To define the topology we will create a new flavor:

```
$ openstack flavor create --ram 1024 --disk 1 --vcpus 4 m1.numa
$ openstack flavor set --property hw:numa_nodes=1 m1.numa
$ openstack flavor show m1.numa
```

Now boot the guest using this new flavor:

```
$ openstack server create --image cirros-0.3.4-x86_64-uec --flavor m1.numa \
    cirros2
```

Looking at the resulting guest XML from libvirt:

```
$ sudo virsh list
 Id    Name                           State
----------------------------------------------------
 1     instance-00000001              running
 2     instance-00000002              running
$ sudo virsh dumpxml instance-00000002
...
<vcpu placement='static'>4</vcpu>
<cputune>
  <vcpupin vcpu='0' cpuset='0-3'/>
  <vcpupin vcpu='1' cpuset='0-3'/>
  <vcpupin vcpu='2' cpuset='0-3'/>
  <vcpupin vcpu='3' cpuset='0-3'/>
  <emulatorpin cpuset='0-3'/>
</cputune>
...
<cpu>
  <topology sockets='4' cores='1' threads='1'/>
  <numa>
    <cell id='0' cpus='0-3' memory='1048576'/>
  </numa>
</cpu>
...
<numatune>
  <memory mode='strict' nodeset='0'/>
  <memnode cellid='0' mode='strict' nodeset='0'/>
</numatune>
```

The XML shows:

- Each guest CPU has been pinned to the physical CPUs associated with a particular NUMA node
- The emulator threads have been pinned to the union of all physical CPUs in the host NUMA node that the guest is placed on
- The guest has been given a virtual NUMA topology with a single node holding all RAM and CPUs
- The guest NUMA node has been strictly pinned to a host NUMA node.

As a further sanity test, check what nova recorded for the instance in the database. This should match the <numatune> information:

```
$ mysql -u root -p123456 nova
MariaDB [nova]> select numa_topology from instance_extra;
+--------------------------------------------------------------------------------+
| numa_topology                                                                  |
+--------------------------------------------------------------------------------+
| {
|     "nova_object.name": "InstanceNUMATopology",
|     "nova_object.data": {
|         "cells": [
|             {
|                 "nova_object.name": "InstanceNUMACell",
|                 "nova_object.data": {
|                     "pagesize": null,
|                     "cpu_topology": null,
|                     "cpuset": [0, 1, 2, 3],
|                     "cpu_policy": null,
|                     "memory": 1024,
|                     "cpu_pinning_raw": null,
|                     "id": 0,
|                     "cpu_thread_policy": null
|                 },
|             }
|         ]
|     },
| }
+--------------------------------------------------------------------------------+
```

Delete this instance:

```
$ openstack server delete cirros2
```

4.18.7 Testing instance boot with 2 NUMA cells requested

Now getting more advanced we tell nova that the guest will have two NUMA nodes. To define the topology we will change the previously defined flavor:

```
$ openstack flavor set --property hw:numa_nodes=2 m1.numa
$ openstack flavor show m1.numa
```

Now boot the guest using this changed flavor:

```
$ openstack server create --image cirros-0.3.4-x86_64-uec --flavor m1.numa \
    cirros2
```

Looking at the resulting guest XML from libvirt:

```
$ sudo virsh list
 Id    Name                           State
----------------------------------------------------
 1     instance-00000001              running
 3     instance-00000003              running
$ sudo virsh dumpxml instance-00000003
...
<vcpu placement='static'>4</vcpu>
<cputune>
  <vcpupin vcpu='0' cpuset='0-3'/>
  <vcpupin vcpu='1' cpuset='0-3'/>
  <vcpupin vcpu='2' cpuset='4-5'/>
  <vcpupin vcpu='3' cpuset='4-5'/>
  <emulatorpin cpuset='0-5'/>
</cputune>
...
<cpu>
  <topology sockets='4' cores='1' threads='1'/>
  <numa>
    <cell id='0' cpus='0-1' memory='524288'/>
    <cell id='1' cpus='2-3' memory='524288'/>
  </numa>
</cpu>
...
<numatune>
  <memory mode='strict' nodeset='0-1'/>
  <memnode cellid='0' mode='strict' nodeset='0'/>
  <memnode cellid='1' mode='strict' nodeset='1'/>
</numatune>
```

The XML shows:

- Each guest CPU has been pinned to the physical CPUs associated with particular NUMA nodes
- The emulator threads have been pinned to the union of all physical CPUs in the host NUMA nodes that the guest is placed on
- The guest has been given a virtual NUMA topology with two nodes, each holding half the RAM and CPUs
- The guest NUMA nodes have been strictly pinned to different host NUMA node

As a further sanity test, check what nova recorded for the instance in the database. This should match the `<numatune>` information:

```
MariaDB [nova]> select numa_topology from instance_extra;
+--------------------------------------------------------------------------------+
| numa_topology                                                                  |
+--------------------------------------------------------------------------------+
| {
|     "nova_object.name": "InstanceNUMATopology",
|     "nova_object.data": {
|         "cells": [
|             {
|                 "nova_object.name": "InstanceNUMACell",
|                 "nova_object.data": {
|                     "pagesize": null,
|                     "cpu_topology": null,
|                     "cpuset": [0, 1],
|                     "cpu_policy": null,
```

(continues on next page)

(continued from previous page)

```
|                         "memory": 512,
|                         "cpu_pinning_raw": null,
|                         "id": 0,
|                         "cpu_thread_policy": null
|                     },
|                 },
|                 {
|                     "nova_object.name": "InstanceNUMACell",
|                     "nova_object.data": {
|                         "pagesize": null,
|                         "cpu_topology": null,
|                         "cpuset": [2, 3],
|                         "cpu_policy": null,
|                         "memory": 512,
|                         "cpu_pinning_raw": null,
|                         "id": 1,
|                         "cpu_thread_policy": null
|                     },
|                 }
|             ]
|         },
| }
+--------------------------------------------------------------------------------+
```

4.19 Testing Serial Console

The main aim of this feature is exposing an interactive web-based serial consoles through a web-socket proxy. This page describes how to test it from a devstack environment.

4.19.1 Setting up a devstack environment

For instructions on how to setup devstack with serial console support enabled see this guide.

4.19.2 Testing the API

Starting a new instance.

```
# cd devstack && . openrc
# nova boot --flavor 1 --image cirros-0.3.2-x86_64-uec cirros1
```

Nova provides a command *nova get-serial-console* which will returns a URL with a valid token to connect to the serial console of VMs.

```
# nova get-serial-console cirros1
+--------+-----------------------------------------------------------------+
| Type   | Url                                                             |
+--------+-----------------------------------------------------------------+
| serial | ws://127.0.0.1:6083/?token=5f7854b7-bf3a-41eb-857a-43fc33f0b1ec |
+--------+-----------------------------------------------------------------+
```

Currently nova does not provide any client able to connect from an interactive console through a web-socket. A simple client for *test purpose* can be written with few lines of Python.

```
# sudo easy_install ws4py || sudo pip install ws4py
# cat >> client.py <<EOF
import sys
from ws4py.client.threadedclient import WebSocketClient
class LazyClient(WebSocketClient):
    def run(self):
        try:
            while not self.terminated:
                try:
                    b = self.sock.recv(4096)
                    sys.stdout.write(b)
                    sys.stdout.flush()
                except: # socket error expected
                    pass
        finally:
            self.terminate()
if __name__ == '__main__':
    if len(sys.argv) != 2 or not sys.argv[1].startswith("ws"):
        print "Usage %s: Please use websocket url"
        print "Example: ws://127.0.0.1:6083/?token=xxx"
        exit(1)
    try:
        ws = LazyClient(sys.argv[1], protocols=['binary'])
        ws.connect()
        while True:
            # keyboard event...
            c = sys.stdin.read(1)
            if c:
                ws.send(c)
        ws.run_forever()
    except KeyboardInterrupt:
        ws.close()
EOF
```

```
# python client.py ws://127.0.0.1:6083/?token=5f7854b7-bf3a-41eb-857a-43fc33f0b1ec
<enter>
cirros1 login
```

4.20 Testing Zero Downtime Upgrade Process

Zero Downtime upgrade eliminates any disruption to nova API service during upgrade.

Nova API services are upgraded at the end. The basic idea of the zero downtime upgrade process is to have the connections drain from the old API before being upgraded. In this process, new connections go to the new API nodes while old connections slowly drain from the old nodes. This ensures that the user sees the max_supported API version as a monotonically increasing number. There might be some performance degradation during the process due to slow HTTP responses and delayed request handling, but there is no API downtime.

This page describes how to test the zero downtime upgrade process.

4.20.1 Environment

- **Multinode devstack environment with 2 nodes:**
  - controller - All services (N release)

  - compute-api - Only n-cpu and n-api services (N release)

- Highly available load balancer (HAProxy) on top of the n-api services. This is required for zero downtime upgrade as it allows one n-api service to run while we upgrade the other. See instructions to setup HAProxy below.

4.20.2 Instructions to setup HAProxy

Install HAProxy and Keepalived on both nodes.

```
# apt-get install haproxy keepalived
```

Let the kernel know that we intend to bind additional IP addresses that wont be defined in the interfaces file. To do this, edit `/etc/sysctl.conf` and add the following line:

```
net.ipv4.ip_nonlocal_bind=1
```

Make this take effect without rebooting.

```
# sysctl -p
```

Configure HAProxy to add backend servers and assign virtual IP to the frontend. On both nodes add the below HAProxy config:

```
# cd /etc/haproxy
# cat >> haproxy.cfg <<EOF

   global
      chroot /var/lib/haproxy
      user haproxy
      group haproxy
      daemon
      log 192.168.0.88 local0
      pidfile  /var/run/haproxy.pid
      stats socket /var/run/haproxy.sock mode 600 level admin
      stats timeout 2m
      maxconn 4000

   defaults
      log  global
      maxconn  8000
      mode  http
      option  redispatch
      retries  3
      stats  enable
      timeout  http-request 10s
      timeout  queue 1m
      timeout  connect 10s
      timeout  client 1m
      timeout  server 1m
      timeout  check 10s

   frontend nova-api-vip
      bind 192.168.0.95:8282             <<ha proxy virtual ip>>
      default_backend nova-api

   backend nova-api
```

(continues on next page)

(continued from previous page)

```
    balance  roundrobin
    option  tcplog
    server  controller 192.168.0.88:8774  check
    server  apicomp  192.168.0.89:8774  check

 EOF
```

Note: Just change the IP for log in the global section on each node.

On both nodes add `keepalived.conf`:

```
# cd /etc/keepalived
# cat >> keepalived.conf <<EOF

   global_defs {
      router_id controller
   }
   vrrp_script haproxy {
      script "killall -0 haproxy"
      interval 2
      weight 2
   }
   vrrp_instance 50 {
      virtual_router_id 50
      advert_int 1
      priority 101
      state MASTER
      interface eth0
      virtual_ipaddress {
         192.168.0.95 dev eth0
      }
      track_script {
         haproxy
      }
   }

EOF
```

Note: Change priority on node2 to 100 (or vice-versa). Add HAProxy virtual IP.

Restart keepalived service.

```
# service keepalived restart
```

Add `ENABLED=1` in `/etc/default/haproxy` and then restart HAProxy service.

```
# service haproxy restart
```

When both the services have restarted, node with the highest priority for keepalived claims the virtual IP. You can check which node claimed the virtual IP using:

```
# ip a
```

4.20.3 Zero Downtime upgrade process

General rolling upgrade process: *Minimal Downtime Upgrade Process*.

Before Upgrade

- Change nova-api endpoint in keystone to point to the HAProxy virtual IP.
- Run tempest tests
- Check if n-api services on both nodes are serving the requests.

Before maintenance window

- Start the upgrade process with controller node.
- Follow the steps from the general rolling upgrade process to install new code and sync the db for schema changes.

During maintenance window

- Set compute option in upgrade_levels to auto in `nova.conf`.

```
[upgrade_levels]
compute = auto
```

- Starting with n-cond restart all services except n-api and n-cpu.
- In small batches gracefully shutdown nova-cpu, then start n-cpu service with new version of the code.
- Run tempest tests.
- Drain connections on n-api while the tempest tests are running. HAProxy allows you to drain the connections by setting weight to zero:

```
# echo "set weight nova-api/<<server>> 0" | sudo socat /var/run/haproxy.sock stdio
```

- OR disable service using:

```
# echo "disable server nova-api/<<server>>" | sudo socat /var/run/haproxy.sock␣
↪stdio
```

- This allows the current node to complete all the pending requests. When this is being upgraded, other api node serves the requests. This way we can achieve zero downtime.
- Restart n-api service and enable n-api using the command:

```
# echo "enable server nova-api/<<server>>" | sudo socat /var/run/haproxy.sock␣
↪stdio
```

- Drain connections from other old api node in the same way and upgrade.
- No tempest tests should fail since there is no API downtime.

After maintenance window

- Follow the steps from general rolling upgrade process to clear any cached service version data and complete all online data migrations.

4.21 Testing Down Cells

This document describes how to recreate a down-cell scenario in a single-node devstack environment. This can be useful for testing the reliability of the controller services when a cell in the deployment is down.

4.21.1 Setup

DevStack config

This guide is based on a devstack install from the Train release using an Ubuntu Bionic 18.04 VM with 8 VCPU, 8 GB RAM and 200 GB of disk following the All-In-One Single Machine guide.

The following minimal local.conf was used:

```
[[local|localrc]]
# Define passwords
OS_PASSWORD=openstack1
SERVICE_TOKEN=$OS_PASSWORD
ADMIN_PASSWORD=$OS_PASSWORD
MYSQL_PASSWORD=$OS_PASSWORD
RABBIT_PASSWORD=$OS_PASSWORD
SERVICE_PASSWORD=$OS_PASSWORD
# Logging config
LOGFILE=$DEST/logs/stack.sh.log
LOGDAYS=2
# Disable non-essential services
disable_service horizon tempest
```

Populate cell1

Create a test server first so there is something in cell1:

```
$ source openrc admin admin
$ IMAGE=$(openstack image list -f value -c ID)
$ openstack server create --wait --flavor m1.tiny --image $IMAGE cell1-server
```

4.21.2 Take down cell1

Break the connection to the cell1 database by changing the `database_connection` URL, in this case with an invalid host IP:

```
mysql> select database_connection from cell_mappings where name='cell1';
+----------------------------------------------------------------+
| database_connection                                            |
+----------------------------------------------------------------+
| mysql+pymysql://root:openstack1@127.0.0.1/nova_cell1?charset=utf8 |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------------------------------------+
1 row in set (0.00 sec)

mysql> update cell_mappings set database_connection='mysql+pymysql://
↪root:openstack1@192.0.0.1/nova_cell1?charset=utf8' where name='cell1';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

4.21.3 Update controller services

Prepare the controller services for the down cell. See *Handling cell failures* for details.

Modify nova.conf

Configure the API to avoid long timeouts and slow start times due to bug 1815697 by modifying `/etc/nova/nova.conf`:

```
[database]
...
max_retries = 1
retry_interval = 1

[upgrade_levels]
...
compute = stein  # N-1 from train release, just something other than "auto"
```

Restart services

Note: It is useful to tail the n-api service logs in another screen to watch for errors / warnings in the logs due to down cells:

```
$ sudo journalctl -f -a -u devstack@n-api.service
```

Restart controller services to flush the cell cache:

```
$ sudo systemctl restart devstack@n-api.service devstack@n-super-cond.service␣
↪devstack@n-sch.service
```

4.21.4 Test cases

1. Try to create a server which should fail and go to cell0.

   ```
   $ openstack server create --wait --flavor m1.tiny --image $IMAGE cell0-server
   ```

   You can expect to see errors like this in the n-api logs:

```
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context [None req-
↪fdaff415-48b9-44a7-b4c3-015214e80b90 None None] Error gathering result from␣
↪cell 4f495a21-294a-4051-9a3d-8b34a250bbb4: DBConnectionError: (pymysql.err.
↪OperationalError) (2003, "Can't connect to MySQL server on u'192.0.0.1' ([Errno␣
↪101] ENETUNREACH)") (Background on this error at: http://sqlalche.me/e/e3q8)
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context Traceback␣
↪(most recent call last):
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context   File "/
↪opt/stack/nova/nova/context.py", line 441, in gather_result
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context     ␣
↪result = fn(cctxt, *args, **kwargs)
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context   File "/
↪opt/stack/nova/nova/db/sqlalchemy/api.py", line 211, in wrapper
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context     with␣
↪reader_mode.using(context):
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context   File "/
↪usr/lib/python2.7/contextlib.py", line 17, in __enter__
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context     ␣
↪return self.gen.next()
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context   File "/
↪usr/local/lib/python2.7/dist-packages/oslo_db/sqlalchemy/enginefacade.py", line␣
↪1061, in _transaction_scope
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context     ␣
↪context=context) as resource:
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context   File "/
↪usr/lib/python2.7/contextlib.py", line 17, in __enter__
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context     ␣
↪return self.gen.next()
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context   File "/
↪usr/local/lib/python2.7/dist-packages/oslo_db/sqlalchemy/enginefacade.py", line␣
↪659, in _session
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context     ␣
↪bind=self.connection, mode=self.mode)
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context   File "/
↪usr/local/lib/python2.7/dist-packages/oslo_db/sqlalchemy/enginefacade.py", line␣
↪418, in _create_session
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context     self._
↪start()
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context   File "/
↪usr/local/lib/python2.7/dist-packages/oslo_db/sqlalchemy/enginefacade.py", line␣
↪510, in _start
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context     ␣
↪engine_args, maker_args)
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context   File "/
↪usr/local/lib/python2.7/dist-packages/oslo_db/sqlalchemy/enginefacade.py", line␣
↪534, in _setup_for_connection
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context     sql_
↪connection=sql_connection, **engine_kwargs)
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context   File "/
↪usr/local/lib/python2.7/dist-packages/debtcollector/renames.py", line 43, in␣
↪decorator
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context     ␣
↪return wrapped(*args, **kwargs)
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context   File "/
↪usr/local/lib/python2.7/dist-packages/oslo_db/sqlalchemy/engines.py", line 201,␣
↪in create_engine
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context     test_
↪conn = _test_connection(engine, max_retries, retry_interval)
```

(continues on next page)

(continued from previous page)

```
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context   File "/
↪usr/local/lib/python2.7/dist-packages/oslo_db/sqlalchemy/engines.py", line 387,␣
↪in _test_connection
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context     six.
↪reraise(type(de_ref), de_ref)
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context   File "
↪<string>", line 3, in reraise
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context␣
↪DBConnectionError: (pymysql.err.OperationalError) (2003, "Can't connect to␣
↪MySQL server on u'192.0.0.1' ([Errno 101] ENETUNREACH)") (Background on this␣
↪error at: http://sqlalche.me/e/e3q8)
Apr 04 20:48:22 train devstack@n-api.service[10884]: ERROR nova.context
Apr 04 20:48:22 train devstack@n-api.service[10884]: WARNING nova.objects.service␣
↪[None req-1cf4bf5c-2f74-4be0-a18d-51ff81df57dd admin admin] Failed to get␣
↪minimum service version for cell 4f495a21-294a-4051-9a3d-8b34a250bbb4
```

2. List servers with the 2.69 microversion for down cells.

   Note: Requires python-openstackclient >= 3.18.0 for v2.69 support.

   The server in cell1 (which is down) will show up with status UNKNOWN:

```
$ openstack --os-compute-api-version 2.69 server list
+--------------------------------------+--------------+---------+----------+------
↪--------------------+--------+
| ID                                   | Name         | Status  | Networks |␣
↪Image                      | Flavor |
+--------------------------------------+--------------+---------+----------+------
↪--------------------+--------+
| 8e90f1f0-e8dd-4783-8bb3-ec8d594e60f1 |              | UNKNOWN |          |      ␣
↪                    |        |
| afd45d84-2bd7-4e49-9dff-93359f742bc1 | cell0-server | ERROR   |          |␣
↪cirros-0.4.0-x86_64-disk |        |
+--------------------------------------+--------------+---------+----------+------
↪--------------------+--------+
```

3. Using v2.1 the UNKNOWN server is filtered out by default due to *api.list_records_by_skipping_down_cells*:

```
$ openstack --os-compute-api-version 2.1 server list
+--------------------------------------+--------------+--------+----------+-------
↪-------------------+---------+
| ID                                   | Name         | Status | Networks | Image␣
↪                   | Flavor  |
+--------------------------------------+--------------+--------+----------+-------
↪-------------------+---------+
| afd45d84-2bd7-4e49-9dff-93359f742bc1 | cell0-server | ERROR  |          |␣
↪cirros-0.4.0-x86_64-disk | m1.tiny |
+--------------------------------------+--------------+--------+----------+-------
↪-------------------+---------+
```

4. Configure nova-api with `list_records_by_skipping_down_cells=False`

```
[api]
list_records_by_skipping_down_cells = False
```

5. Restart nova-api and then listing servers should fail:

```
$ sudo systemctl restart devstack@n-api.service
$ openstack --os-compute-api-version 2.1 server list
Unexpected API Error. Please report this at http://bugs.launchpad.net/nova/ and␣
↪attach the Nova API log if possible.
<class 'nova.exception.NovaException'> (HTTP 500) (Request-ID: req-e2264d67-5b6c-
↪4f17-ae3d-16c7562f1b69)
```

6. Try listing compute services with a down cell.

   The services from the down cell are skipped:

```
$ openstack --os-compute-api-version 2.1 compute service list
+----+------------------+-------+----------+---------+-------+--------------------
↪--------+
| ID | Binary           | Host  | Zone     | Status  | State | Updated At        ␣
↪        |
+----+------------------+-------+----------+---------+-------+--------------------
↪--------+
|  2 | nova-scheduler   | train | internal | enabled | up    | 2019-04-
↪04T21:12:47.000000 |
|  6 | nova-consoleauth | train | internal | enabled | up    | 2019-04-
↪04T21:12:38.000000 |
|  7 | nova-conductor   | train | internal | enabled | up    | 2019-04-
↪04T21:12:47.000000 |
+----+------------------+-------+----------+---------+-------+--------------------
↪--------+
```

   With 2.69 the nova-compute service from cell1 is shown with status UNKNOWN:

```
$ openstack --os-compute-api-version 2.69 compute service list
+--------------------------------------+------------------+-------+----------+----
↪-----+-------+----------------------------+
| ID                                   | Binary           | Host  | Zone     |␣
↪Status  | State | Updated At                 |
+--------------------------------------+------------------+-------+----------+----
↪-----+-------+----------------------------+
| f68a96d9-d994-4122-a8f9-1b0f68ed69c2 | nova-scheduler   | train | internal |␣
↪enabled | up    | 2019-04-04T21:13:47.000000 |
| 70cd668a-6d60-4a9a-ad83-f863920d4c44 | nova-consoleauth | train | internal |␣
↪enabled | up    | 2019-04-04T21:13:38.000000 |
| ca88f023-1de4-49e0-90b0-581e16bebaed | nova-conductor   | train | internal |␣
↪enabled | up    | 2019-04-04T21:13:47.000000 |
|                                      | nova-compute     | train |          |␣
↪UNKNOWN |       |                            |
+--------------------------------------+------------------+-------+----------+----
↪-----+-------+----------------------------+
```

4.21.5 Future

This guide could be expanded for having multiple non-cell0 cells where one cell is down while the other is available and go through scenarios where the down cell is marked as disabled to take it out of scheduling consideration.

4.22 Profiling With Eventlet

When performance of one of the Nova services is worse than expected, and other sorts of analysis do not lead to candidate fixes, profiling is an excellent tool for producing detailed analysis of what methods in the code are called the most and which consume the most time.

Because most Nova services use eventlet, the standard profiling tool provided with Python, cProfile, will not work. Something is required to keep track of changing tasks. Thankfully eventlet comes with `eventlet.green.profile.Profile`, a mostly undocumented class that provides a similar (but not identical) API to the one provided by Pythons `Profile` while outputting the same format.

Note: The eventlet Profile outputs the `prof` format produced by `profile`, which is not the same as that output by `cProfile`. Some analysis tools (for example, SnakeViz) only read the latter so the options for analyzing eventlet profiling are not always deluxe (see below).

4.22.1 Setup

This guide assumes the Nova service being profiled is running devstack, but that is not necessary. What is necessary is that the code associated with the service can be changed and the service restarted, in place.

Profiling the entire service will produce mostly noise and the output will be confusing because different tasks will operate during the profile run. It is better to begin the process with a candidate task or method *within* the service that can be associated with an identifier. For example, `select_destinations` in the `FilterScheduler` can be associated with the list of `instance_uuids` passed to it and it runs only once for that set of instance uuids.

The process for profiling is:

1. Identify the method to be profiled.
2. Populate the environment with sufficient resources to exercise the code. For example you may wish to use the FakeVirtDriver to have nova aware of multiple `nova-compute` processes. Or you may wish to launch many instances if you are evaluating a method that loops over instances.
3. At the start of that method, change the code to instantiate a `Profile` object and `start()` it.
4. At the end of that method, change the code to `stop()` profiling and write the data (with `dump_stats()`) to a reasonable location.
5. Restart the service.
6. Cause the method being evaluated to run.
7. Analyze the profile data with the pstats module.

Note: `stop()` and `start()` are two of the ways in which the eventlet `Profile` API differs from the stdlib. There the methods are `enable()` and `disable()`.

4.22.2 Example

For this example we will analyze `select_destinations` in the `FilterScheduler`. A known problem is that it does excessive work when presented with too many candidate results from the Placement service. Wed like to know why.

Well configure and run devstack with FakeVirtDriver so there are several candidate hypervisors (the following `local.conf` is also useful for other profiling and benchmarking scenarios so not all changes are relevant here):

```
[[local|localrc]]
ADMIN_PASSWORD=secret
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
USE_PYTHON3=True
VIRT_DRIVER=fake
# You may use different numbers of fake computes, but be careful: 100 will
# completely overwhelm a 16GB, 16VPCU server. In the test profiles below a
# value of 50 was used, on a 16GB, 16VCPU server.
NUMBER_FAKE_NOVA_COMPUTE=25
disable_service cinder
disable_service horizon
disable_service dstat
disable_service tempest

[[post-config|$NOVA_CONF]]
rpc_response_timeout = 300

# Disable filtering entirely. For some profiling this will not be what you
# want.
[filter_scheduler]
enabled_filters = '"'
# Send only one type of notifications to avoid notification overhead.
[notifications]
notification_format = unversioned
```

Change the code in `nova/scheduler/filter_scheduler.py` as follows:

```
diff --git a/nova/scheduler/filter_scheduler.py b/nova/scheduler/filter_scheduler.py
index 672f23077e..cb0f87fe48 100644
--- a/nova/scheduler/filter_scheduler.py
+++ b/nova/scheduler/filter_scheduler.py
@@ -49,92 +49,99 @@ class FilterScheduler(driver.Scheduler):
     def select_destinations(self, context, spec_obj, instance_uuids,
             alloc_reqs_by_rp_uuid, provider_summaries,
             allocation_request_version=None, return_alternates=False):
         """Returns a list of lists of Selection objects, which represent the
         hosts and (optionally) alternates for each instance.

         :param context: The RequestContext object
         :param spec_obj: The RequestSpec object
         :param instance_uuids: List of UUIDs, one for each value of the spec
                                object's num_instances attribute
         :param alloc_reqs_by_rp_uuid: Optional dict, keyed by resource provider
                                       UUID, of the allocation_requests that may
                                       be used to claim resources against
                                       matched hosts. If None, indicates either
                                       the placement API wasn't reachable or
                                       that there were no allocation_requests
                                       returned by the placement API. If the
                                       latter, the provider_summaries will be an
                                       empty dict, not None.
         :param provider_summaries: Optional dict, keyed by resource provider
                                    UUID, of information that will be used by
```

(continues on next page)

(continued from previous page)

```
                                    the filters/weighers in selecting matching
                                    hosts for a request. If None, indicates that
                                    the scheduler driver should grab all compute
                                    node information locally and that the
                                    Placement API is not used. If an empty dict,
                                    indicates the Placement API returned no
                                    potential matches for the requested
                                    resources.
        :param allocation_request_version: The microversion used to request the
                                           allocations.
        :param return_alternates: When True, zero or more alternate hosts are
                                  returned with each selected host. The number
                                  of alternates is determined by the
                                  configuration option
                                  `CONF.scheduler.max_attempts`.
        """
+       from eventlet.green import profile
+       pr = profile.Profile()
+       pr.start()
+
        self.notifier.info(
            context, 'scheduler.select_destinations.start',
            dict(request_spec=spec_obj.to_legacy_request_spec_dict()))
        compute_utils.notify_about_scheduler_action(
            context=context, request_spec=spec_obj,
            action=fields_obj.NotificationAction.SELECT_DESTINATIONS,
            phase=fields_obj.NotificationPhase.START)

        host_selections = self._schedule(context, spec_obj, instance_uuids,
                alloc_reqs_by_rp_uuid, provider_summaries,
                allocation_request_version, return_alternates)
        self.notifier.info(
            context, 'scheduler.select_destinations.end',
            dict(request_spec=spec_obj.to_legacy_request_spec_dict()))
        compute_utils.notify_about_scheduler_action(
            context=context, request_spec=spec_obj,
            action=fields_obj.NotificationAction.SELECT_DESTINATIONS,
            phase=fields_obj.NotificationPhase.END)
+       pr.stop()
+       pr.dump_stats('/tmp/select_destinations/%s.prof' % ':'.join(instance_uuids))
+
        return host_selections
```

Make a `/tmp/select_destinations` directory that is writable by the user nova-scheduler will run as. This is where the profile output will go.

Restart the scheduler service. Note that `systemctl restart` may not kill things sufficiently dead, so:

```
sudo systemctl stop devstack@n-sch
sleep 5
sudo systemctl start devstack@n-sch
```

Create a server (which will call `select_destinations`):

```
openstack server create --image cirros-0.4.0-x86_64-disk --flavor c1 x1
```

In `/tmp/select_destinations` there should be a file with a name using the uuid of the created server with a

`.prof` extension.

Change to that directory and view the profile using the pstats interactive mode:

```
python3 -m pstats ef044142-f3b8-409d-9af6-c60cea39b273.prof
```

Note: The major version of python used to analyze the profile data must be the same as the version used to run the process being profiled.

Sort stats by their cumulative time:

```
ef044142-f3b8-409d-9af6-c60cea39b273.prof% sort cumtime
ef044142-f3b8-409d-9af6-c60cea39b273.prof% stats 10
Tue Aug  6 17:17:56 2019    ef044142-f3b8-409d-9af6-c60cea39b273.prof

         603477 function calls (587772 primitive calls) in 2.294 seconds

   Ordered by: cumulative time
   List reduced from 2484 to 10 due to restriction <10>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    1.957    1.957 profile:0(start)
        1    0.000    0.000    1.911    1.911 /mnt/share/opt/stack/nova/nova/
↪scheduler/filter_scheduler.py:113(_schedule)
        1    0.000    0.000    1.834    1.834 /mnt/share/opt/stack/nova/nova/
↪scheduler/filter_scheduler.py:485(_get_all_host_states)
        1    0.000    0.000    1.834    1.834 /mnt/share/opt/stack/nova/nova/
↪scheduler/host_manager.py:757(get_host_states_by_uuids)
        1    0.004    0.004    1.818    1.818 /mnt/share/opt/stack/nova/nova/
↪scheduler/host_manager.py:777(_get_host_states)
  104/103    0.001    0.000    1.409    0.014 /usr/local/lib/python3.6/dist-packages/
↪oslo_versionedobjects/base.py:170(wrapper)
       50    0.001    0.000    1.290    0.026 /mnt/share/opt/stack/nova/nova/
↪scheduler/host_manager.py:836(_get_instance_info)
       50    0.001    0.000    1.289    0.026 /mnt/share/opt/stack/nova/nova/
↪scheduler/host_manager.py:820(_get_instances_by_host)
      103    0.001    0.000    0.890    0.009 /usr/local/lib/python3.6/dist-packages/
↪sqlalchemy/orm/query.py:3325(__iter__)
       50    0.001    0.000    0.776    0.016 /mnt/share/opt/stack/nova/nova/objects/
↪host_mapping.py:99(get_by_host)
```

From this we can make a couple of useful inferences about `get_by_host`:

- It is called once for each of the 50 `FakeVirtDriver` hypervisors configured for these tests.
- It (and the methods it calls internally) consumes about 40% of the entire time spent running (`0.776 / 1.957`) the `select_destinations` method (indicated by `profile:0(start)`, above).

Several other sort modes can be used. List those that are available by entering `sort` without arguments.

4.22.3 Caveats

Real world use indicates that the eventlet profiler is not perfect. There are situations where it will not always track switches between greenlets as well as it could. This can result in profile data that does not make sense or random slowdowns in the system being profiled. There is no one size fits all solution to these issues; profiling eventlet services is more an art than science. However, this section tries to provide a (hopefully) growing body of advice on what to do to work around problems.

General Advice

- Try to profile chunks of code that operate mostly within one module or class and do not have many collaborators. The more convoluted the path through the code, the more confused the profiler gets.
- Similarly, where possible avoid profiling code that will trigger many greenlet context switches; either specific spawns, or multiple types of I/O. Instead, narrow the focus of the profiler.
- If possible, avoid RPC.

In nova-compute

The creation of this caveat section was inspired by issues experienced while profiling `nova-compute`. The `nova-compute` process is not allowed to speak with a database server directly. Instead communication is mediated through the conductor, communication happening via `oslo.versionedobjects` and remote calls. Profiling methods such as `_update_available_resources` in the ResourceTracker, which needs information from the database, results in profile data that can be analyzed but is incorrect and misleading.

This can be worked around by temporarily changing `nova-compute` to allow it to speak to the database directly:

```
diff --git a/nova/cmd/compute.py b/nova/cmd/compute.py
index 01fd20de2e..655d503158 100644
--- a/nova/cmd/compute.py
+++ b/nova/cmd/compute.py
@@ -50,8 +50,10 @@ def main():

     gmr.TextGuruMeditation.setup_autorun(version, conf=CONF)

-    cmd_common.block_db_access('nova-compute')
-    objects_base.NovaObject.indirection_api = conductor_rpcapi.ConductorAPI()
+    # Temporarily allow access to the database. You must update the config file
+    # used by this process to set [database]/connection to the cell1 database.
+    # cmd_common.block_db_access('nova-compute')
+    # objects_base.NovaObject.indirection_api = conductor_rpcapi.ConductorAPI()
     objects.Service.enable_min_version_cache()
     server = service.Service.create(binary='nova-compute',
                                     topic=compute_rpcapi.RPC_TOPIC)
```

The configuration file used by the `nova-compute` process must also be updated to ensure that it contains a setting for the relevant database:

```
[database]
connection = mysql+pymysql://root:secret@127.0.0.1/nova_cell1?charset=utf8
```

In a single node devstack setup `nova_cell1` is the right choice. The connection string will vary in other setups.

Once these changes are made, along with the profiler changes indicated in the example above, `nova-compute` can be restarted and with luck some useful profiling data will emerge.

4.23 How to get (more) involved with Nova

So you want to get more involved with Nova? Or you are new to Nova and wondering where to start?

We are working on building easy ways for you to get help and ideas on how to learn more about Nova and how the Nova community works.

Any questions, please ask! If you are unsure who to ask, then please contact the PTL.

4.23.1 How do I get started?

There are quite a few global docs on this:

- https://docs.openstack.org/contributors/
- https://www.openstack.org/community/
- https://www.openstack.org/assets/welcome-guide/OpenStackWelcomeGuide.pdf
- https://wiki.openstack.org/wiki/How_To_Contribute

There is more general info, non Nova specific info here:

- https://wiki.openstack.org/wiki/Mentoring
- https://docs.openstack.org/upstream-training/

What should I work on?

So you are starting out your Nova journey, where is a good place to start?

If youd like to learn how Nova works before changing anything (good idea!), we recommend looking for reviews with -1s and -2s and seeing why they got downvoted. There is also the *Code Review Guide for Nova*. Once you have some understanding, start reviewing patches. Its OK to ask people to explain things you dont understand. Its also OK to see some potential problems but put a +0.

Another way is to look for a subteam youd like to get involved with and review their patches. See: https://etherpad.openstack.org/p/train-nova-subteam-tracking

Once youre ready to write code, take a look at some of the work already marked as low-hanging fruit:

- https://bugs.launchpad.net/nova/+bugs?field.tag=low-hanging-fruit

How do I get my feature in?

The best way of getting your feature in is well it depends.

First concentrate on solving your problem and/or use case, dont fixate on getting the code you have working merged. Its likely things will need significant re-work after you discuss how your needs match up with all the existing ways Nova is currently being used. The good news, is this process should leave you with a feature thats more flexible and doesnt lock you into your current way of thinking.

A key part of getting code merged, is helping with reviewing other peoples code. Great reviews of others code will help free up more core reviewer time to look at your own patches. In addition, you will understand how the review is thinking when they review your code.

Also, work out if any on going efforts are blocking your feature and helping out speeding those up. The spec review process should help with this effort.

For more details on our process, please see: *Nova team process*.

What is expected of a good contributor?

TODO - need more info on this

4.23.2 Top Tips for working with the Nova community

Here are some top tips around engaging with the Nova community:

- IRC
  - we talk a lot in #openstack-nova
  - do ask us questions in there, and we will try to help you
  - not sure about asking questions? feel free to listen in around other peoples questions
  - we recommend you setup an IRC bouncer: https://docs.openstack.org/contributors/common/irc.html
- Email
  - Use the [nova] tag in the mailing lists
  - Filtering on [nova] and [all] can help tame the list
- Be Open
  - i.e. dont review your teams code in private, do it publicly in gerrit
  - i.e. be ready to talk about openly about problems you are having, not theoretical issues
  - that way you can start to gain the trust of the wider community
- Got a problem? Please ask!
  - Please raise any problems and ask questions early
  - we want to help you before you are frustrated or annoyed
  - unsure who to ask? Just ask in IRC, or check out the list of Nova people.
- Talk about problems first, then solutions
  - Nova is a big project. At first, it can be hard to see the big picture
  - Dont think about merging your patch, instead think about solving your problem
  - conversations are more productive that way
- Its not the decision thats important, its the reason behind it thats important
  - Dont like the way the community is going?
  - Please ask why we were going that way, and please engage with the debate
  - If you dont, we are unable to learn from what you have to offer
- No one will decide, this is stuck, who can help me?
  - its rare, but it happens
  - its the Nova PTLs job to help you
  - but if you dont ask, its hard for them to help you

4.23.3 Process

It can feel like you are faced with a wall of process. We are a big community, to make sure the right communication happens, we do use a minimal amount of process.

If you find something that doesnt make sense, please:

- ask questions to find out *why* it happens

- if you know of a better way to do it, please speak up
- one better way might be to remove the process if it no longer helps

To learn more about Novas process, please read *Nova team process*.

Why bother with any process?

Why is it worth creating a bug or blueprint to track your code review? This may seem like silly process, but there is usually a good reason behind it.

We have lots of code to review, and we have tools to try and get to really important code reviews first. If yours is really important, but not picked up by our tools, its possible you just get lost in the bottom of a big queue.

If you have a bug fix, you have done loads of work to identify the issue, and test out your fix, and submit it. By adding a bug report, you are making it easier for other folks who hit the same problem to find your work, possibly saving them the hours of pain you went through. With any luck that gives all those people the time to fix different bugs, all that might have affected you, if you had not given them the time go fix it.

Its similar with blueprints. You have worked out how to scratch your itch, lets tell others about that great new feature you have added, so they can use that. Also, it stops someone with a similar idea going through all the pain of creating a feature only to find you already have that feature ready and up for review, or merged into the latest release.

Hopefully this gives you an idea why we have applied a small layer of process to what we are doing. Having said all this, we need to unlearn old habits to move forward, there may be better ways to do things, and we are open to trying them. Please help be part of the solution.

4.23.4 Why do code reviews if I am not in nova-core?

Code reviews are the life blood of the Nova developer community.

There is a good discussion on how you do good reviews, and how anyone can be a reviewer: http://docs.openstack.org/infra/manual/developers.html#peer-review

In the draft process guide, I discuss how doing reviews can help get your code merged faster: *Nova team process*.

Lets look at some of the top reasons why participating with code reviews really helps you:

- Doing more reviews, and seeing what other reviewers notice, will help you better understand what is expected of code that gets merged into master
- Having more non-core people do great reviews, leaves less review work for the core reviewers to do, so we are able get more code merged
- Empathy is one of the keys to a happy community. If you are used to doing code reviews, you will better understand the comments you get when people review your code. As you do more code reviews, and see what others notice, you will get a better idea of what people are looking for when then apply a +2 to your code.
- TODO - needs more detail

What are the most useful types of code review comments? Well here are a few to the top ones:

- Fundamental flaws are the biggest thing to spot. Does the patch break a whole set of existing users, or an existing feature?
- Consistency of behaviour is really important. Does this bit of code do things differently to where similar things happen else where in Nova?
- Is the code easy to maintain, well tested and easy to read? Code is read order of magnitude times more than it is written, so optimise for the reader of the code, not the writer.

- TODO - what others should go here?

Lets look at some problems people hit when starting out doing code reviews:

- My +1 doesnt mean anything, why should I bother?
  - So your +1 really does help. Some really useful -1 votes that lead to a +1 vote helps get code into a position
- When to use -1 vs 0 vs +1
  - Please see the guidelines here: http://docs.openstack.org/infra/manual/developers.html#peer-review
- I have already reviewed this code internally, no point in adding a +1 externally?
  - Please talk to your company about doing all code reviews in the public, that is a much better way to get involved. showing how the code has evolved upstream, is much better than trying to perfect code internally, before uploading for public review. You can use Draft mode, and mark things as WIP if you prefer, but please do the reviews upstream.
- Where do I start? What should I review?
  - There are various tools, but a good place to start is: https://etherpad.openstack.org/p/train-nova-subteam-tracking
  - Depending on the time in the cycle, its worth looking at NeedsCodeReview blueprints: https://blueprints.launchpad.net/nova/
  - Custom Gerrit review dashboards often provide a more manageable view of the outstanding reviews, and help focus your efforts:
    - Nova Review Inbox: https://goo.gl/1vTS0Z
    - Small Bug Fixes: http://ow.ly/WAw1J
  - Maybe take a look at things you want to see merged, bug fixes and features, or little code fixes
  - Look for things that have been waiting a long time for a review: https://review.opendev.org/#/q/project:openstack/nova+status:open+age:2weeks
  - If you get through the above lists, try other tools, such as: http://status.openstack.org/reviews

4.23.5 How to do great code reviews?

http://docs.openstack.org/infra/manual/developers.html#peer-review

For more tips, please see: *Why do code reviews if I am not in nova-core?*

4.23.6 How do I become nova-core?

You dont have to be nova-core to be a valued member of the Nova community. There are many, many ways you can help. Every quality review that helps someone get their patch closer to being ready to merge helps everyone get their code merged faster.

The first step to becoming nova-core is learning how to be an active member of the Nova community, including learning how to do great code reviews. For more details see: https://wiki.openstack.org/wiki/Nova/CoreTeam#Membership_Expectations

If you feel like you have the time to commit to all the nova-core membership expectations, reach out to the Nova PTL who will be able to find you an existing member of nova-core to help mentor you. If all goes well, and you seem like a good candidate, your mentor will contact the rest of the nova-core team to ask them to start looking at your reviews, so they are able to vote for you, if you get nominated for join nova-core.

We encourage all mentoring, where possible, to occur on #openstack-nova so everyone can learn and benefit from your discussions.

The above mentoring is available to every one who wants to learn how to better code reviews, even if you dont ever want to commit to becoming nova-core. If you already have a mentor, thats great, the process is only there for folks who are still trying to find a mentor. Being admitted to the mentoring program no way guarantees you will become a member of nova-core eventually, its here to help you improve, and help you have the sort of involvement and conversations that can lead to becoming a member of nova-core.

4.23.7 How to do great nova-spec reviews?

https://specs.openstack.org/openstack/nova-specs/specs/train/template.html

Blueprints, Specs and Priorities.

Spec reviews are always a step ahead of the normal code reviews. Follow the above links for some great information on specs/reviews.

The following could be some important tips:

1. The specs are published as html documents. Ensure that the author has a proper render of the same via the .rst file.

2. More often than not, its important to know that there are no overlaps across multiple specs.

3. Ensure that a proper dependency of the spec is identified. For example - a user desired feature that requires a proper base enablement should be a dependent spec.

4. Ask for clarity on changes that appear ambiguous to you.

5. Every release nova gets a huge set of spec proposals and thats a huge task for the limited set of nova cores to complete. Helping the cores with additional reviews is always a great thing.

4.23.8 How to do great bug triage?

https://wiki.openstack.org/wiki/Nova/BugTriage

Sylvain Bauza and Stephen Finucane gave a nice presentation on this topic at the Queens summit in Sydney.

4.23.9 How to step up into a project leadership role?

There are many ways to help lead the Nova project:

- Mentoring efforts, and getting started tips: https://wiki.openstack.org/wiki/Nova/Mentoring
- Info on process, with a focus on how you can go from an idea to getting code merged Nova: *Nova team process*
- Consider leading an existing Nova subteam or forming a new one.
- Consider becoming a Bug tag owner.
- Contact the PTL about becoming a Czar Nova People.

4.24 Nova team process

Nova is always evolving its processes, but its important to explain why we have them: so we can all work to ensure that the interactions we need to happen do happen. The process exists to make productive communication between all members of our community easier.

4.24.1 OpenStack Wide Patterns

Nova follows most of the generally adopted norms for OpenStack projects. You can get more details here:

- https://docs.openstack.org/infra/manual/developers.html
- https://docs.openstack.org/project-team-guide/

If you are new to Nova, please read this first: *How to get (more) involved with Nova*.

4.24.2 Dates overview

For Train, please see: https://wiki.openstack.org/wiki/Nova/Train_Release_Schedule

Feature Freeze

Feature freeze primarily provides a window of time to help the horizontal teams prepare their items for release, while giving developers time to focus on stabilising what is currently in master, and encouraging users and packagers to perform tests (automated, and manual) on the release, to spot any major bugs.

The Nova release process is aligned with the development cycle schedule used by many OpenStack projects, including the following steps.

- Feature Proposal Freeze
  - make sure all code is up for review
  - so we can optimise for completed features, not lots of half completed features
- Feature Freeze
  - make sure all feature code is merged
- String Freeze
  - give translators time to translate all our strings

  Note: debug logs are no longer translated

- Dependency Freeze
  - time to co-ordinate the final list of dependencies, and give packagers time to package them
  - generally it is also quite destabilising to take upgrades (beyond bug fixes) this late

As with all processes here, there are exceptions. The exceptions at this stage need to be discussed with the horizontal teams that might be affected by changes beyond this point, and as such are discussed with one of the OpenStack release managers.

Spec and Blueprint Approval Freeze

This is a (mostly) Nova specific process.

Why we have a Spec Freeze:

- specs take a long time to review and reviewing specs throughout the cycle distracts from code reviews
- keeping specs open and being slow at reviewing them (or just ignoring them) annoys the spec submitters

- we generally have more code submitted that we can review, this time bounding is a useful way to limit the number of submissions

By the freeze date, we expect all blueprints that will be approved for the cycle to be listed on launchpad and all relevant specs to be merged. For Train, blueprints can be found at https://blueprints.launchpad.net/nova/train and specs at https://specs.openstack.org/openstack/nova-specs/specs/train/index.html

Starting with Liberty, we are keeping a backlog open for submission at all times.

Note: The focus is on accepting and agreeing problem statements as being in scope, rather than queueing up work items for the next release. We are still working on a new lightweight process to get out of the backlog and approved for a particular release. For more details on backlog specs, please see: http://specs.openstack.org/openstack/nova-specs/specs/backlog/index.html

---

There can be exceptions, usually its an urgent feature request that comes up after the initial deadline. These will generally be discussed at the weekly Nova meeting, by adding the spec or blueprint to discuss in the appropriate place in the meeting agenda here (ideally make yourself available to discuss the blueprint, or alternatively make your case on the ML before the meeting): https://wiki.openstack.org/wiki/Meetings/Nova#Agenda_for_next_meeting

Non-priority Feature Freeze

This is a Nova specific process.

This only applies to low priority blueprints in this list: https://blueprints.launchpad.net/nova/train

We currently have a very finite amount of review bandwidth. In order to make code review time for the agreed community wide priorities, we have to not do some other things. In each cycle, milestones are used to bound when certain types of work will be active and reviewed and to avoid crushing the gate with too much code near the end of the cycle.

For example, in the Liberty cycle, we reserved the liberty-3 milestone for priority features and bug fixes and did not merge any non-priority things during liberty-3. This meant that liberty-2 was the Feature Freeze for blueprints that were not a priority for the Liberty cycle.

You can see the list of priorities for each release: http://specs.openstack.org/openstack/nova-specs/#priorities

For things that are very close to merging, its possible to request an exception for one week after the freeze date, given the patches get enough +2s from the core team to get the code merged. But we expect this list to be zero, if everything goes to plan (no massive gate failures, etc). For history of the process see: http://lists.openstack.org/pipermail/openstack-dev/2015-July/070920.html

Exception process:

- Please add request in here: https://etherpad.openstack.org/p/train-nova-non-priority-feature-freeze (ideally with core reviewers to sponsor your patch, normally the folks who have already viewed those patches)
- make sure you make your request before the end of the feature freeze exception period
- nova-drivers will meet to decide what gets an exception (for some history see: http://lists.openstack.org/pipermail/openstack-dev/2015-February/056208.html)
- an initial list of exceptions (probably just a PTL compiled list at that point) will be available for discussion during the next Nova meeting
- the aim is to merge the code for all exceptions early in the following week

Alternatives:

- It was hoped to make this a continuous process using slots to control what gets reviewed, but this was rejected by the community when it was last discussed. There is hope this can be resurrected to avoid the lumpy nature of this process.
- Currently the runways/kanban ideas are blocked on us adopting something like phabricator that could support such workflows

String Freeze

String Freeze provides an opportunity for translators to translate user-visible messages to a variety of languages. By not changing strings after the date of the string freeze, the job of the translators is made a bit easier. For more information on string and other OpenStack-wide release processes see the release management docs.

4.24.3 How do I get my code merged?

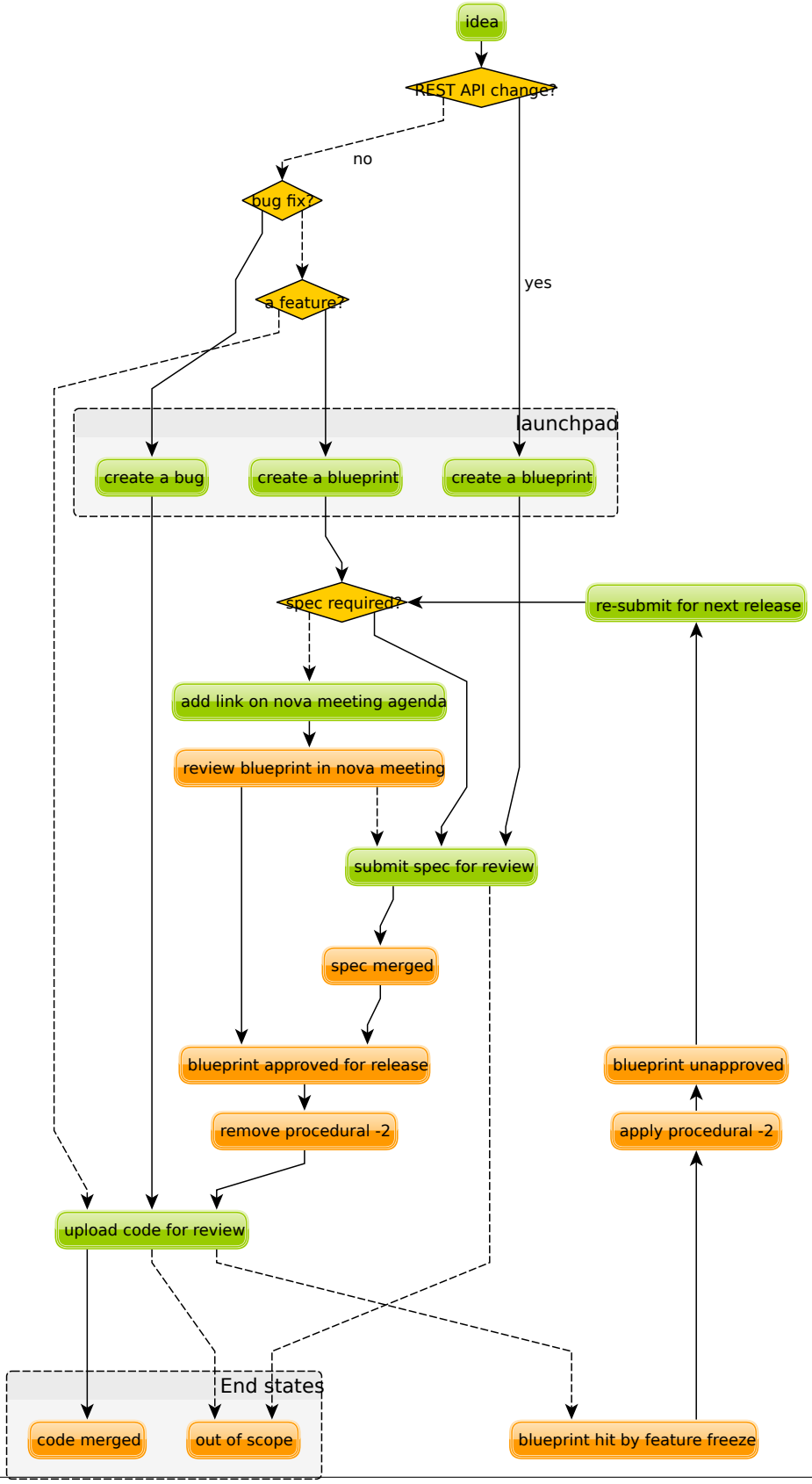
OK, so you are new to Nova, and you have been given a feature to implement. How do I make that happen?

You can get most of your questions answered here:

- https://docs.openstack.org/infra/manual/developers.html

But lets put a Nova specific twist on things

Overview


idea
REST API change?
no
yes
bug fix?
a feature?
launchpad
create a bug
create a blueprint
create a blueprint
spec required?
re-submit for next release
add link on nova meeting agenda
review blueprint in nova meeting
submit spec for review
spec merged
blueprint approved for release
blueprint unapproved
remove procedural -2
apply procedural -2
upload code for review
End states
code merged
out of scope
blueprint hit by feature freeze

Where do you track bugs?

We track bugs here:

- https://bugs.launchpad.net/nova

If you fix an issue, please raise a bug so others who spot that issue can find the fix you kindly created for them.

Also before submitting your patch its worth checking to see if someone has already fixed it for you (Launchpad helps you with that, at little, when you create the bug report).

When do I need a blueprint vs a spec?

For more details refer to *Blueprints, Specs and Priorities*.

To understand this question, we need to understand why blueprints and specs are useful.

But here is the rough idea:

- if it needs a spec, it will need a blueprint.
- if its an API change, it needs a spec.
- if its a single small patch that touches a small amount of code, with limited deployer and doc impact, it probably doesnt need a spec.

If you are unsure, please ask the PTL on IRC, or one of the other nova-drivers.

How do I get my blueprint approved?

So you need your blueprint approved? Here is how:

- if you dont need a spec, please add a link to your blueprint to the agenda for the next nova meeting: https://wiki.openstack.org/wiki/Meetings/Nova
  - be sure your blueprint description has enough context for the review in that meeting.
- if you need a spec, then please submit a nova-spec for review, see: https://docs.openstack.org/infra/manual/developers.html

Got any more questions? Contact the PTL or one of the other nova-specs-core who are awake at the same time as you. IRC is best as you will often get an immediate response, if they are too busy send him/her an email.

How do I get a procedural -2 removed from my patch?

When feature freeze hits, any patches for blueprints that are still in review get a procedural -2 to stop them merging. In Nova a blueprint is only approved for a single release. To have the -2 removed, you need to get the blueprint approved for the current release (see *How do I get my blueprint approved?*).

Why are the reviewers being mean to me?

Code reviews take intense concentration and a lot of time. This tends to lead to terse responses with very little preamble or nicety. That said, theres no excuse for being actively rude or mean. OpenStack has a Code of Conduct (https://www.openstack.org/legal/community-code-of-conduct/) and if you feel this has been breached please raise the matter privately. Either with the relevant parties, the PTL or failing those, the OpenStack Foundation.

That said, there are many objective reasons for applying a -1 or -2 to a patch:

- Firstly and simply, patches must address their intended purpose successfully.

- Patches must not have negative side-effects like wiping the database or causing a functional regression. Usually removing anything, however tiny, requires a deprecation warning be issued for a cycle.
- Code must be maintainable, that is it must adhere to coding standards and be as readable as possible for an average OpenStack developer (we acknowledge that this person is not easy to define).
- Patches must respect the direction of the project, for example they should not make approved specs substantially more difficult to implement.
- Release coordinators need the correct process to be followed so scope can be tracked accurately. Bug fixes require bugs, features require blueprints and all but the simplest features require specs. If there is a blueprint, it must be approved for the release/milestone the patch is attempting to merge into.

Please particularly bear in mind that a -2 does not mean never ever nor does it mean your idea is bad and you are dumb. It simply means do not merge today. You may need to wait some time, rethink your approach or even revisit the problem definition but there is almost always some way forward. The core who applied the -2 should tell you what you need to do.

My code review seems stuck, what can I do?

First and foremost - address any -1s and -2s! The review load on Nova is high enough that patches with negative reviews often get filtered out entirely. A few tips:

- Be precise. Ensure youre not talking at cross purposes.
- Try to understand where the reviewer is coming from. They may have a very different perspective and/or use-case to you.
- If you dont understand the problem, ask them to explain - this is common and helpful behaviour.
- Be positive. Everyones patches have issues, including core reviewers. No-one cares once the issues are fixed.
- Try not to flip-flop. When two reviewers are pulling you in different directions, stop pushing code and negotiate the best way forward.
- If the reviewer does not respond to replies left on the patchset, reach out to them on IRC or email. If they still dont respond, you can try to ask their colleagues if theyre on holiday (or simply wait). Finally, you can ask for mediation in the Nova meeting by adding it to the agenda (https://wiki.openstack.org/wiki/Meetings/Nova). This is also what you should do if you are unable to negotiate a resolution to an issue.

Secondly, Nova is a big project, look for things that have been waiting a long time for a review: https://review.opendev.org/#/q/project:openstack/nova+status:open+age:2weeks

Eventually you should get some +1s from people working through the review queue. Expect to get -1s as well. You can ask for reviews within your company, 1-2 are useful (not more), especially if those reviewers are known to give good reviews. You can spend some time while you wait reviewing other peoples code - they may reciprocate and you may learn something (*Why do code reviews when Im not core?*).

If youve waited an appropriate amount of time and you havent had any +1s, you can ask on IRC for reviews. Please dont ask for core review straight away, especially not directly (IRC or email). Core reviewer time is very valuable and gaining some +1s is a good way to show your patch meets basic quality standards.

Once you have a few +1s, be patient. Remember the average wait times. You can ask for reviews each week in IRC, it helps to ask when cores are awake.

Bugs

It helps to apply correct tracking information.

- Put Closes-Bug, Partial-Bug or Related-Bug in the commit message tags as necessary.

- If you have to raise a bug in Launchpad first, do it - this helps someone else find your fix.
- Make sure the bug has the correct priority and tag set.

Features

Again, it helps to apply correct tracking information. For blueprint-only features:

- Put your blueprint in the commit message, EG blueprint simple-feature.
- Mark the blueprint as NeedsCodeReview if you are finished.
- Maintain the whiteboard on the blueprint so its easy to understand which patches need reviews.
- Use a single topic for all related patches. All patches for one blueprint should share a topic.

For blueprint and spec features, do everything for blueprint-only features and also:

- If its a project or subteam priority, add it to: https://etherpad.openstack.org/p/train-nova-subteam-tracking
- Ensure your spec is approved for the current release cycle.

If your code is a project or subteam priority, the cores interested in that priority might not mind a ping after it has sat with +1s for a week. If you abuse this privilege, youll lose respect.

If its not a priority, your blueprint/spec has been approved for the cycle and you have been patient, you can raise it during the Nova meeting. The outcome may be that your spec gets unapproved for the cycle, so that priority items can take focus. If this happens to you, sorry - it should not have been approved in the first place, Nova team bit off more than they could chew, it is their mistake not yours. You can re-propose it for the next cycle.

If its not a priority and your spec has not been approved, your code will not merge this cycle. Please re-propose your spec for the next cycle.

4.24.4 Nova Process Mission

This section takes a high level look at the guiding principles behind the Nova process.

Open

Our mission is to have:

- Open Source
- Open Design
- Open Development
- Open Community

We have to work out how to keep communication open in all areas. We need to be welcoming and mentor new people, and make it easy for them to pickup the knowledge they need to get involved with OpenStack. For more info on Open, please see: https://wiki.openstack.org/wiki/Open

Interoperable API, supporting a vibrant ecosystem

An interoperable API that gives users on-demand access to compute resources is at the heart of *novas mission*.

Nova has a vibrant ecosystem of tools built on top of the current Nova API. All features should be designed to work with all technology combinations, so the feature can be adopted by our ecosystem. If a new feature is not adopted by

the ecosystem, it will make it hard for your users to make use of those features, defeating most of the reason to add the feature in the first place. The microversion system allows users to isolate themselves

This is a very different aim to being pluggable or wanting to expose all capabilities to end users. At the same time, it is not just a lowest common denominator set of APIs. It should be discoverable which features are available, and while no implementation details should leak to the end users, purely admin concepts may need to understand technology specific details that back the interoperable and more abstract concepts that are exposed to the end user. This is a hard goal, and one area we currently dont do well is isolating image creators from these technology specific details.

Smooth Upgrades

As part of our mission for a vibrant ecosystem around our APIs, we want to make it easy for those deploying Nova to upgrade with minimal impact to their users. Here is the scope of Novas upgrade support:

- upgrade from any commit, to any future commit, within the same major release
- only support upgrades between N and N+1 major versions, to reduce technical debt relating to upgrades

Here are some of the things we require developers to do, to help with upgrades:

- when replacing an existing feature or configuration option, make it clear how to transition to any replacement
- deprecate configuration options and features before removing them
  - i.e. continue to support and test features for at least one release before they are removed
  - this gives time for operator feedback on any removals
- End User API will always be kept backwards compatible

Interaction goals

When thinking about the importance of process, we should take a look at: http://agilemanifesto.org

With that in mind, lets look at how we want different members of the community to interact. Lets start with looking at issues we have tried to resolve in the past (currently in no particular order). We must:

- have a way for everyone to review blueprints and designs, including allowing for input from operators and all types of users (keep it open)
- take care to not expand Novas scope any more than absolutely necessary
- ensure we get sufficient focus on the core of Nova so that we can maintain or improve the stability and flexibility of the overall codebase
- support any API we release approximately forever. We currently release every commit, so were motivated to get the API right the first time
- avoid low priority blueprints that slow work on high priority work, without blocking those forever
- focus on a consistent experience for our users, rather than ease of development
- optimise for completed blueprints, rather than more half completed blueprints, so we get maximum value for our users out of our review bandwidth
- focus efforts on a subset of patches to allow our core reviewers to be more productive
- set realistic expectations on what can be reviewed in a particular cycle, to avoid sitting in an expensive rebase loop
- be aware of users that do not work on the project full time

- be aware of users that are only able to work on the project at certain times that may not align with the overall community cadence
- discuss designs for non-trivial work before implementing it, to avoid the expense of late-breaking design issues

4.24.5 FAQs

Why bother with all this process?

We are a large community, spread across multiple timezones, working with several horizontal teams. Good communication is a challenge and the processes we have are mostly there to try and help fix some communication challenges.

If you have a problem with a process, please engage with the community, discover the reasons behind our current process, and help fix the issues you are experiencing.

Why dont you remove old process?

We do! For example, in Liberty we stopped trying to predict the milestones when a feature will land.

As we evolve, it is important to unlearn new habits and explore if things get better if we choose to optimise for a different set of issues.

Why are specs useful?

Spec reviews allow anyone to step up and contribute to reviews, just like with code. Before we used gerrit, it was a very messy review process, that felt very closed to most people involved in that process.

As Nova has grown in size, it can be hard to work out how to modify Nova to meet your needs. Specs are a great way of having that discussion with the wider Nova community.

For Nova to be a success, we need to ensure we dont break our existing users. The spec template helps focus the mind on the impact your change might have on existing users and gives an opportunity to discuss the best way to deal with those issues.

However, there are some pitfalls with the process. Here are some top tips to avoid them:

- keep it simple. Shorter, simpler, more decomposed specs are quicker to review and merge much quicker (just like code patches).
- specs can help with documentation but they are only intended to document the design discussion rather than document the final code.
- dont add details that are best reviewed in code, its better to leave those things for the code review.

If we have specs, why still have blueprints?

We use specs to record the design agreement, we use blueprints to track progress on the implementation of the spec.

Currently, in Nova, specs are only approved for one release, and must be re-submitted for each release you want to merge the spec, although that is currently under review.

Why do we have priorities?

To be clear, there is no nova dev team manager, we are an open team of professional software developers, that all work for a variety of (mostly competing) companies that collaborate to ensure the Nova project is a success.

Over time, a lot of technical debt has accumulated, because there was a lack of collective ownership to solve those cross-cutting concerns. Before the Kilo release, it was noted that progress felt much slower, because we were unable to get appropriate attention on the architectural evolution of Nova. This was important, partly for major concerns like upgrades and stability. We agreed its something we all care about and it needs to be given priority to ensure that these things get fixed.

Since Kilo, priorities have been discussed at the summit. This turns in to a spec review which eventually means we get a list of priorities here: http://specs.openstack.org/openstack/nova-specs/#priorities

Allocating our finite review bandwidth to these efforts means we have to limit the reviews we do on non-priority items. This is mostly why we now have the non-priority Feature Freeze. For more on this, see below.

Blocking a priority effort is one of the few widely acceptable reasons to block someone adding a feature. One of the great advantages of being more explicit about that relationship is that people can step up to help review and/or implement the work that is needed to unblock the feature they want to get landed. This is a key part of being an Open community.

Why is there a Feature Freeze (and String Freeze) in Nova?

The main reason Nova has a feature freeze is that it allows people working on docs and translations to sync up with the latest code. Traditionally this happens at the same time across multiple projects, so the docs are synced between what used to be called the integrated release.

We also use this time period as an excuse to focus our development efforts on bug fixes, ideally lower risk bug fixes, and improving test coverage.

In theory, with a waterfall hat on, this would be a time for testing and stabilisation of the product. In Nova we have a much stronger focus on keeping every commit stable, by making use of extensive continuous testing. In reality, we frequently see the biggest influx of fixes in the few weeks after the release, as distributions do final testing of the released code.

It is hoped that the work on Feature Classification will lead us to better understand the levels of testing of different Nova features, so we will be able to reduce and dependency between Feature Freeze and regression testing. It is also likely that the move away from integrated releases will help find a more developer friendly approach to keep the docs and translations in sync.

Why is there a non-priority Feature Freeze in Nova?

We have already discussed why we have priority features.

The rate at which code can be merged to Nova is primarily constrained by the amount of time able to be spent reviewing code. Given this, earmarking review time for priority items means depriving it from non-priority items.

The simplest way to make space for the priority features is to stop reviewing and merging non-priority features for a whole milestone. The idea being developers should focus on bug fixes and priority features during that milestone, rather than working on non-priority features.

A known limitation of this approach is developer frustration. Many developers are not being given permission to review code, work on bug fixes or work on priority features, and so feel very unproductive upstream. An alternative approach of slots or runways has been considered, that uses a kanban style approach to regulate the influx of work onto the review queue. We are yet to get agreement on a more balanced approach, so the existing system is being continued to ensure priority items are more likely to get the attention they require.

Why do you still use Launchpad?

We are actively looking for an alternative to Launchpads bugs and blueprints.

Originally the idea was to create Storyboard. However development stalled for a while so interest waned. The project has become more active recently so it may be worth looking again: https://storyboard.openstack.org/#!/page/about

When should I submit my spec?

Ideally we want to get all specs for a release merged before the summit. For things that we cant get agreement on, we can then discuss those at the summit. There will always be ideas that come up at the summit and need to be finalised after the summit. This causes a rush which is best avoided.

How can I get my code merged faster?

So no-one is coming to review your code, how do you speed up that process?

Firstly, make sure you are following the above process. If its a feature, make sure you have an approved blueprint. If its a bug, make sure it is triaged, has its priority set correctly, it has the correct bug tag and is marked as in progress. If the blueprint has all the code up for review, change it from Started into NeedsCodeReview so people know only reviews are blocking you, make sure it hasnt accidentally got marked as implemented.

Secondly, if you have a negative review (-1 or -2) and you responded to that in a comment or uploading a new change with some updates, but that reviewer hasnt come back for over a week, its probably a good time to reach out to the reviewer on IRC (or via email) to see if they could look again now you have addressed their comments. If you cant get agreement, and your review gets stuck (i.e. requires mediation), you can raise your patch during the Nova meeting and we will try to resolve any disagreement.

Thirdly, is it in merge conflict with master or are any of the CI tests failing? Particularly any third-party CI tests that are relevant to the code you are changing. If youre fixing something that only occasionally failed before, maybe recheck a few times to prove the tests stay passing. Without green tests, reviewers tend to move on and look at the other patches that have the tests passing.

OK, so you have followed all the process (i.e. your patches are getting advertised via the projects tracking mechanisms), and your patches either have no reviews, or only positive reviews. Now what?

Have you considered reviewing other peoples patches? Firstly, participating in the review process is the best way for you to understand what reviewers are wanting to see in the code you are submitting. As you get more practiced at reviewing it will help you to write merge-ready code. Secondly, if you help review other peoples code and help get their patches ready for the core reviewers to add a +2, it will free up a lot of non-core and core reviewer time, so they are more likely to get time to review your code. For more details, please see: *Why do code reviews when Im not core?*

Please note, I am not recommending you go to ask people on IRC or via email for reviews. Please try to get your code reviewed using the above process first. In many cases multiple direct pings generate frustration on both sides and that tends to be counter productive.

Now you have got your code merged, lets make sure you dont need to fix this bug again. The fact the bug exists means there is a gap in our testing. Your patch should have included some good unit tests to stop the bug coming back. But dont stop there, maybe its time to add tempest tests, to make sure your use case keeps working? Maybe you need to set up a third party CI so your combination of drivers will keep working? Getting that extra testing in place should stop a whole heap of bugs, again giving reviewers more time to get to the issues or features you want to add in the future.

4.24.6 Process Evolution Ideas

We are always evolving our process as we try to improve and adapt to the changing shape of the community. Here we discuss some of the ideas, along with their pros and cons.

Splitting out the virt drivers (or other bits of code)

Currently, Nova doesnt have strong enough interfaces to split out the virt drivers, scheduler or REST API. This is seen as the key blocker. Lets look at both sides of the debate here.

Reasons for the split:

- can have separate core teams for each repo
  - this leads to quicker turn around times, largely due to focused teams
- splitting out things from core means less knowledge required to become core in a specific area

Reasons against the split:

- loss of interoperability between drivers
  - this is a core part of Novas mission, to have a single API across all deployments, and a strong ecosystem of tools and apps built on that
  - we can overcome some of this with stronger interfaces and functional tests
- new features often need changes in the API and virt driver anyway
  - the new depends-on can make these cross-repo dependencies easier
- loss of code style consistency across the code base
- fear of fragmenting the nova community, leaving few to work on the core of the project
- could work in subteams within the main tree

TODO - need to complete analysis

Subteam recommendation as a +2

There are groups of people with great knowledge of particular bits of the code base. It may be a good idea to give their recommendation of a merge greater strength. In addition, having the subteam focus review efforts on a subset of patches should help concentrate the nova-core reviews they get, and increase the velocity of getting code merged.

The first part is for subgroups to show they can do a great job of recommending patches. This is starting in here: https://etherpad.openstack.org/p/train-nova-subteam-tracking

Ideally this would be done with gerrit user tags rather than an etherpad. There are some investigations by sdague in how feasible it would be to add tags to gerrit.

Stop having to submit a spec for each release

As mentioned above, we use blueprints for tracking, and specs to record design decisions. Targeting specs to a specific release is a heavyweight solution and blurs the lines between specs and blueprints. At the same time, we dont want to lose the opportunity to revise existing blueprints. Maybe there is a better balance?

What about this kind of process:

- backlog has these folders:
  - backlog/incomplete - merge a partial spec
  - backlog/complete - merge complete specs (remove tracking details, such as assignee part of the template)
  - ?? backlog/expired - specs are moved here from incomplete or complete when no longer seem to be given attention (after 1 year, by default)

  - /implemented - when a spec is complete it gets moved into the release directory and possibly updated to reflect what actually happened
  - there will no longer be a per-release approved spec list

To get your blueprint approved:

- add it to the next nova meeting
  - if a spec is required, update the URL to point to the spec merged in a spec to the blueprint
  - ensure there is an assignee in the blueprint
- a day before the meeting, a note is sent to the ML to review the list before the meeting
- discuss any final objections in the nova-meeting
  - this may result in a request to refine the spec, if things have changed since it was merged
- trivial cases can be approved in advance by a nova-driver, so not all folks need to go through the meeting

This still needs more thought, but should decouple the spec review from the release process. It is also more compatible with a runway style system, that might be less focused on milestones.

Runways

Runways are a form of Kanban, where we look at optimising the flow through the system, by ensuring we focus our efforts on reviewing a specific subset of patches.

The idea goes something like this:

- define some states, such as: design backlog, design review, code backlog, code review, test+doc backlog, complete
- blueprints must be in one of the above state
  - large or high priority bugs may also occupy a code review slot
- core reviewer member moves item between the slots
  - must not violate the rules on the number of items in each state
  - states have a limited number of slots, to ensure focus
  - certain percentage of slots are dedicated to priorities, depending on point in the cycle, and the type of the cycle, etc

Reasons for:

- more focused review effort, get more things merged more quickly
- more upfront about when your code is likely to get reviewed
- smooth out current lumpy non-priority feature freeze system

Reasons against:

- feels like more process overhead
- control is too centralised

Replacing Milestones with SemVer Releases

You can deploy any commit of Nova and upgrade to a later commit in that same release. Making our milestones versioned more like an official release would help signal to our users that people can use the milestones in production, and get a level of upgrade support.

It could go something like this:

- 14.0.0 is milestone 1
- 14.0.1 is milestone 2 (maybe, because we add features, it should be 14.1.0?)
- 14.0.2 is milestone 3
- we might do other releases (once a critical bug is fixed?), as it makes sense, but we will always be the time bound ones
- 14.0.3 two weeks after milestone 3, adds only bug fixes (and updates to RPC versions?)
  - maybe a stable branch is created at this point?
- 14.1.0 adds updated translations and co-ordinated docs
  - this is released from the stable branch?
- 15.0.0 is the next milestone, in the following cycle
  - not the bump of the major version to signal an upgrade incompatibility with 13.x

We are currently watching Ironic to see how their use of semver goes, and see what lessons need to be learnt before we look to maybe apply this technique during M.

Feature Classification

This is a look at moving forward the *support matrix effort*.

The things we need to cover:

- note what is tested, and how often that test passes (via 3rd party CI, or otherwise)
  - link to current test results for stable and master (time since last pass, recent pass rate, etc)
  - TODO - sync with jogo on his third party CI audit and getting trends, ask infra
- include experimental features (untested feature)
- get better at the impact of volume drivers and network drivers on available features (not just hypervisor drivers)

Main benefits:

- users get a clear picture of what is known to work
- be clear about when experimental features are removed, if no tests are added
- allows a way to add experimental things into Nova, and track either their removal or maturation

4.25 Scope of the Nova project

Nova is focusing on doing an awesome job of its core mission. This document aims to clarify that core mission.

This is a living document to help record where we agree about what Nova should and should not be doing, and why. Please treat this as a discussion of interesting, and hopefully useful, examples. It is not intended to be an exhaustive policy statement.

4.25.1 Mission

Our mission statement starts with:

> To implement services and associated libraries to provide massively scalable, on demand, self service access to compute resources.

Our official mission statement also includes the following examples of compute resources: bare metal, virtual machines, and containers. For the full official mission statement see: https://governance.openstack.org/tc/reference/projects/nova.html#mission

This document aims to help clarify what the mission statement means.

4.25.2 Compute Resources

Nova is all about access to compute resources. This section looks at the types of compute resource Nova works with.

Virtual Servers

Nova was originally focused purely on providing access to virtual servers running on a variety of different hypervisors. The majority of users use Nova only to provide access to virtual servers from a single hypervisor, however, its possible to have a Nova deployment include multiple different types of hypervisors, while at the same time offering containers and bare metal servers.

Containers

The Nova API is not a good fit for a lot of container use cases. The Magnum project intends to deliver a good container experience built on top of Nova.

Nova allows you to use containers in a similar way to how you would use on demand virtual machines. We want to maintain this distinction, so we maintain the integrity and usefulness of the existing Nova API.

For example, Nova is not designed to spin up new containers for every apache request, nor do we plan to control what goes on inside containers. They get the same metadata provided to them as virtual machines, to do with as they see fit.

Bare Metal Servers

Ironic project has been pioneering the idea of treating physical machines in a similar way to on demand virtual machines.

Novas driver is able to allow a multi-tenant cloud style use of Ironic controlled resources.

While currently there are operations that are a fundamental part of our virtual machine abstraction that are not currently available in ironic, such as attaching iSCSI volumes, it does not fundamentally change the semantics of our API, and as such is a suitable Nova driver. Moreover, it is expected that gap with shrink over time.

Driver Parity

Our goal for the Nova API is to provide a consistent abstraction to access on demand compute resources. We are not aiming to expose all features of all hypervisors. Where the details of the underlying hypervisor leak through our APIs, we have failed in this goal, and we must work towards better abstractions that are more interoperable. This is one reason why we put so much emphasis on the use of Tempest in third party CI systems.

The key tenet of driver parity is that if a feature is supported in a driver, it must feel the same to users, as if they where using any of the other drivers that also support that feature. The exception is that, if possible for widely different performance characteristics, but the effect of that API call must be identical.

Following on from that, should a feature only be added to one of the drivers, we must make every effort to ensure another driver could be implemented to match that behavior.

It is important that drivers support enough features, so the API actually provides a consistent abstraction. For example, being unable to create a server or delete a server would severely undermine that goal. In fact, Nova only ever manages resources it creates.

4.25.3 Upgrades

Nova is widely used in production. As such we need to respect the needs of our existing users. At the same time we need evolve the current code base, including both adding and removing features.

This section outlines how we expect people to upgrade, and what we do to help existing users that upgrade in the way we expect.

Upgrade expectations

Our upgrade plan is to concentrate on upgrades from N-1 to the Nth release. So for someone running juno, they would have to upgrade to kilo before upgrading to liberty. This is designed to balance the need for a smooth upgrade, against having to keep maintaining the compatibility code to make that upgrade possible. We talk about this approach as users consuming the stable branch.

In addition, we also support users upgrading from the master branch, technically, between any two commits within the same release cycle. In certain cases, when crossing release boundaries, you must upgrade to the stable branch, before upgrading to the tip of master. This is to support those that are doing some level of Continuous Deployment from the tip of master into production. Many of the public cloud provides running OpenStack use this approach so they are able to get access to bug fixes and features they work on into production sooner.

This becomes important when you consider reverting a commit that turns out to have been bad idea. We have to assume any public API change may have already been deployed into production, and as such cannot be reverted. In a similar way, a database migration may have been deployed.

Any commit that will affect an upgrade gets the UpgradeImpact tag added to the commit message, so there is no requirement to wait for release notes.

Dont break existing users

As a community we are aiming towards a smooth upgrade process, where users must be unaware you have just upgraded your deployment, except that there might be additional feature available and improved stability and performance of some existing features.

We dont ever want to remove features our users rely on. Sometimes we need to migrate users to a new implementation of that feature, which may require extra steps by the deployer, but the end users must be unaffected by such changes. However there are times when some features become a problem to maintain, and fall into disrepair. We aim to be honest with our users and highlight the issues we have, so we are in a position to find help to fix that situation. Ideally we are able to rework the feature so it can be maintained, but in some rare cases, the feature no longer works, is not tested, and no one is stepping forward to maintain that feature, the best option can be to remove that feature.

When we remove features, we need to warn users by first marking those features as deprecated, before we finally remove the feature. The idea is to get feedback on how important the feature is to our user base. Where a feature is important we work with the whole community to find a path forward for those users.

4.25.4 API Scope

Nova aims to provide a highly interoperable and stable REST API for our users to get self-service access to compute resources.

No more API Proxies

Nova API current has some APIs that are now (in kilo) mostly just a proxy to other OpenStack services. If it were possible to remove a public API, these are some we might start with. As such, we dont want to add any more.

The first example is the API that is a proxy to the Glance v1 API. As Glance moves to deprecate its v1 API, we need to translate calls from the old v1 API we expose, to Glances v2 API.

The next API to mention is the networking APIs, in particular the security groups API. If you are using nova-network, Nova is still the only way to perform these network operations. But if you use Neutron, security groups has a much richer Neutron API, and if you use both Nova API and Neutron API, the miss match can lead to some very unexpected results, in certain cases.

Our intention is to avoid adding to the problems we already have in this area.

No more Orchestration

Nova is a low level infrastructure API. It is plumbing upon which richer ideas can be built. Heat and Magnum being great examples of that.

While we have some APIs that could be considered orchestration, and we must continue to maintain those, we do not intend to add any more APIs that do orchestration.

Third Party APIs

Nova aims to focus on making a great API that is highly interoperable across all Nova deployments.

We have historically done a very poor job of implementing and maintaining compatibility with third party APIs inside the Nova tree.

As such, all new efforts should instead focus on external projects that provide third party compatibility on top of the Nova API. Where needed, we will work with those projects to extend the Nova API such that its possible to add that functionality on top of the Nova API. However, we do not intend to add API calls for those services to persist third party API specific information in the Nova database. Instead we want to focus on additions that enhance the existing Nova API.

4.25.5 Scalability

Our mission includes the text massively scalable. Lets discuss what that means.

Nova has three main axes of scale: Number of API requests, number of compute nodes and number of active instances. In many cases the number of compute nodes and active instances are so closely related, you rarely need to consider those separately. There are other items, such as the number of tenants, and the number of instances per tenant. But, again, these are very rarely the key scale issue. Its possible to have a small cloud with lots of requests for very short lived VMs, or a large cloud with lots of longer lived VMs. These need to scale out different components of the Nova system to reach their required level of scale.

Ideally all Nova components are either scaled out to match the number of API requests and build requests, or scaled out to match the number of running servers. If we create components that have their load increased relative to both of

these items, we can run into inefficiencies or resource contention. Although it is possible to make that work in some cases, this should always be considered.

We intend Nova to be usable for both small and massive deployments. Where small involves 1-10 hypervisors and massive deployments are single regions with greater than 10,000 hypervisors. That should be seen as our current goal, not an upper limit.

There are some features that would not scale well for either the small scale or the very large scale. Ideally we would not accept these features, but if there is a strong case to add such features, we must work hard to ensure you can run without that feature at the scale you are required to run.

4.25.6 IaaS not Batch Processing

Currently Nova focuses on providing on-demand compute resources in the style of classic Infrastructure-as-a-service clouds. A large pool of compute resources that people can consume in a self-service way.

Nova is not currently optimized for dealing with a larger number of requests for compute resources compared with the amount of compute resources currently available. We generally assume that a level of spare capacity is maintained for future requests. This is needed for users who want to quickly scale out, and extra capacity becomes available again as users scale in. While spare capacity is also not required, we are not optimizing for a system that aims to run at 100% capacity at all times. As such our quota system is more focused on limiting the current level of resource usage, rather than ensuring a fair balance of resources between all incoming requests. This doesnt exclude adding features to support making a better use of spare capacity, such as spot instances.

There have been discussions around how to change Nova to work better for batch job processing. But the current focus is on how to layer such an abstraction on top of the basic primitives Nova currently provides, possibly adding additional APIs where that makes good sense. Should this turn out to be impractical, we may have to revise our approach.

4.25.7 Deployment and Packaging

Nova does not plan on creating its own packaging or deployment systems.

Our CI infrastructure is powered by Devstack. This can also be used by developers to test their work on a full deployment of Nova.

We do not develop any deployment or packaging for production deployments. Being widely adopted by many distributions and commercial products, we instead choose to work with all those parties to ensure they are able to effectively package and deploy Nova.

4.26 Chronological PTL guide

This is just a reference guide that a PTL may use as an aid, if they choose.

4.26.1 New PTL

- Update the nova meeting chair
  - https://github.com/openstack-infra/irc-meetings/blob/master/meetings/nova-team-meeting.yaml
- Update the team wiki
  - https://wiki.openstack.org/wiki/Nova#People
- Get acquainted with the release schedule

  - Example: https://wiki.openstack.org/wiki/Nova/Stein_Release_Schedule

4.26.2 Project Team Gathering

- Create PTG planning etherpad, retrospective etherpad and alert about it in nova meeting and dev mailing list
  - Example: https://etherpad.openstack.org/p/nova-ptg-stein
- Run sessions at the PTG
- Have a priorities discussion at the PTG
  - Example: https://etherpad.openstack.org/p/nova-ptg-stein-priorities
- Sign up for group photo at the PTG (if applicable)
- Open review runways for the cycle
  - Example: https://etherpad.openstack.org/p/nova-runways-stein

4.26.3 After PTG

- Send PTG session summaries to the dev mailing list
- Make sure the cycle priorities spec gets reviewed and merged
  - Example: https://specs.openstack.org/openstack/nova-specs/priorities/stein-priorities.html
- Run the count-blueprints script daily to gather data for the cycle burndown chart

4.26.4 A few weeks before milestone 1

- Plan a spec review day
- Periodically check the series goals others have proposed in the Set series goals link:
  - Example: https://blueprints.launchpad.net/nova/stein/+setgoals

4.26.5 Milestone 1

- Do milestone release of nova and python-novaclient (in launchpad only)
  - This is launchpad bookkeeping only. With the latest release team changes, projects no longer do milestone releases. See: https://releases.openstack.org/reference/release_models.html#cycle-with-milestones-legacy
  - For nova, set the launchpad milestone release as released with the date
- Release other libraries if there are significant enough changes since last release. When releasing the first version of a library for the cycle, bump the minor version to leave room for future stable branch releases
  - os-vif
- Release stable branches of nova
  - `git checkout <stable branch>`
  - `git log --no-merges <last tag>..`
    - Examine commits that will go into the release and use it to decide whether the release is a major, minor, or revision bump according to semver

  - Then, propose the release with version according to semver x.y.z
    * X - backward-incompatible changes
    * Y - features
    * Z - bug fixes
  - Use the new-release command to generate the release
    * https://releases.openstack.org/reference/using.html#using-new-release-command

4.26.6 Summit

- Prepare the project update presentation. Enlist help of others
- Prepare the on-boarding session materials. Enlist help of others

4.26.7 A few weeks before milestone 2

- Plan a spec review day (optional)
- Periodically check the series goals others have proposed in the Set series goals link:
  - Example: https://blueprints.launchpad.net/nova/stein/+setgoals

4.26.8 Milestone 2

- Spec freeze
- Release nova and python-novaclient
- Release other libraries as needed
- Stable branch releases of nova
- For nova, set the launchpad milestone release as released with the date

4.26.9 Shortly after spec freeze

- Create a blueprint status etherpad to help track, especially non-priority blueprint work, to help things get done by Feature Freeze (FF). Example:
  - https://etherpad.openstack.org/p/nova-stein-blueprint-status
- Create or review a patch to add the next releases specs directory so people can propose specs for next release after spec freeze for current release

4.26.10 Non-client library release freeze

- Final release for os-vif

4.26.11 Milestone 3

- Feature freeze day
- Client library freeze, release python-novaclient
- Close out all blueprints, including catch all blueprints like mox, versioned notifications
- Stable branch releases of nova
- For nova, set the launchpad milestone release as released with the date

4.26.12 Week following milestone 3

- Consider announcing the FFE (feature freeze exception process) to have people propose FFE requests to a special etherpad where they will be reviewed and possibly sponsored:
  - https://docs.openstack.org/nova/latest/contributor/process.html#non-priority-feature-freeze
  - Note: if there is only a short time between FF and RC1 (lately its been 2 weeks), then the only likely candidates will be low-risk things that are almost done

4.26.13 A few weeks before RC

- Make a RC1 todos etherpad and tag bugs as `<release>-rc-potential` and keep track of them, example:
  - https://etherpad.openstack.org/p/nova-stein-rc-potential
- Go through the bug list and identify any rc-potential bugs and tag them

4.26.14 RC

- Do steps described on the release checklist wiki:
  - https://wiki.openstack.org/wiki/Nova/ReleaseChecklist
- If we want to drop backward-compat RPC code, we have to do a major RPC version bump and coordinate it just before the major release:
  - https://wiki.openstack.org/wiki/RpcMajorVersionUpdates
  - Example: https://review.opendev.org/541035
- Merge latest translations means translation patches
  - Check for translations with:
    - https://review.opendev.org/#/q/status:open+project:openstack/nova+branch:master+topic:zanata/translations
- Should NOT plan to have more than one RC if possible. RC2 should only happen if there was a mistake and something was missed for RC, or a new regression was discovered
- Do the RPC version aliases just before RC1 if no further RCs are planned. Else do them at RC2. In the past, we used to update all service version aliases (example: https://review.opendev.org/230132) but since we really only support compute being backlevel/old during a rolling upgrade, we only need to update the compute service alias, see related IRC discussion: http://eavesdrop.openstack.org/irclogs/%23openstack-nova/%23openstack-nova.2018-08-08.log.html#t2018-08-08T17:13:45
  - Example: https://review.opendev.org/642599

  - More detail on how version aliases work: https://docs.openstack.org/nova/latest/configuration/config.html#upgrade-levels

- Write the reno prelude for the release GA
  - Example: https://review.opendev.org/644412
- Write the cycle-highlights in marketing-friendly sentences and propose to the openstack/releases repo. Usually based on reno prelude but made more readable and friendly
  - Example: https://review.opendev.org/644697

4.26.15 Immediately after RC

- Look for bot proposed changes to reno and stable/<cycle>
- Follow the post-release checklist
  - https://wiki.openstack.org/wiki/Nova/ReleaseChecklist
  - Add database migration placeholders
    - Example: https://review.openstack.org/650964
  - Drop old RPC compat code (if there was a RPC major version bump)
    - Example: https://review.openstack.org/543580
- Create the launchpad series for the next cycle
- Set the development focus of the project to the new cycle series
- Set the status of the new series to active development
- Set the last series status to current stable branch release
- Set the previous to last series status to supported
- Repeat launchpad steps ^ for python-novaclient
- Register milestones in launchpad for the new cycle based on the new cycle release schedule
- Make sure the specs directory for the next cycle gets created so people can start proposing new specs
- Make sure to move implemented specs from the previous release
  - Use `tox -e move-implemented-specs <release>`
  - Also remove template from `doc/source/specs/<release>/index.rst`
  - Also delete template file `doc/source/specs/<release>/template.rst`
- Create new release wiki:
  - Example: https://wiki.openstack.org/wiki/Nova/Train_Release_Schedule

4.26.16 Miscellaneous Notes

How to approve a launchpad blueprint

- Set the approver as the person who +W the spec, or set to self if its specless
- Set the Direction => Approved and Definition => Approved and make sure the Series goal is set to the current release. If code is already proposed, set Implementation => Needs Code Review

- Add a comment to the Whiteboard explaining the approval, with a date (launchpad does not record approval dates). For example: We discussed this in the team meeting and agreed to approve this for <release>. – <nick> <YYYYMMDD>

How to complete a launchpad blueprint

- Set Implementation => Implemented. The completion date will be recorded by launchpad

4.27 REST API Version History

This documents the changes made to the REST API with every microversion change. The description for each version should be a verbose one which has enough information to be suitable for use in user documentation.

4.27.1 2.1

This is the initial version of the v2.1 API which supports microversions. The V2.1 API is from the REST API userss point of view exactly the same as v2.0 except with strong input validation.

A user can specify a header in the API request:

```
X-OpenStack-Nova-API-Version: <version>
```

where `<version>` is any valid api version for this API.

If no version is specified then the API will behave as if a version request of v2.1 was requested.

4.27.2 2.2

Added Keypair type.

A user can request the creation of a certain type of keypair (`ssh` or `x509`) in the `os-keypairs` plugin

If no keypair type is specified, then the default `ssh` type of keypair is created.

Fixes status code for `os-keypairs` create method from 200 to 201

Fixes status code for `os-keypairs` delete method from 202 to 204

4.27.3 2.3 (Maximum in Kilo)

Exposed additional attributes in `os-extended-server-attributes`: `reservation_id`, `launch_index`, `ramdisk_id`, `kernel_id`, `hostname`, `root_device_name`, `userdata`.

Exposed `delete_on_termination` for `volumes_attached` in `os-extended-volumes`.

This change is required for the extraction of EC2 API into a standalone service. It exposes necessary properties absent in public nova APIs yet. Add info for Standalone EC2 API to cut access to Nova DB.

4.27.4 2.4

Show the `reserved` status on a `FixedIP` object in the `os-fixed-ips` API extension. The extension allows one to `reserve` and `unreserve` a fixed IP but the show method does not report the current status.

4.27.5 2.5

Before version 2.5, the command `nova list --ip6 xxx` returns all servers for non-admins, as the filter option is silently discarded. There is no reason to treat ip6 different from ip, though, so we just add this option to the allowed list.

4.27.6 2.6

A new API for getting remote console is added:

```
POST /servers/<uuid>/remote-consoles
{
  "remote_console": {
    "protocol": ["vnc"|"rdp"|"serial"|"spice"],
    "type": ["novnc"|"xpvnc"|"rdp-html5"|"spice-html5"|"serial"]
  }
}
```

Example response:

```
{
  "remote_console": {
    "protocol": "vnc",
    "type": "novnc",
    "url": "http://example.com:6080/vnc_auto.html?path=%3Ftoken%3DXYZ"
  }
}
```

The old APIs `os-getVNCConsole`, `os-getSPICEConsole`, `os-getSerialConsole` and `os-getRDPConsole` are removed.

4.27.7 2.7

Check the `is_public` attribute of a flavor before adding tenant access to it. Reject the request with `HTTPConflict` error.

4.27.8 2.8

Add `mks` protocol and `webmks` type for remote consoles.

4.27.9 2.9

Add a new `locked` attribute to the detailed view, update, and rebuild action. `locked` will be `true` if anyone is currently holding a lock on the server, `false` otherwise.

4.27.10 2.10

Added `user_id` parameter to `os-keypairs` plugin, as well as a new property in the request body, for the create operation.

Administrators will be able to list, get details and delete keypairs owned by users other than themselves and to create new keypairs on behalf of their users.

4.27.11 2.11

Exposed attribute `forced_down` for `os-services`. Added ability to change the `forced_down` attribute by calling an update.

4.27.12 2.12 (Maximum in Liberty)

Exposes VIF `net_id` attribute in `os-virtual-interfaces`. User will be able to get Virtual Interfaces `net_id` in Virtual Interfaces list and can determine in which network a Virtual Interface is plugged into.

4.27.13 2.13

Add information `project_id` and `user_id` to `os-server-groups` API response data.

4.27.14 2.14

Remove `onSharedStorage` parameter from servers evacuate action. Nova will automatically detect if the instance is on shared storage.

`adminPass` is removed from the response body. The user can get the password with the servers `os-server-password` action.

4.27.15 2.15

From this version of the API users can choose soft-affinity and soft-anti-affinity rules too for server-groups.

4.27.16 2.16

Exposes new `host_status` attribute for servers/detail and servers/{server_id}. Ability to get nova-compute status when querying servers. By default, this is only exposed to cloud administrators.

4.27.17 2.17

Add a new API for triggering crash dump in an instance. Different operation systems in instance may need different configurations to trigger crash dump.

4.27.18 2.18

Establishes a set of routes that makes project_id an optional construct in v2.1.

4.27.19 2.19

Allow the user to set and get the server description. The user will be able to set the description when creating, rebuilding, or updating a server, and get the description as part of the server details.

4.27.20 2.20

From this version of the API user can call detach and attach volumes for instances which are in `shelved` and `shelved_offloaded` state.

4.27.21 2.21

The `os-instance-actions` API now returns information from deleted instances.

4.27.22 2.22

A new resource, `servers:migrations`, is added. A new API to force live migration to complete added:

```
POST /servers/<uuid>/migrations/<id>/action
{
  "force_complete": null
}
```

4.27.23 2.23

From this version of the API users can get the migration summary list by index API or the information of a specific migration by get API. And the old top-level resource `/os-migrations` wont be extended anymore. Add `migration_type` for old `/os-migrations` API, also add `ref` link to the `/servers/{uuid}/migrations/{id}` for it when the migration is an in-progress live-migration.

4.27.24 2.24

A new API call to cancel a running live migration:

```
DELETE /servers/<uuid>/migrations/<id>
```

4.27.25 2.25 (Maximum in Mitaka)

Modify input parameter for `os-migrateLive`. The `block_migration` field now supports an `auto` value and the `disk_over_commit` flag is removed.

4.27.26 2.26

Added support of server tags.

A user can create, update, delete or check existence of simple string tags for servers by the `os-server-tags` plugin.

Tags have the following schema restrictions:

- Tag is a Unicode bytestring no longer than 60 characters.
- Tag is a non-empty string.
- / is not allowed to be in a tag name
- Comma is not allowed to be in a tag name in order to simplify requests that specify lists of tags

- All other characters are allowed to be in a tag name
- Each server can have up to 50 tags.

The resource point for these operations is `/servers/<server_id>/tags`.

A user can add a single tag to the server by making a `PUT` request to `/servers/<server_id>/tags/<tag>`.

where `<tag>` is any valid tag name.

A user can replace **all** current server tags to the new set of tags by making a `PUT` request to the `/servers/<server_id>/tags`. The new set of tags must be specified in request body. This set must be in list `tags`.

A user can remove specified tag from the server by making a `DELETE` request to `/servers/<server_id>/tags/<tag>`.

where `<tag>` is tag name which user wants to remove.

A user can remove **all** tags from the server by making a `DELETE` request to the `/servers/<server_id>/tags`.

A user can get a set of server tags with information about server by making a `GET` request to `/servers/<server_id>`.

Request returns dictionary with information about specified server, including list `tags`:

```
{
    'id': {server_id},
    ...
    'tags': ['foo', 'bar', 'baz']
}
```

A user can get **only** a set of server tags by making a `GET` request to `/servers/<server_id>/tags`.

Response

```
{
   'tags': ['foo', 'bar', 'baz']
}
```

A user can check if a tag exists or not on a server by making a `GET` request to `/servers/{server_id}/tags/{tag}`.

Request returns `204 No Content` if tag exist on a server or `404 Not Found` if tag doesnt exist on a server.

A user can filter servers in `GET /servers` request by new filters:

- `tags`
- `tags-any`
- `not-tags`
- `not-tags-any`

These filters can be combined. Also user can use more than one string tags for each filter. In this case string tags for each filter must be separated by comma. For example:

```
GET /servers?tags=red&tags-any=green,orange
```

4.27.27 2.27

Added support for the new form of microversion headers described in the Microversion Specification. Both the original form of header and the new form is supported.

4.27.28 2.28

Nova API `hypervisor.cpu_info` change from string to JSON object.

From this version of the API the hypervisors `cpu_info` field will be returned as JSON object (not string) by sending GET request to the `/v2.1/os-hypervisors/{hypervisor_id}`.

4.27.29 2.29

Updates the POST request body for the `evacuate` action to include the optional `force` boolean field defaulted to False. Also changes the evacuate action behaviour when providing a `host` string field by calling the nova scheduler to verify the provided host unless the `force` attribute is set.

4.27.30 2.30

Updates the POST request body for the `live-migrate` action to include the optional `force` boolean field defaulted to False. Also changes the live-migrate action behaviour when providing a `host` string field by calling the nova scheduler to verify the provided host unless the `force` attribute is set.

4.27.31 2.31

Fix `os-console-auth-tokens` to return connection info for all types of tokens, not just RDP.

4.27.32 2.32

Adds an optional, arbitrary tag item to the networks item in the server boot request body. In addition, every item in the block_device_mapping_v2 array can also have an optional, arbitrary tag item. These tags are used to identify virtual device metadata, as exposed in the metadata API and on the config drive. For example, a network interface on the virtual PCI bus tagged with nic1 will appear in the metadata along with its bus (PCI), bus address (ex: 0000:00:02.0), MAC address, and tag (nic1).

Note: A bug has caused the tag attribute to no longer be accepted for networks starting with version 2.37 and for block_device_mapping_v2 starting with version 2.33. In other words, networks could only be tagged between versions 2.32 and 2.36 inclusively and block devices only in version 2.32. As of version 2.42 the tag attribute has been restored and both networks and block devices can be tagged again.

4.27.33 2.33

Support pagination for hypervisor by accepting limit and marker from the GET API request:

```
GET /v2.1/{tenant_id}/os-hypervisors?marker={hypervisor_id}&limit={limit}
```

In the context of device tagging at server create time, 2.33 also removes the tag attribute from block_device_mapping_v2. This is a bug that is fixed in 2.42, in which the tag attribute is reintroduced.

4.27.34 2.34

Checks in `os-migrateLive` before live-migration actually starts are now made in background. `os-migrateLive` is not throwing *400 Bad Request* if pre-live-migration checks fail.

4.27.35 2.35

Added pagination support for keypairs.

Optional parameters limit and marker were added to GET /os-keypairs request, the default sort_key was changed to name field as ASC order, the generic request format is:

```
GET /os-keypairs?limit={limit}&marker={kp_name}
```

4.27.36 2.36

All the APIs which proxy to another service were deprecated in this version, also the fping API. Those APIs will return 404 with Microversion 2.36. The network related quotas and limits are removed from API also. The deprecated API endpoints as below:

```
'/images'
'/os-networks'
'/os-tenant-networks'
'/os-fixed-ips'
'/os-floating-ips'
'/os-floating-ips-bulk'
'/os-floating-ip-pools'
'/os-floating-ip-dns'
'/os-security-groups'
'/os-security-group-rules'
'/os-security-group-default-rules'
'/os-volumes'
'/os-snapshots'
'/os-baremetal-nodes'
'/os-fping'
```

Note: A regression was introduced in this microversion which broke the `force` parameter in the `PUT /os-quota-sets` API. The fix will have to be applied to restore this functionality.

4.27.37 2.37

Added support for automatic allocation of networking, also known as Get Me a Network. With this microversion, when requesting the creation of a new server (or servers) the `networks` entry in the `server` portion of the request body is required. The `networks` object in the request can either be a list or an enum with values:

1. *none* which means no networking will be allocated for the created server(s).
2. *auto* which means either a network that is already available to the project will be used, or if one does not exist, will be automatically created for the project. Automatic network allocation for a project only happens once for a project. Subsequent requests using *auto* for the same project will reuse the network that was previously allocated.

Also, the `uuid` field in the `networks` object in the server create request is now strictly enforced to be in UUID format.

In the context of device tagging at server create time, 2.37 also removes the tag attribute from networks. This is a bug that is fixed in 2.42, in which the tag attribute is reintroduced.

4.27.38 2.38 (Maximum in Newton)

Before version 2.38, the command `nova list --status invalid_status` was returning empty list for non admin user and 500 InternalServerError for admin user. As there are sufficient statuses defined already, any invalid status should not be accepted. From this version of the API admin as well as non admin user will get 400 HTTPBadRequest if invalid status is passed to nova list command.

4.27.39 2.39

Deprecates image-metadata proxy API that is just a proxy for Glance API to operate the image metadata. Also removes the extra quota enforcement with Nova *metadata* quota (quota checks for createImage and createBackup actions in Nova were removed). After this version Glance configuration option *image_property_quota* should be used to control the quota of image metadatas. Also, removes the *maxImageMeta* field from *os-limits* API response.

4.27.40 2.40

Optional query parameters `limit` and `marker` were added to the `os-simple-tenant-usage` endpoints for pagination. If a limit isnt provided, the configurable `max_limit` will be used which currently defaults to 1000.

```
GET /os-simple-tenant-usage?limit={limit}&marker={instance_uuid}
GET /os-simple-tenant-usage/{tenant_id}?limit={limit}&marker={instance_uuid}
```

A tenants usage statistics may span multiple pages when the number of instances exceeds limit, and API consumers will need to stitch together the aggregate results if they still want totals for all instances in a specific time window, grouped by tenant.

Older versions of the `os-simple-tenant-usage` endpoints will not accept these new paging query parameters, but they will start to silently limit by `max_limit` to encourage the adoption of this new microversion, and circumvent the existing possibility of DoS-like usage requests when there are thousands of instances.

4.27.41 2.41

The uuid attribute of an aggregate is now returned from calls to the */os-aggregates* endpoint. This attribute is autogenerated upon creation of an aggregate. The *os-aggregates* API resource endpoint remains an administrator-only API.

4.27.42 2.42 (Maximum in Ocata)

In the context of device tagging at server create time, a bug has caused the tag attribute to no longer be accepted for networks starting with version 2.37 and for block_device_mapping_v2 starting with version 2.33. Microversion 2.42 restores the tag parameter to both networks and block_device_mapping_v2, allowing networks and block devices to be tagged again.

4.27.43 2.43

The `os-hosts` API is deprecated as of the 2.43 microversion. Requests made with microversion >= 2.43 will result in a 404 error. To list and show host details, use the `os-hypervisors` API. To enable or disable a service, use the `os-services` API. There is no replacement for the *shutdown*, *startup*, *reboot*, or *maintenance_mode* actions as those are system-level operations which should be outside of the control of the compute service.

4.27.44 2.44

The following APIs which are considered as proxies of Neutron networking API, are deprecated and will result in a 404 error response in new Microversion:

```
POST /servers/{server_uuid}/action
{
    "addFixedIp": {...}
}

POST /servers/{server_uuid}/action
{
    "removeFixedIp": {...}
}

POST /servers/{server_uuid}/action
{
    "addFloatingIp": {...}
}

POST /servers/{server_uuid}/action
{
    "removeFloatingIp": {...}
}
```

Those server actions can be replaced by calling the Neutron API directly.

The nova-network specific API to query the servers interfaces is deprecated:

```
GET /servers/{server_uuid}/os-virtual-interfaces
```

To query attached neutron interfaces for a specific server, the API *GET /servers/{server_uuid}/os-interface* can be used.

4.27.45 2.45

The `createImage` and `createBackup` server action APIs no longer return a `Location` header in the response for the snapshot image, they now return a json dict in the response body with an `image_id` key and uuid value.

4.27.46 2.46

The request_id created for every inbound request is now returned in `X-OpenStack-Request-ID` in addition to `X-Compute-Request-ID` to be consistent with the rest of OpenStack. This is a signaling only microversion, as these header settings happen well before microversion processing.

4.27.47 2.47

Replace the `flavor` name/ref with the actual flavor details from the embedded flavor object when displaying server details. Requests made with microversion >= 2.47 will no longer return the flavor ID/link but instead will return a subset of the flavor details. If the user is prevented by policy from indexing extra-specs, then the `extra_specs` field will not be included in the flavor information.

4.27.48 2.48

Before version 2.48, VM diagnostics response was just a blob of data returned by each hypervisor. From this version VM diagnostics response is standardized. It has a set of fields which each hypervisor will try to fill. If a hypervisor driver is unable to provide a specific field then this field will be reported as None.

4.27.49 2.49

Continuing from device role tagging at server create time introduced in version 2.32 and later fixed in 2.42, microversion 2.49 allows the attachment of network interfaces and volumes with an optional `tag` parameter. This tag is used to identify the virtual devices in the guest and is exposed in the metadata API. Because the config drive cannot be updated while the guest is running, it will only contain metadata of devices that were tagged at boot time. Any changes made to devices while the instance is running - be it detaching a tagged device or performing a tagged device attachment - will not be reflected in the config drive.

Tagged volume attachment is not supported for shelved-offloaded instances.

4.27.50 2.50

The `server_groups` and `server_group_members` keys are exposed in GET & PUT `os-quota-class-sets` APIs Response body. Networks related quotas have been filtered out from os-quota-class. Below quotas are filtered out and not available in `os-quota-class-sets` APIs from this microversion onwards.

- fixed_ips
- floating_ips
- networks,
- security_group_rules
- security_groups

4.27.51 2.51

There are two changes for the 2.51 microversion:

- Add `volume-extended` event name to the `os-server-external-events` API. This will be used by the Block Storage service when extending the size of an attached volume. This signals the Compute service to perform any necessary actions on the compute host or hypervisor to adjust for the new volume block device size.
- Expose the `events` field in the response body for the `GET /servers/{server_id}/os-instance-actions/{request_id}` API. This is useful for API users to monitor when a volume extend operation completes for the given server instance. By default only users with the administrator role will be able to see event `traceback` details.

4.27.52 2.52

Adds support for applying tags when creating a server. The tag schema is the same as in the *2.26* microversion.

4.27.53 2.53 (Maximum in Pike)

os-services

Services are now identified by uuid instead of database id to ensure uniqueness across cells. This microversion brings the following changes:

- `GET /os-services` returns a uuid in the `id` field of the response
- `DELETE /os-services/{service_uuid}` requires a service uuid in the path
- The following APIs have been superseded by `PUT /os-services/{service_uuid}/`:
  - `PUT /os-services/disable`
  - `PUT /os-services/disable-log-reason`
  - `PUT /os-services/enable`
  - `PUT /os-services/force-down`

  `PUT /os-services/{service_uuid}` takes the following fields in the body:
  - `status` - can be either enabled or disabled to enable or disable the given service
  - `disabled_reason` - specify with status=disabled to log a reason for why the service is disabled
  - `forced_down` - boolean indicating if the service was forced down by an external service
- `PUT /os-services/{service_uuid}` will now return a full service resource representation like in a `GET` response

os-hypervisors

Hypervisors are now identified by uuid instead of database id to ensure uniqueness across cells. This microversion brings the following changes:

- `GET /os-hypervisors/{hypervisor_hostname_pattern}/search` is deprecated and replaced with the `hypervisor_hostname_pattern` query parameter on the `GET /os-hypervisors` and `GET /os-hypervisors/detail` APIs. Paging with `hypervisor_hostname_pattern` is not supported.
- `GET /os-hypervisors/{hypervisor_hostname_pattern}/servers` is deprecated and replaced with the `with_servers` query parameter on the `GET /os-hypervisors` and `GET /os-hypervisors/detail` APIs.
- `GET /os-hypervisors/{hypervisor_id}` supports the `with_servers` query parameter to include hosted server details in the response.
- `GET /os-hypervisors/{hypervisor_id}` and `GET /os-hypervisors/{hypervisor_id}/uptime` APIs now take a uuid value for the `{hypervisor_id}` path parameter.
- The `GET /os-hypervisors` and `GET /os-hypervisors/detail` APIs will now use a uuid marker for paging across cells.
- The following APIs will now return a uuid value for the hypervisor id and optionally service id fields in the response:
  - `GET /os-hypervisors`
  - `GET /os-hypervisors/detail`

  - `GET /os-hypervisors/{hypervisor_id}`
  - `GET /os-hypervisors/{hypervisor_id}/uptime`

4.27.54 2.54

Allow the user to set the server key pair while rebuilding.

4.27.55 2.55

Adds a `description` field to the flavor resource in the following APIs:

- `GET /flavors`
- `GET /flavors/detail`
- `GET /flavors/{flavor_id}`
- `POST /flavors`
- `PUT /flavors/{flavor_id}`

The embedded flavor description will not be included in server representations.

4.27.56 2.56

Updates the POST request body for the `migrate` action to include the the optional `host` string field defaulted to `null`. If `host` is set the migrate action verifies the provided host with the nova scheduler and uses it as the destination for the migration.

4.27.57 2.57

The 2.57 microversion makes the following changes:

- The `personality` parameter is removed from the server create and rebuild APIs.
- The `user_data` parameter is added to the server rebuild API.
- The `maxPersonality` and `maxPersonalitySize` limits are excluded from the `GET /limits` API response.
- The `injected_files`, `injected_file_content_bytes` and `injected_file_path_bytes` quotas are removed from the `os-quota-sets` and `os-quota-class-sets` APIs.

4.27.58 2.58

Add pagination support and `changes-since` filter for os-instance-actions API. Users can now use `limit` and `marker` to perform paginated query when listing instance actions. Users can also use `changes-since` filter to filter the results based on the last time the instance action was updated.

4.27.59 2.59

Added pagination support for migrations, there are four changes:

- Add pagination support and `changes-since` filter for os-migrations API. Users can now use `limit` and `marker` to perform paginate query when listing migrations.
- Users can also use `changes-since` filter to filter the results based on the last time the migration record was updated.
- `GET /os-migrations`, `GET /servers/{server_id}/migrations/{migration_id}` and `GET /servers/{server_id}/migrations` will now return a uuid value in addition to the migrations id in the response.
- The query parameter schema of the `GET /os-migrations` API no longer allows additional properties.

4.27.60 2.60 (Maximum in Queens)

From this version of the API users can attach a `multiattach` capable volume to multiple instances. The API request for creating the additional attachments is the same. The chosen virt driver and the volume back end has to support the functionality as well.

4.27.61 2.61

Exposes flavor extra_specs in the flavor representation. Now users can see the flavor extra-specs in flavor APIs response and do not need to call `GET /flavors/{flavor_id}/os-extra_specs` API. If the user is prevented by policy from indexing extra-specs, then the `extra_specs` field will not be included in the flavor information. Flavor extra_specs will be included in Response body of the following APIs:

- `GET /flavors/detail`
- `GET /flavors/{flavor_id}`
- `POST /flavors`
- `PUT /flavors/{flavor_id}`

4.27.62 2.62

Adds `host` (hostname) and `hostId` (an obfuscated hashed host id string) fields to the instance action `GET /servers/{server_id}/os-instance-actions/{req_id}` API. The display of the newly added `host` field will be controlled via policy rule `os_compute_api:os-instance-actions:events`, which is the same policy used for the `events.traceback` field. If the user is prevented by policy, only `hostId` will be displayed.

4.27.63 2.63

Adds support for the `trusted_image_certificates` parameter, which is used to define a list of trusted certificate IDs that can be used during image signature verification and certificate validation. The list is restricted to a maximum of 50 IDs. Note that `trusted_image_certificates` is not supported with volume-backed servers.

The `trusted_image_certificates` request parameter can be passed to the server create and rebuild APIs:

- `POST /servers`
- `POST /servers/{server_id}/action (rebuild)`

The `trusted_image_certificates` parameter will be in the response body of the following APIs:

- `GET /servers/detail`
- `GET /servers/{server_id}`
- `PUT /servers/{server_id}`
- `POST /servers/{server_id}/action (rebuild)`

4.27.64 2.64

Enable users to define the policy rules on server group policy to meet more advanced policy requirement. This microversion brings the following changes in server group APIs:

- Add `policy` and `rules` fields in the request of POST `/os-server-groups`. The `policy` represents the name of policy. The `rules` field, which is a dict, can be applied to the policy, which currently only support `max_server_per_host` for `anti-affinity` policy.
- The `policy` and `rules` fields will be returned in response body of POST, GET `/os-server-groups` API and GET `/os-server-groups/{server_group_id}` API.
- The `policies` and `metadata` fields have been removed from the response body of POST, GET `/os-server-groups` API and GET `/os-server-groups/{server_group_id}` API.

4.27.65 2.65 (Maximum in Rocky)

Add support for abort live migrations in `queued` and `preparing` status for API `DELETE /servers/{server_id}/migrations/{migration_id}`.

4.27.66 2.66

The `changes-before` filter can be included as a request parameter of the following APIs to filter by changes before or equal to the resource `updated_at` time:

- `GET /servers`
- `GET /servers/detail`
- `GET /servers/{server_id}/os-instance-actions`
- `GET /os-migrations`

4.27.67 2.67

Adds the `volume_type` parameter to `block_device_mapping_v2`, which can be used to specify cinder `volume_type` when creating a server.

4.27.68 2.68

Remove support for forced live migration and evacuate server actions.

4.27.69 2.69

Add support for returning minimal constructs for `GET /servers`, `GET /servers/detail`, `GET /servers/{server_id}` and `GET /os-services` when there is a transient unavailability condition in the deployment like an infrastructure failure. Starting from this microversion, the responses from the down part of the infrastructure for the above four requests will have missing key values to make it more resilient. The response body will only have a minimal set of information obtained from the available information in the API database for the down cells. See handling down cells for more information.

4.27.70 2.70

Exposes virtual device tags for volume attachments and virtual interfaces (ports). A `tag` parameter is added to the response body for the following APIs:

Volumes

- GET /servers/{server_id}/os-volume_attachments (list)
- GET /servers/{server_id}/os-volume_attachments/{volume_id} (show)
- POST /servers/{server_id}/os-volume_attachments (attach)

Ports

- GET /servers/{server_id}/os-interface (list)
- GET /servers/{server_id}/os-interface/{port_id} (show)
- POST /servers/{server_id}/os-interface (attach)

4.27.71 2.71

The `server_groups` parameter will be in the response body of the following APIs to list the server groups to which the server belongs:

- `GET /servers/{server_id}`
- `PUT /servers/{server_id}`
- `POST /servers/{server_id}/action (rebuild)`

4.27.72 2.72 (Maximum in Stein)

API microversion 2.72 adds support for creating servers with neutron ports that has resource request, e.g. neutron ports with QoS minimum bandwidth rule. Deleting servers with such ports have already been handled properly as well as detaching these type of ports.

API limitations:

- Creating servers with Neutron networks having QoS minimum bandwidth rule is not supported.
- Attaching Neutron ports and networks having QoS minimum bandwidth rule is not supported.
- Moving (resizing, migrating, live-migrating, evacuating, unshelving after shelve offload) servers with ports having resource request is not yet supported.

4.27.73 2.73

API microversion 2.73 adds support for specifying a reason when locking the server and exposes this information via `GET /servers/detail`, `GET /servers/{server_id}`, `PUT servers/{server_id}` and `POST /servers/{server_id}/action` where the action is rebuild. It also supports `locked` as a filter/sort parameter for `GET /servers/detail` and `GET /servers`.

4.27.74 2.74

API microversion 2.74 adds support for specifying optional `host` and/or `hypervisor_hostname` parameters in the request body of `POST /servers`. These request a specific destination host/node to boot the requested server. These parameters are mutually exclusive with the special `availability_zone` format of `zone:host:node`. Unlike `zone:host:node`, the `host` and/or `hypervisor_hostname` parameters still allow scheduler filters to be run. If the requested host/node is unavailable or otherwise unsuitable, earlier failure will be raised. There will be also a new policy named `compute:servers:create:requested_destination`. By default, it can be specified by administrators only.

4.27.75 2.75

Multiple API cleanups are done in API microversion 2.75:

- 400 error response for an unknown parameter in the querystring or request body.
- Make the server representation consistent among GET, PUT and rebuild server API responses. `PUT /servers/{server_id}` and `POST /servers/{server_id}/action {rebuild}` API responses are modified to add all the missing fields which are returned by `GET /servers/{server_id}`.
- Change the default return value of the `swap` field from the empty string to 0 (integer) in flavor APIs.
- Always return the `servers` field in the response of the `GET /os-hypervisors`, `GET /os-hypervisors/detail` and `GET /os-hypervisors/{hypervisor_id}` APIs even when there are no servers on a hypervisor.

4.27.76 2.76

Adds `power-update` event name to `os-server-external-events` API. The changes to the power state of an instance caused by this event can be viewed through `GET /servers/{server_id}/os-instance-actions` and `GET /servers/{server_id}/os-instance-actions/{request_id}`.

4.27.77 2.77

API microversion 2.77 adds support for specifying availability zone when unshelving a shelved offloaded server.

4.27.78 2.78

Add server sub-resource `topology` to show server NUMA information.

- `GET /servers/{server_id}/topology`

The default behavior is configurable using two new policies:

- `compute:server:topology:index`

- `compute:server:topology:host:index`

4.27.79 2.79 (Maximum in Train)

API microversion 2.79 adds support for specifying the `delete_on_termination` field in the request body when attaching a volume to a server, to support configuring whether to delete the data volume when the server is destroyed. Also, `delete_on_termination` is added to the GET responses when showing attached volumes.

The affected APIs are as follows:

- `POST /servers/{server_id}/os-volume_attachments`
- `GET /servers/{server_id}/os-volume_attachments`
- `GET /servers/{server_id}/os-volume_attachments/{volume_id}`

4.28 Conductor as a place for orchestrating tasks

In addition to its roles as a database proxy and object backporter the conductor service also serves as a centralized place to manage the execution of workflows which involve the scheduler. Rebuild, resize/migrate, and building an instance are managed here. This was done in order to have a better separation of responsibilities between what compute nodes should handle and what the scheduler should handle, and to clean up the path of execution. Conductor was chosen because in order to query the scheduler in a synchronous manner it needed to happen after the API had returned a response otherwise API response times would increase. And changing the scheduler call from asynchronous to synchronous helped to clean up the code.

To illustrate this the old process for building an instance was:

- API receives request to build an instance.
- API sends an RPC cast to the scheduler to pick a compute.
- Scheduler sends an RPC cast to the compute to build the instance, which means the scheduler needs to be able to communicate with all computes.
  - If the build succeeds it stops here.
  - If the build fails then the compute decides if the max number of scheduler retries has been hit. If so the build stops there.
    - If the build should be rescheduled the compute sends an RPC cast to the scheduler in order to pick another compute.

This was overly complicated and meant that the logic for scheduling/rescheduling was distributed throughout the code. The answer to this was to change to process to be the following:

- API receives request to build an instance.
- API sends an RPC cast to the conductor to build an instance. (or runs locally if conductor is configured to use local_mode)
- Conductor sends an RPC call to the scheduler to pick a compute and waits for the response. If there is a scheduler fail it stops the build at the conductor.
- Conductor sends an RPC cast to the compute to build the instance.
  - If the build succeeds it stops here.
  - If the build fails then compute sends an RPC cast to conductor to build an instance. This is the same RPC message that was sent by the API.

This new process means the scheduler only deals with scheduling, the compute only deals with building an instance, and the conductor manages the workflow. The code is now cleaner in the scheduler and computes.

4.29 Guru Meditation Reports

Nova contains a mechanism whereby developers and system administrators can generate a report about the state of a running Nova executable. This report is called a *Guru Meditation Report* (*GMR* for short).

4.29.1 Generating a GMR

A *GMR* can be generated by sending the *USR2* signal to any Nova process with support (see below). The *GMR* will then be outputted standard error for that particular process.

For example, suppose that `nova-api` has process id `8675`, and was run with `2>/var/log/nova/nova-api-err.log`. Then, `kill -USR2 8675` will trigger the Guru Meditation report to be printed to `/var/log/nova/nova-api-err.log`.

4.29.2 Structure of a GMR

The *GMR* is designed to be extensible; any particular executable may add its own sections. However, the base *GMR* consists of several sections:

Package Shows information about the package to which this process belongs, including version information

Threads Shows stack traces and thread ids for each of the threads within this process

Green Threads Shows stack traces for each of the green threads within this process (green threads dont have thread ids)

Configuration Lists all the configuration options currently accessible via the CONF object for the current process

4.29.3 Adding Support for GMRs to New Executables

Adding support for a *GMR* to a given executable is fairly easy.

First import the module, as well as the Nova version module:

```
from oslo_reports import guru_meditation_report as gmr
from nova import version
```

Then, register any additional sections (optional):

```
TextGuruMeditation.register_section('Some Special Section',
                                    some_section_generator)
```

Finally (under main), before running the main loop of the executable (usually `service.server(server)` or something similar), register the *GMR* hook:

```
TextGuruMeditation.setup_autorun(version)
```

4.29.4 Extending the GMR

As mentioned above, additional sections can be added to the GMR for a particular executable. For more information, see the inline documentation under `oslo.reports`

4.30 Internationalization

Nova uses the oslo.i18n library to support internationalization. The oslo.i18n library is built on top of gettext and provides functions that are used to enable user-facing strings such as log messages to appear in the appropriate language in different locales.

Nova exposes the oslo.i18n library support via the `nova/i18n.py` integration module. This module provides the functions needed to wrap translatable strings. It provides the `_()` wrapper for general user-facing messages (such as ones that end up in command line responses, or responses over the network).

One upon a time there was an effort to translate log messages in OpenStack projects. But starting with the Ocata release these are no longer being supported. Log messages **should not** be translated. Any use of `_LI()`, `_LW()`, `_LE()`, `_LC()` are vestigial and will be removed over time. No new uses of these should be added.

You should use the basic wrapper `_()` for strings which are not log messages that are expected to get to an end user:

```
raise nova.SomeException(_('Invalid service catalogue'))
```

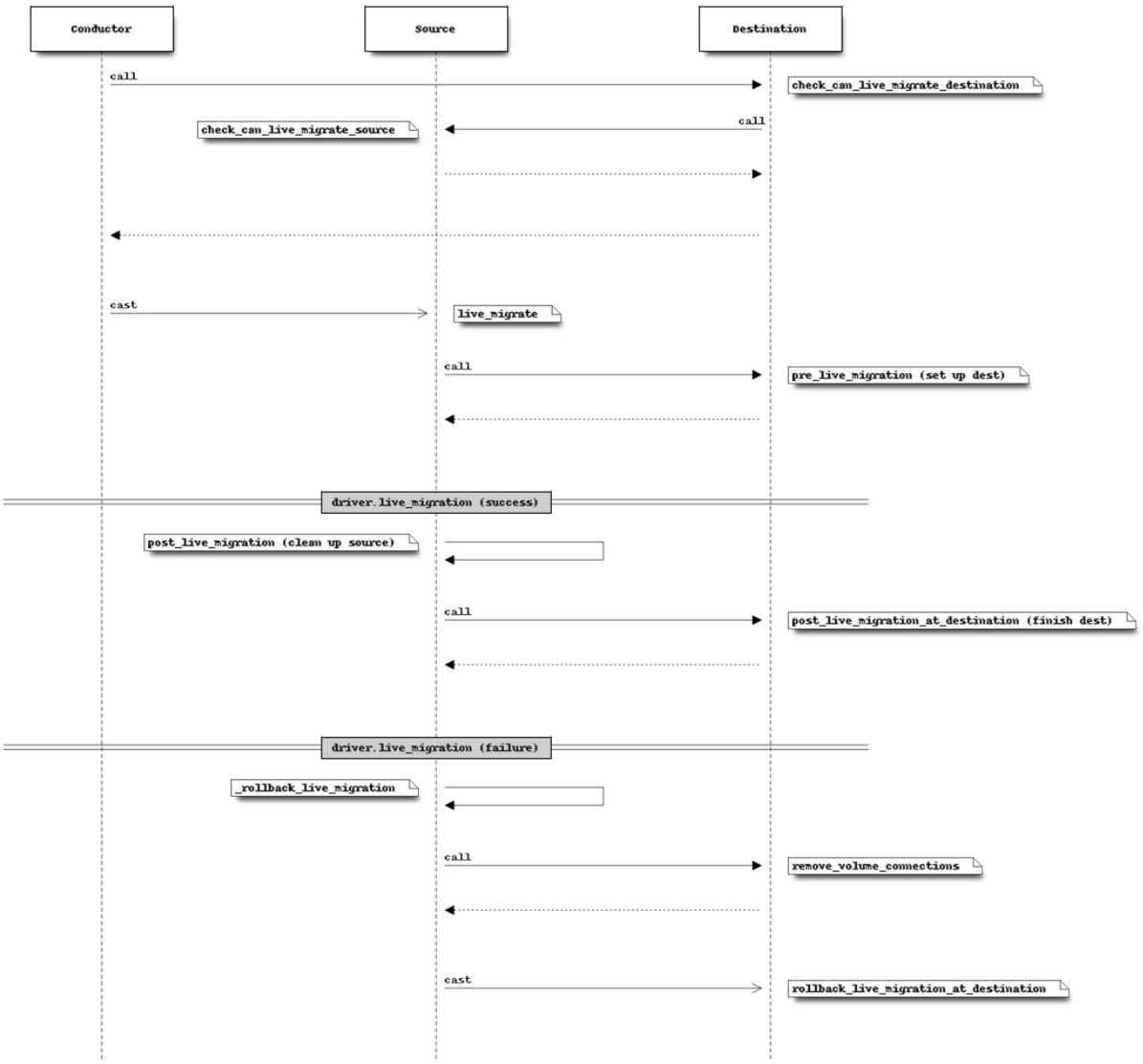
Do not use `locals()` for formatting messages because: 1. It is not as clear as using explicit dicts. 2. It could produce hidden errors during refactoring. 3. Changing the name of a variable causes a change in the message. 4. It creates a lot of otherwise unused variables.

If you do not follow the project conventions, your code may cause hacking checks to fail.

The `_()` function can be imported with

```
from nova.i18n import _
```

4.31 Live Migration



4.32 Notifications in Nova

Similarly to other OpenStack services Nova emits notifications to the message bus with the Notifier class provided by oslo.messaging. From the notification consumer point of view a notification consists of two parts: an envelope with a fixed structure defined by oslo.messaging and a payload defined by the service emitting the notification. The envelope format is the following:

```
{
    "priority": <string, selected from a predefined list by the sender>,
    "event_type": <string, defined by the sender>,
    "timestamp": <string, the isotime of when the notification emitted>,
    "publisher_id": <string, defined by the sender>,
    "message_id": <uuid, generated by oslo>,
```

(continues on next page)

(continued from previous page)

```
    "payload": <json serialized dict, defined by the sender>
}
```

Notifications can be completely disabled by setting the following in your nova configuration file:

```
[oslo_messaging_notifications]
driver = noop
```

There are two types of notifications in Nova: legacy notifications which have an unversioned payload and newer notifications which have a versioned payload.

4.32.1 Unversioned notifications

Nova code uses the nova.rpc.get_notifier call to get a configured oslo.messaging Notifier object and it uses the oslo provided functions on the Notifier object to emit notifications. The configuration of the returned Notifier object depends on the parameters of the get_notifier call and the value of the oslo.messaging configuration options `driver` and `topics`. There are notification configuration options in Nova which are specific for certain notification types like *`notifications.notify_on_state_change`*, *`notifications.default_level`*, etc.

The structure of the payload of the unversioned notifications is defined in the code that emits the notification and no documentation or enforced backward compatibility contract exists for that format.

4.32.2 Versioned notifications

The versioned notification concept is created to fix the shortcomings of the unversioned notifications. The envelope structure of the emitted notification is the same as in the unversioned notification case as it is provided by oslo.messaging. However the payload is not a free form dictionary but a serialized oslo versionedobjects object.

For example the wire format of the `service.update` notification looks like the following:

```
{
    "priority":"INFO",
    "payload":{
        "nova_object.namespace":"nova",
        "nova_object.name":"ServiceStatusPayload",
        "nova_object.version":"1.0",
        "nova_object.data":{
            "host":"host1",
            "disabled":false,
            "last_seen_up":null,
            "binary":"nova-compute",
            "topic":"compute",
            "disabled_reason":null,
            "report_count":1,
            "forced_down":false,
            "version":2
        }
    },
    "event_type":"service.update",
    "publisher_id":"nova-compute:host1"
}
```

The serialized oslo versionedobject as a payload provides a version number to the consumer so the consumer can detect if the structure of the payload is changed. Nova provides the following contract regarding the versioned notification payload:

- the payload version defined by the `nova_object.version` field of the payload will be increased if and only if the syntax or the semantics of the `nova_object.data` field of the payload is changed.
- a minor version bump indicates a backward compatible change which means that only new fields are added to the payload so a well written consumer can still consume the new payload without any change.
- a major version bump indicates a backward incompatible change of the payload which can mean removed fields, type change, etc in the payload.
- there is an additional field nova_object.name for every payload besides nova_object.data and nova_object.version. This field contains the name of the nova internal representation of the payload type. Client code should not depend on this name.

There is a Nova configuration parameter *notifications.notification_format* that can be used to specify which notifications are emitted by Nova.

The versioned notifications are emitted to a different topic than the legacy notifications. By default they are emitted to versioned_notifications but it is configurable in the nova.conf with the *notifications.versioned_notifications_topics* config option.

A presentation from the Train summit goes over the background and usage of versioned notifications, and provides a demo.

How to add a new versioned notification

To support the above contract from the Nova code every versioned notification is modeled with oslo versionedobjects. Every versioned notification class shall inherit from the `nova.notifications.objects.base.NotificationBase` which already defines three mandatory fields of the notification `event_type`, `publisher` and `priority`. The new notification class shall add a new field `payload` with an appropriate payload type. The payload object of the notifications shall inherit from the `nova.notifications.objects.base.NotificationPayloadBase` class and shall define the fields of the payload as versionedobject fields. The base classes are described in the following section.

The nova.notifications.objects.base module

class nova.notifications.objects.base.**EventType**(*object*, *action*, *phase=None*)
Bases: nova.notifications.objects.base.NotificationObject

to_notification_event_type_field()
Serialize the object to the wire format.

class nova.notifications.objects.base.**NotificationBase**(***kwargs*)
Bases: nova.notifications.objects.base.NotificationObject

Base class for versioned notifications.

Every subclass shall define a payload field.

emit(*context*)
Send the notification.

class nova.notifications.objects.base.**NotificationObject**(***kwargs*)
Bases: nova.objects.base.NovaObject

Base class for every notification related versioned object.

class nova.notifications.objects.base.**NotificationPayloadBase**
Bases: nova.notifications.objects.base.NotificationObject

Base class for the payload of versioned notifications.

populate_schema(*set_none=True*, ***kwargs*)
Populate the object based on the SCHEMA and the source objects

Parameters **kwargs** – A dict contains the source object at the key defined in the SCHEMA

class nova.notifications.objects.base.**NotificationPublisher**(*host*, *source*)
Bases: nova.notifications.objects.base.NotificationObject

nova.notifications.objects.base.**notification_sample**(*sample*)
Class decorator to attach the notification sample information to the notification object for documentation generation purposes.

Parameters **sample** – the path of the sample json file relative to the doc/notification_samples/ directory in the nova repository root.

Please note that the notification objects shall not be registered to the NovaObjectRegistry to avoid mixing nova internal objects with the notification objects. Instead of that use the register_notification decorator on every concrete notification object.

The following code example defines the necessary model classes for a new notification `myobject.update`:

```
@notification.notification_sample('myobject-update.json')
@object_base.NovaObjectRegistry.register.register_notification
class MyObjectNotification(notification.NotificationBase):
    # Version 1.0: Initial version
    VERSION = '1.0'

    fields = {
        'payload': fields.ObjectField('MyObjectUpdatePayload')
    }


@object_base.NovaObjectRegistry.register.register_notification
class MyObjectUpdatePayload(notification.NotificationPayloadBase):
    # Version 1.0: Initial version
    VERSION = '1.0'
    fields = {
        'some_data': fields.StringField(),
        'another_data': fields.StringField(),
    }
```

After that the notification can be populated and emitted with the following code:

```
payload = MyObjectUpdatePayload(some_data="foo", another_data="bar")
MyObjectNotification(
    publisher=notification.NotificationPublisher.from_service_obj(
        <nova.objects.service.Service instance that emits the notification>),
    event_type=notification.EventType(
        object='myobject',
        action=fields.NotificationAction.UPDATE),
    priority=fields.NotificationPriority.INFO,
    payload=payload).emit(context)
```

The above code will generate the following notification on the wire:

```
{
    "priority":"INFO",
    "payload":{
        "nova_object.namespace":"nova",
```

(continues on next page)

(continued from previous page)

```
        "nova_object.name":"MyObjectUpdatePayload",
        "nova_object.version":"1.0",
        "nova_object.data":{
            "some_data":"foo",
            "another_data":"bar",
        }
    },
    "event_type":"myobject.update",
    "publisher_id":"<the name of the service>:<the host where the service runs>"
}
```

There is a possibility to reuse an existing versionedobject as notification payload by adding a `SCHEMA` field for the payload class that defines a mapping between the fields of existing objects and the fields of the new payload object. For example the service.status notification reuses the existing `nova.objects.service.Service` object when defines the notifications payload:

```
@notification.notification_sample('service-update.json')
@object_base.NovaObjectRegistry.register.register_notification
class ServiceStatusNotification(notification.NotificationBase):
    # Version 1.0: Initial version
    VERSION = '1.0'

    fields = {
        'payload': fields.ObjectField('ServiceStatusPayload')
    }

@object_base.NovaObjectRegistry.register.register_notification
class ServiceStatusPayload(notification.NotificationPayloadBase):
    SCHEMA = {
        'host': ('service', 'host'),
        'binary': ('service', 'binary'),
        'topic': ('service', 'topic'),
        'report_count': ('service', 'report_count'),
        'disabled': ('service', 'disabled'),
        'disabled_reason': ('service', 'disabled_reason'),
        'availability_zone': ('service', 'availability_zone'),
        'last_seen_up': ('service', 'last_seen_up'),
        'forced_down': ('service', 'forced_down'),
        'version': ('service', 'version')
    }
    # Version 1.0: Initial version
    VERSION = '1.0'
    fields = {
        'host': fields.StringField(nullable=True),
        'binary': fields.StringField(nullable=True),
        'topic': fields.StringField(nullable=True),
        'report_count': fields.IntegerField(),
        'disabled': fields.BooleanField(),
        'disabled_reason': fields.StringField(nullable=True),
        'availability_zone': fields.StringField(nullable=True),
        'last_seen_up': fields.DateTimeField(nullable=True),
        'forced_down': fields.BooleanField(),
        'version': fields.IntegerField(),
    }

    def populate_schema(self, service):
        super(ServiceStatusPayload, self).populate_schema(service=service)
```

If the SCHEMA field is defined then the payload object needs to be populated with the populate_schema call before it can be emitted:

```
payload = ServiceStatusPayload()
payload.populate_schema(service=<nova.object.service.Service object>)
ServiceStatusNotification(
    publisher=notification.NotificationPublisher.from_service_obj(
        <nova.object.service.Service object>),
    event_type=notification.EventType(
        object='service',
        action=fields.NotificationAction.UPDATE),
    priority=fields.NotificationPriority.INFO,
    payload=payload).emit(context)
```

The above code will emit the *already shown notification* on the wire.

Every item in the SCHEMA has the syntax of:

```
<payload field name which needs to be filled>:
    (<name of the parameter of the populate_schema call>,
     <the name of a field of the parameter object>)
```

The mapping defined in the SCHEMA field has the following semantics. When the populate_schema function is called the content of the SCHEMA field is enumerated and the value of the field of the pointed parameter object is copied to the requested payload field. So in the above example the host field of the payload object is populated from the value of the host field of the service object that is passed as a parameter to the populate_schema call.

A notification payload object can reuse fields from multiple existing objects. Also a notification can have both new and reused fields in its payload.

Note that the notifications publisher instance can be created two different ways. It can be created by instantiating the NotificationPublisher object with a host and a source string parameter or it can be generated from a Service object by calling NotificationPublisher.from_service_obj function.

Versioned notifications shall have a sample file stored under doc/sample_notifications directory and the notification object shall be decorated with the notification_sample decorator. For example the service.update notification has a sample file stored in doc/sample_notifications/service-update.json and the ServiceUpdateNotification class is decorated accordingly.

Notification payload classes can use inheritance to avoid duplicating common payload fragments in nova code. However the leaf classes used directly in a notification should be created with care to avoid future needs of adding extra level of inheritance that changes the name of the leaf class as that name is present in the payload class. If this cannot be avoided and the only change is the renaming then the version of the new payload shall be the same as the old payload was before the rename. See[1] as an example. If the renaming involves any other changes on the payload (e.g. adding new fields) then the version of the new payload shall be higher than the old payload was. See[2] as an example.

What should be in the notification payload

This is just a guideline. You should always consider the actual use case that requires the notification.

- Always include the identifier (e.g. uuid) of the entity that can be used to query the whole entity over the REST API so that the consumer can get more information about the entity.
- You should consider including those fields that are related to the event you are sending the notification about. For example if a change of a field of the entity triggers an update notification then you should include the field to the payload.

[1] https://review.opendev.org/#/c/463001/
[2] https://review.opendev.org/#/c/453077/

- An update notification should contain information about what part of the entity is changed. Either by filling the nova_object.changes part of the payload (note that it is not supported by the notification framework currently) or sending both the old state and the new state of the entity in the payload.
- You should never include a nova internal object in the payload. Create a new object and use the SCHEMA field to map the internal object to the notification payload. This way the evolution of the internal object model can be decoupled from the evolution of the notification payload.

  Important: This does not mean that every field from internal objects should be mirrored in the notification payload objects. Think about what is actually needed by a consumer before adding it to a payload. When in doubt, if no one is requesting specific information in notifications, then leave it out until someone asks for it.

- The delete notification should contain the same information as the create or update notifications. This makes it possible for the consumer to listen only to the delete notifications but still filter on some fields of the entity (e.g. project_id).

What should NOT be in the notification payload

- Generally anything that contains sensitive information about the internals of the nova deployment, for example fields that contain access credentials to a cell database or message queue (see bug 1823104).

Existing versioned notifications

Note: Versioned notifications are added in each release, so the samples represented below may not necessarily be in an older version of nova. Ensure you are looking at the correct version of the documentation for the release you are using.

Event type	Notification class	Payload
aggregate.add_host.end	AggregateNotification	Aggre
aggregate.add_host.start	AggregateNotification	Aggre
aggregate.create.end	AggregateNotification	Aggre
aggregate.create.start	AggregateNotification	Aggre
aggregate.delete.end	AggregateNotification	Aggre
aggregate.delete.start	AggregateNotification	Aggre
aggregate.remove_host.end	AggregateNotification	Aggre
aggregate.remove_host.start	AggregateNotification	Aggre
aggregate.update_metadata.end	AggregateNotification	Aggre
aggregate.update_metadata.start	AggregateNotification	Aggre
aggregate.update_prop.end	AggregateNotification	Aggre
aggregate.update_prop.start	AggregateNotification	Aggre
compute_task.build_instances.error	ComputeTaskNotification	Compu
compute_task.migrate_server.error	ComputeTaskNotification	Compu
compute_task.rebuild_server.error	ComputeTaskNotification	Compu
compute.exception	ExceptionNotification	Excep
flavor.create	FlavorNotification	Flavo
flavor.delete	FlavorNotification	Flavo
flavor.update	FlavorNotification	Flavo
instance.delete.end	InstanceActionNotification	Insta

Table 168 – continued from previous page

instance.delete.start	InstanceActionNotification	Insta
instance.evacuate	InstanceActionNotification	Insta
instance.interface_attach.end	InstanceActionNotification	Insta
instance.interface_attach.error	InstanceActionNotification	Insta
instance.interface_attach.start	InstanceActionNotification	Insta
instance.interface_detach.end	InstanceActionNotification	Insta
instance.interface_detach.start	InstanceActionNotification	Insta
instance.live_migration_abort.end	InstanceActionNotification	Insta
instance.live_migration_abort.start	InstanceActionNotification	Insta
instance.live_migration_force_complete.end	InstanceActionNotification	Insta
instance.live_migration_force_complete.start	InstanceActionNotification	Insta
instance.live_migration_post.end	InstanceActionNotification	Insta
instance.live_migration_post.start	InstanceActionNotification	Insta
instance.live_migration_post_dest.end	InstanceActionNotification	Insta
instance.live_migration_post_dest.start	InstanceActionNotification	Insta
instance.live_migration_pre.end	InstanceActionNotification	Insta
instance.live_migration_pre.start	InstanceActionNotification	Insta
instance.live_migration_rollback.end	InstanceActionNotification	Insta
instance.live_migration_rollback.start	InstanceActionNotification	Insta
instance.live_migration_rollback_dest.end	InstanceActionNotification	Insta
instance.live_migration_rollback_dest.start	InstanceActionNotification	Insta
instance.lock	InstanceActionNotification	Insta
instance.pause.end	InstanceActionNotification	Insta
instance.pause.start	InstanceActionNotification	Insta
instance.power_off.end	InstanceActionNotification	Insta
instance.power_off.start	InstanceActionNotification	Insta
instance.power_on.end	InstanceActionNotification	Insta
instance.power_on.start	InstanceActionNotification	Insta
instance.reboot.end	InstanceActionNotification	Insta
instance.reboot.error	InstanceActionNotification	Insta
instance.reboot.start	InstanceActionNotification	Insta
instance.resize.end	InstanceActionNotification	Insta
instance.resize.error	InstanceActionNotification	Insta
instance.resize.start	InstanceActionNotification	Insta
instance.resize_confirm.end	InstanceActionNotification	Insta
instance.resize_confirm.start	InstanceActionNotification	Insta
instance.resize_finish.end	InstanceActionNotification	Insta
instance.resize_finish.start	InstanceActionNotification	Insta
instance.resize_revert.end	InstanceActionNotification	Insta
instance.resize_revert.start	InstanceActionNotification	Insta
instance.restore.end	InstanceActionNotification	Insta
instance.restore.start	InstanceActionNotification	Insta
instance.resume.end	InstanceActionNotification	Insta
instance.resume.start	InstanceActionNotification	Insta
instance.shelve.end	InstanceActionNotification	Insta
instance.shelve.start	InstanceActionNotification	Insta
instance.shelve_offload.end	InstanceActionNotification	Insta
instance.shelve_offload.start	InstanceActionNotification	Insta
instance.shutdown.end	InstanceActionNotification	Insta
instance.shutdown.start	InstanceActionNotification	Insta

Table 168 – continued from previous page

instance.soft_delete.end	InstanceActionNotification	Insta
instance.soft_delete.start	InstanceActionNotification	Insta
instance.suspend.end	InstanceActionNotification	Insta
instance.suspend.start	InstanceActionNotification	Insta
instance.trigger_crash_dump.end	InstanceActionNotification	Insta
instance.trigger_crash_dump.start	InstanceActionNotification	Insta
instance.unlock	InstanceActionNotification	Insta
instance.unpause.end	InstanceActionNotification	Insta
instance.unpause.start	InstanceActionNotification	Insta
instance.unrescue.end	InstanceActionNotification	Insta
instance.unrescue.start	InstanceActionNotification	Insta
instance.unshelve.end	InstanceActionNotification	Insta
instance.unshelve.start	InstanceActionNotification	Insta
instance.rebuild.end	InstanceActionRebuildNotification	Insta
instance.rebuild.error	InstanceActionRebuildNotification	Insta
instance.rebuild.start	InstanceActionRebuildNotification	Insta
instance.rebuild_scheduled	InstanceActionRebuildNotification	Insta
instance.rescue.end	InstanceActionRescueNotification	Insta
instance.rescue.start	InstanceActionRescueNotification	Insta
instance.resize_prep.end	InstanceActionResizePrepNotification	Insta
instance.resize_prep.start	InstanceActionResizePrepNotification	Insta
instance.snapshot.end	InstanceActionSnapshotNotification	Insta
instance.snapshot.start	InstanceActionSnapshotNotification	Insta
instance.volume_attach.end	InstanceActionVolumeNotification	Insta
instance.volume_attach.error	InstanceActionVolumeNotification	Insta
instance.volume_attach.start	InstanceActionVolumeNotification	Insta
instance.volume_detach.end	InstanceActionVolumeNotification	Insta
instance.volume_detach.start	InstanceActionVolumeNotification	Insta
instance.volume_swap.end	InstanceActionVolumeSwapNotification	Insta
instance.volume_swap.error	InstanceActionVolumeSwapNotification	Insta
instance.volume_swap.start	InstanceActionVolumeSwapNotification	Insta
instance.create.end	InstanceCreateNotification	Insta
instance.create.error	InstanceCreateNotification	Insta
instance.create.start	InstanceCreateNotification	Insta
instance.exists	InstanceExistsNotification	Insta
instance.update	InstanceUpdateNotification	Insta
keypair.create.end	KeypairNotification	Keypa
keypair.create.start	KeypairNotification	Keypa
keypair.delete.end	KeypairNotification	Keypa
keypair.delete.start	KeypairNotification	Keypa
keypair.import.end	KeypairNotification	Keypa
keypair.import.start	KeypairNotification	Keypa
libvirt.connect.error	LibvirtErrorNotification	Libvi
metrics.update	MetricsNotification	Metri
scheduler.select_destinations.end	SelectDestinationsNotification	Reque
scheduler.select_destinations.start	SelectDestinationsNotification	Reque
server_group.add_member	ServerGroupNotification	Serve
server_group.create	ServerGroupNotification	Serve
server_group.delete	ServerGroupNotification	Serve
service.create	ServiceStatusNotification	Servi

Table 168 – continued from previous page

service.delete	ServiceStatusNotification	Servi
service.update	ServiceStatusNotification	Servi
volume.usage	VolumeUsageNotification	Volum

4.33 REST API Policy Enforcement

The following describes some of the shortcomings in how policy is used and enforced in nova, along with some benefits of fixing those issues. Each issue has a section dedicated to describing the underlying cause and historical context in greater detail.

4.33.1 Problems with current system

The following is a list of issues with the existing policy enforcement system:

- Default policies lack exhaustive testing
- Mismatch between authoritative scope and resources
- Policies are inconsistently named
- Current defaults do not use default roles provided from keystone
- Policy enforcement is spread across multiple levels and components
- Some policies use hard-coded check strings
- Some APIs do not use granular rules

Addressing the list above helps operators by:

1. Providing them with flexible and useful defaults
2. Reducing the likelihood of writing and maintaining custom policies
3. Improving interoperability between deployments
4. Increasing RBAC confidence through first-class testing and verification
5. Reducing complexity by using consistent policy naming conventions
6. Exposing more functionality to end-users, safely, making the entire nova API more self-serviceable resulting in less operational overhead for operators to do things on behalf of users

Additionally, the following is a list of benefits to contributors:

1. Reduce developer maintenance and cost by isolating policy enforcement into a single layer
2. Reduce complexity by using consistent policy naming conventions
3. Increased confidence in RBAC refactoring through exhaustive testing that prevents regressions before they merge

4.33.2 Future of policy enforcement

The generic rule for all the improvement is keep V2 API back-compatible. Because V2 API may be deprecated after V2.1 parity with V2. This can reduce the risk we take. The improvement just for EC2 and V2.1 API. There isnt any user for V2.1, as it isnt ready yet. We have to do change for EC2 API. EC2 API wont be removed like v2 API. If we keep back-compatible for EC2 API also, the old compute api layer checks wont be removed forever. EC2 API is

really small than Nova API. Its about 29 APIs without volume and image related(those policy check done by cinder and glance). So it will affect user less.

Enforcement policy at REST API layer

The policy should be only enforced at REST API layer. This is clear for user to know where the policy will be enforced. If the policy spread into multiple layer of nova code, user wont know when and where the policy will be enforced if they didnt have knowledge about nova code.

Remove all the permission checking under REST API layer. Policy will only be enforced at REST API layer.

This will affect the EC2 API and V2.1 API, there are some API just have policy enforcement at Compute/Network API layer, those policy will be move to API layer and renamed.

Removes hard-code permission checks

Hard-coded permission checks make it impossible to supply a configurable policy. They should be removed in order to make nova auth completely configurable.

This will affect EC2 API and Nova V2.1 API. User need update their policy rule to match the old hard-code permission.

For Nova V2 API, the hard-code permission checks will be moved to REST API layer to guarantee it wont break the back-compatibility. That may ugly some hard-code permission check in API layer, but V2 API will be removed once V2.1 API ready, so our choice will reduce the risk.

Use different prefix in policy rule name for EC2/V2/V2.1 API

Currently all the APIs(Nova v2/v2.1 API, EC2 API) use same set of policy rules. Especially there isnt obvious mapping between those policy rules and EC2 API. User can know clearly which policy should be configured for specific API.

Nova should provide different prefix for policy rule name that used to group them, and put them in different policy configure file in policy.d

- EC2 API: Use prefix ec2_api. The rule looks like ec2_api:[action]
- Nova V2 API: After we move to V2.1, we neednt spend time to change V2 api rule, and neednt to bother deployer upgrade their policy config. So just keep V2 API policy rule named as before.
- Nova V2.1 API: We name the policy rule as os_compute_api:[extension]:[action]. The core API may be changed in the future, so we neednt name them as compute or compute_extension to distinguish the core or extension API.

This will affect EC2 API and V2.1 API. For EC2 API, it need deployer update their policy config. For V2.1 API, there isnt any user yet, so there wont any effect.

4.33.3 Existed Nova API being restricted

Nova provide default policy rules for all the APIs. Operator should only make the policy rule more permissive. If the Operator make the API to be restricted that make break the existed API user or application. Thats kind of back-incompatible. SO Operator can free to add additional permission to the existed API.

4.33.4 Policy Enforcement by user_id

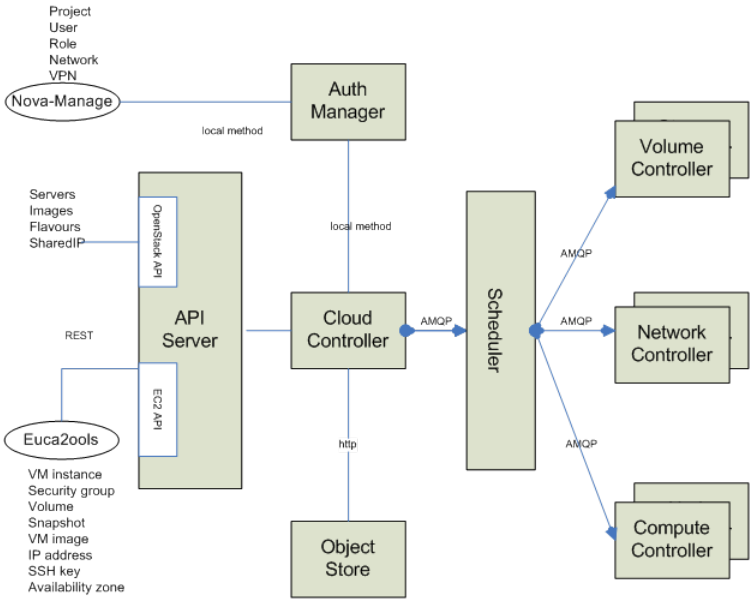
In the legacy v2 API, the policy enforces with target object, and some operators implement user-based authorization based on that. Actually only project-based authorization is well tested, the user based authorization is untested and isnt supported by Nova. In the future, the nova will remove all the supports for user-based authorization.

4.34 AMQP and Nova

AMQP is the messaging technology chosen by the OpenStack cloud. The AMQP broker, default to Rabbitmq, sits between any two Nova components and allows them to communicate in a loosely coupled fashion. More precisely, Nova components (the compute fabric of OpenStack) use Remote Procedure Calls (RPC hereinafter) to communicate to one another; however such a paradigm is built atop the publish/subscribe paradigm so that the following benefits can be achieved:

- Decoupling between client and servant (such as the client does not need to know where the servants reference is).
- Full a-synchronism between client and servant (such as the client does not need the servant to run at the same time of the remote call).
- Random balancing of remote calls (such as if more servants are up and running, one-way calls are transparently dispatched to the first available servant).

Nova uses direct, fanout, and topic-based exchanges. The architecture looks like the one depicted in the figure below:



Nova implements RPC (both request+response, and one-way, respectively nicknamed `rpc.call` and `rpc.cast`) over AMQP by providing an adapter class which take cares of marshaling and unmarshaling of messages into function calls. Each Nova service (for example Compute, Scheduler, etc.) create two queues at the initialization time, one which accepts messages with routing keys `NODE-TYPE.NODE-ID` (for example `compute.hostname`) and another, which accepts messages with routing keys as generic `NODE-TYPE` (for example `compute`). The former is used specifically when Nova-API needs to redirect commands to a specific node like `openstack server delete $instance`. In this case, only the compute node whose hosts hypervisor is running the virtual machine can kill the instance. The API acts as a consumer when RPC calls are request/response, otherwise it acts as a publisher only.

4.34.1 Nova RPC Mappings

The figure below shows the internals of a message broker node (referred to as a RabbitMQ node in the diagrams) when a single instance is deployed and shared in an OpenStack cloud. Every Nova component connects to the message broker and, depending on its personality (for example a compute node or a network node), may use the queue either as an Invoker (such as API or Scheduler) or a Worker (such as Compute or Network). Invokers and Workers do not actually exist in the Nova object model, but we are going to use them as an abstraction for sake of clarity. An Invoker is a component that sends messages in the queuing system via two operations: 1) `rpc.call` and ii) `rpc.cast`; a Worker is a component that receives messages from the queuing system and reply accordingly to `rpc.call` operations.

Figure 2 shows the following internal elements:

Topic Publisher A Topic Publisher comes to life when an `rpc.call` or an `rpc.cast` operation is executed; this object is instantiated and used to push a message to the queuing system. Every publisher connects always to the same topic-based exchange; its life-cycle is limited to the message delivery.

Direct Consumer A Direct Consumer comes to life if (and only if) an `rpc.call` operation is executed; this object is instantiated and used to receive a response message from the queuing system. Every consumer connects to a unique direct-based exchange via a unique exclusive queue; its life-cycle is limited to the message delivery; the exchange and queue identifiers are determined by a UUID generator, and are marshaled in the message sent by the Topic Publisher (only `rpc.call` operations).

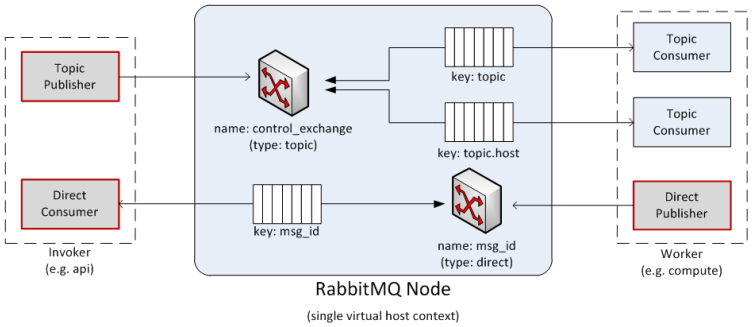
Topic Consumer A Topic Consumer comes to life as soon as a Worker is instantiated and exists throughout its life-cycle; this object is used to receive messages from the queue and it invokes the appropriate action as defined by the Worker role. A Topic Consumer connects to the same topic-based exchange either via a shared queue or via a unique exclusive queue. Every Worker has two topic consumers, one that is addressed only during `rpc.cast` operations (and it connects to a shared queue whose exchange key is `topic`) and the other that is addressed only during `rpc.call` operations (and it connects to a unique queue whose exchange key is `topic.host`).

Direct Publisher A Direct Publisher comes to life only during `rpc.call` operations and it is instantiated to return the message required by the request/response operation. The object connects to a direct-based exchange whose identity is dictated by the incoming message.

Topic Exchange The Exchange is a routing table that exists in the context of a virtual host (the multi-tenancy mechanism provided by RabbitMQ etc); its type (such as topic vs. direct) determines the routing policy; a message broker node will have only one topic-based exchange for every topic in Nova.

Direct Exchange This is a routing table that is created during `rpc.call` operations; there are many instances of this kind of exchange throughout the life-cycle of a message broker node, one for each `rpc.call` invoked.

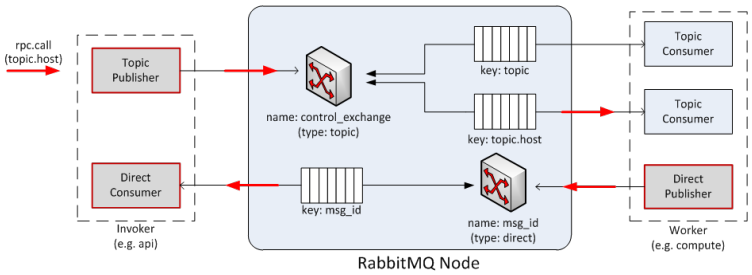
Queue Element A Queue is a message bucket. Messages are kept in the queue until a Consumer (either Topic or Direct Consumer) connects to the queue and fetch it. Queues can be shared or can be exclusive. Queues whose routing key is `topic` are shared amongst Workers of the same personality.



4.34.2 RPC Calls

The diagram below shows the message flow during an `rpc.call` operation:

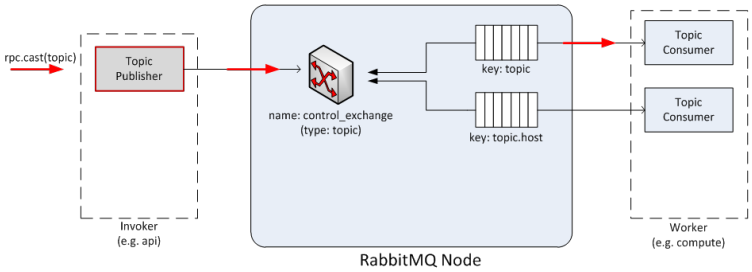
1. A Topic Publisher is instantiated to send the message request to the queuing system; immediately before the publishing operation, a Direct Consumer is instantiated to wait for the response message.
2. Once the message is dispatched by the exchange, it is fetched by the Topic Consumer dictated by the routing key (such as topic.host) and passed to the Worker in charge of the task.
3. Once the task is completed, a Direct Publisher is allocated to send the response message to the queuing system.
4. Once the message is dispatched by the exchange, it is fetched by the Direct Consumer dictated by the routing key (such as `msg_id`) and passed to the Invoker.



4.34.3 RPC Casts

The diagram below shows the message flow during an `rpc.cast` operation:

1. A Topic Publisher is instantiated to send the message request to the queuing system.
2. Once the message is dispatched by the exchange, it is fetched by the Topic Consumer dictated by the routing key (such as topic) and passed to the Worker in charge of the task.



4.34.4 AMQP Broker Load

At any given time the load of a message broker node running RabbitMQ etc is function of the following parameters:

Throughput of API calls The number of API calls (more precisely `rpc.call` ops) being served by the OpenStack cloud dictates the number of direct-based exchanges, related queues and direct consumers connected to them.

Number of Workers There is one queue shared amongst workers with the same personality; however there are as many exclusive queues as the number of workers; the number of workers dictates also the number of routing keys within the topic-based exchange, which is shared amongst all workers.

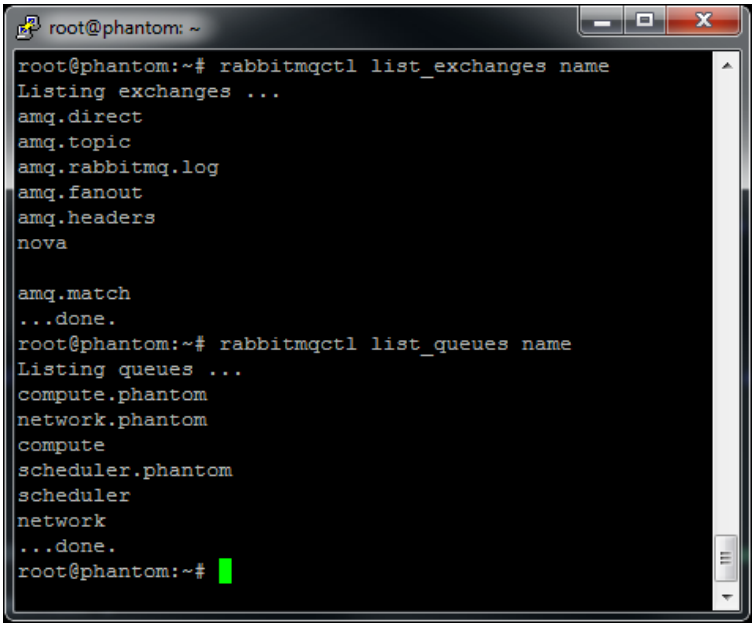
The figure below shows the status of a RabbitMQ node after Nova components bootstrap in a test environment. Exchanges and queues being created by Nova components are:

- Exchanges

1. nova (topic exchange)

- Queues

1. `compute.phantom` (`phantom` is hostname)
2. `compute`
3. `network.phantom` (`phantom` is hostname)
4. `network`
5. `scheduler.phantom` (`phantom` is hostname)
6. `scheduler`



4.34.5 RabbitMQ Gotchas

Nova uses Kombu to connect to the RabbitMQ environment. Kombu is a Python library that in turn uses AMQPLib, a library that implements the standard AMQP 0.8 at the time of writing. When using Kombu, Invokers and Workers need the following parameters in order to instantiate a Connection object that connects to the RabbitMQ server (please note that most of the following material can be also found in the Kombu documentation; it has been summarized and revised here for sake of clarity):

`hostname` The hostname to the AMQP server.

`userid` A valid username used to authenticate to the server.

`password` The password used to authenticate to the server.

`virtual_host` The name of the virtual host to work with. This virtual host must exist on the server, and the user must have access to it. Default is /.

`port` The port of the AMQP server. Default is `5672` (amqp).

The following parameters are default:

`insist` Insist on connecting to a server. In a configuration with multiple load-sharing servers, the Insist option tells the server that the client is insisting on a connection to the specified server. Default is False.

connect_timeout The timeout in seconds before the client gives up connecting to the server. The default is no timeout.

ssl Use SSL to connect to the server. The default is False.

More precisely Consumers need the following parameters:

connection The above mentioned Connection object.

queue Name of the queue.

exchange Name of the exchange the queue binds to.

routing_key The interpretation of the routing key depends on the value of the `exchange_type` attribute.

Direct exchange If the routing key property of the message and the `routing_key` attribute of the queue are identical, then the message is forwarded to the queue.

Fanout exchange Messages are forwarded to the queues bound the exchange, even if the binding does not have a key.

Topic exchange If the routing key property of the message matches the routing key of the key according to a primitive pattern matching scheme, then the message is forwarded to the queue. The message routing key then consists of words separated by dots (`.`, like domain names), and two special characters are available; star (`*`) and hash (`#`). The star matches any word, and the hash matches zero or more words. For example `.stock.#` matches the routing keys `usd.stock` and `eur.stock.db` but not `stock.nasdaq`.

durable This flag determines the durability of both exchanges and queues; durable exchanges and queues remain active when a RabbitMQ server restarts. Non-durable exchanges/queues (transient exchanges/queues) are purged when a server restarts. It is worth noting that AMQP specifies that durable queues cannot bind to transient exchanges. Default is True.

auto_delete If set, the exchange is deleted when all queues have finished using it. Default is False.

exclusive Exclusive queues (such as non-shared) may only be consumed from by the current connection. When exclusive is on, this also implies `auto_delete`. Default is False.

exchange_type AMQP defines several default exchange types (routing algorithms) that covers most of the common messaging use cases.

auto_ack Acknowledgment is handled automatically once messages are received. By default `auto_ack` is set to False, and the receiver is required to manually handle acknowledgment.

no_ack It disable acknowledgment on the server-side. This is different from `auto_ack` in that acknowledgment is turned off altogether. This functionality increases performance but at the cost of reliability. Messages can get lost if a client dies before it can deliver them to the application.

auto_declare If this is True and the exchange name is set, the exchange will be automatically declared at instantiation. Auto declare is on by default.

Publishers specify most the parameters of Consumers (such as they do not specify a queue name), but they can also specify the following:

delivery_mode The default delivery mode used for messages. The value is an integer. The following delivery modes are supported by RabbitMQ:

1 (transient) The message is transient. Which means it is stored in memory only, and is lost if the server dies or restarts.

2 (persistent) The message is persistent. Which means the message is stored both in-memory, and on disk, and therefore preserved if the server dies or restarts.

The default value is `2` (persistent). During a send operation, Publishers can override the delivery mode of messages so that, for example, transient messages can be sent over a durable queue.

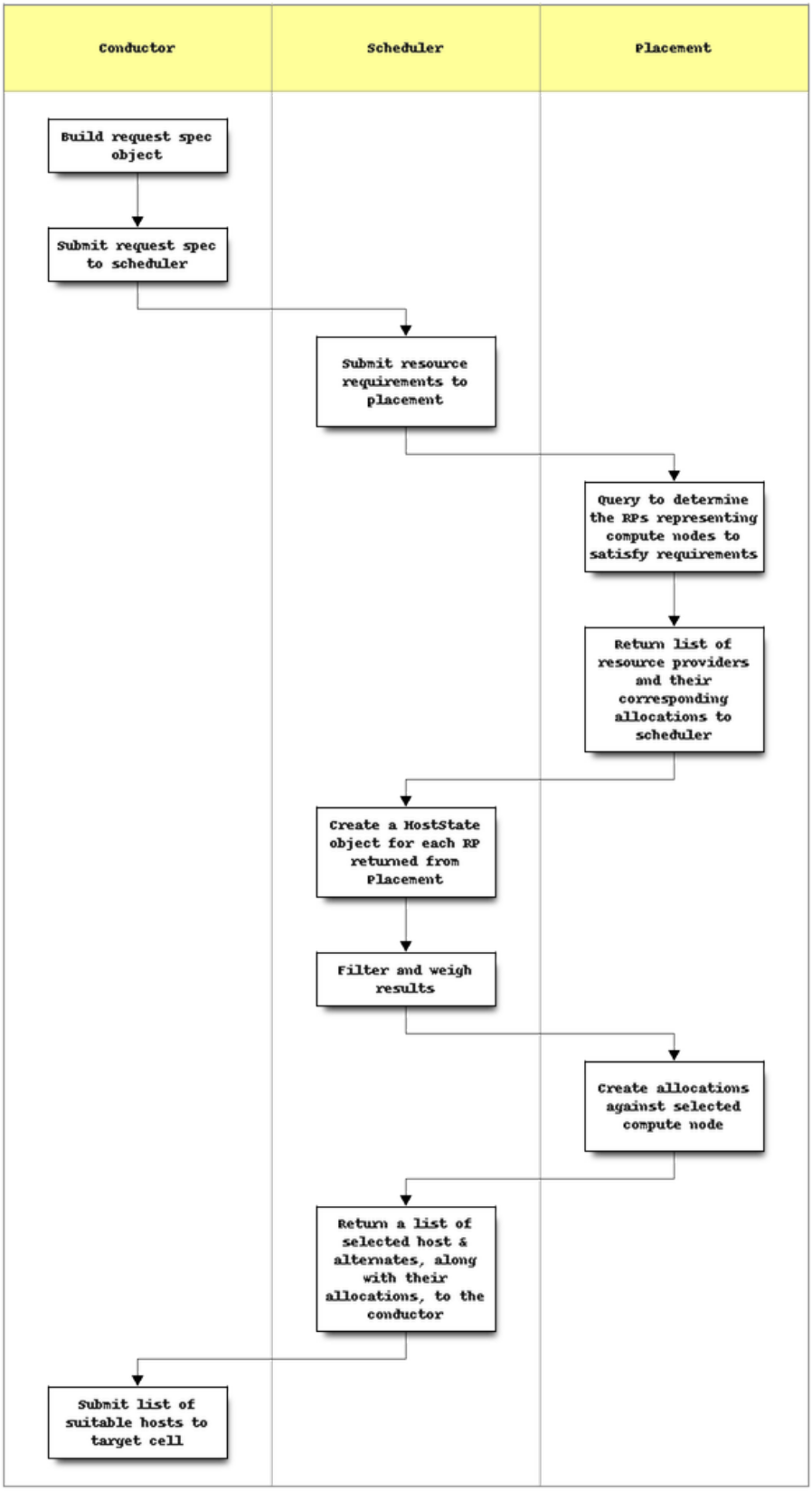
4.35 Scheduling

This is an overview of how scheduling works in nova from Pike onwards. For information on the scheduler itself, refer to *Filter Scheduler*. For an overview of why weve changed how the scheduler works, refer to *Scheduler Evolution*.

4.35.1 Overview

The scheduling process is described below.

Note: This is current as of the 16.0.0 Pike release. Any mention of alternative hosts passed between the scheduler and conductor(s) is future work.


Conductor
Scheduler
Placement
Build request spec object
Submit request spec to scheduler
Submit resource requirements to placement
Query to determine the RPs representing compute nodes to satisfy requirements
Return list of resource providers and their corresponding allocations to scheduler
Create a HostState object for each RP returned from Placement
Filter and weigh results
Create allocations against selected compute node
Return a list of selected host & alternates, along with their allocations, to the conductor
Submit list of suitable hosts to target cell

As the above diagram illustrates, scheduling works like so:

1. Scheduler gets a request spec from the super conductor, containing resource requirements. The super conductor operates at the top level of a deployment, as contrasted with the cell conductor, which operates within a particular cell.
2. Scheduler sends those requirements to placement.
3. Placement runs a query to determine the resource providers (in this case, compute nodes) that can satisfy those requirements.
4. Placement then constructs a data structure for each compute node as documented in the spec. The data structure contains summaries of the matching resource provider information for each compute node, along with the AllocationRequest that will be used to claim the requested resources if that compute node is selected.
5. Placement returns this data structure to the Scheduler.
6. The Scheduler creates HostState objects for each compute node contained in the provider summaries. These HostState objects contain the information about the host that will be used for subsequent filtering and weighing.
7. Since the request spec can specify one or more instances to be scheduled. The Scheduler repeats the next several steps for each requested instance.
8. Scheduler runs these HostState objects through the filters and weighers to further refine and rank the hosts to match the request.
9. Scheduler then selects the HostState at the top of the ranked list, and determines its matching AllocationRequest from the data returned by Placement. It uses that AllocationRequest as the body of the request sent to Placement to claim the resources.
10. If the claim is not successful, that indicates that another process has consumed those resources, and the host is no longer able to satisfy the request. In that event, the Scheduler moves on to the next host in the list, repeating the process until it is able to successfully claim the resources.
11. Once the Scheduler has found a host for which a successful claim has been made, it needs to select a number of alternate hosts. These are hosts from the ranked list that are in the same cell as the selected host, which can be used by the cell conductor in the event that the build on the selected host fails for some reason. The number of alternates is determined by the configuration option *scheduler.max_attempts*.
12. Scheduler creates two list structures for each requested instance: one for the hosts (selected + alternates), and the other for their matching AllocationRequests.
13. To create the alternates, Scheduler determines the cell of the selected host. It then iterates through the ranked list of HostState objects to find a number of additional hosts in that same cell. It adds those hosts to the host list, and their AllocationRequest to the allocation list.
14. Once those lists are created, the Scheduler has completed what it needs to do for a requested instance.
15. Scheduler repeats this process for any additional requested instances. When all instances have been scheduled, it creates a 2-tuple to return to the super conductor, with the first element of the tuple being a list of lists of hosts, and the second being a list of lists of the AllocationRequests.
16. Scheduler returns that 2-tuple to the super conductor.
17. For each requested instance, the super conductor determines the cell of the selected host. It then sends a 2-tuple of ([hosts], [AllocationRequests]) for that instance to the target cell conductor.
18. Target cell conductor tries to build the instance on the selected host. If it fails, it uses the AllocationRequest data for that host to unclaim the resources for the selected host. It then iterates through the list of alternates by first attempting to claim the resources, and if successful, building the instance on that host. Only when all alternates fail does the build request fail.

4.36 Scheduler Evolution

Evolving the scheduler has been a priority item over several releases: http://specs.openstack.org/openstack/nova-specs/#priorities

The scheduler has become tightly coupled with the rest of nova, limiting its capabilities, accuracy, flexibility and maintainability. The goal of scheduler evolution is to bring about a better separation of concerns between scheduling functionality and the rest of nova.

Once this effort has completed, its conceivable that the nova-scheduler could become a separate git repo, outside of nova but within the compute project. This is not the current focus.

4.36.1 Problem Use Cases

Many users are wanting to do more advanced things with the scheduler, but the current architecture is not ready to support those use cases in a maintainable way. A few examples will help to illustrate where the scheduler falls short:

Cross Project Affinity

It can be desirable, when booting from a volume, to use a compute node that is close to the shared storage where that volume is. Similarly, for the sake of performance, it can be desirable to use a compute node that is in a particular location in relation to a pre-created port.

Filter Scheduler Alternatives

For certain use cases, radically different schedulers may perform much better than the filter scheduler. We should not block this innovation. It is unreasonable to assume a single scheduler will work for all use cases.

However, to enable this kind of innovation in a maintainable way, a single strong scheduler interface is required.

Project Scale issues

There are many interesting ideas for new schedulers, like the solver scheduler, and frequent requests to add new filters and weights to the scheduling system. The current nova team does not have the bandwidth to deal with all these requests. A dedicated scheduler team could work on these items independently of the rest of nova.

The tight coupling that currently exists makes it impossible to work on the scheduler in isolation. A stable interface is required before the code can be split out.

4.36.2 Key areas we are evolving

Here we discuss, at a high level, areas that are being addressed as part of the scheduler evolution work.

Versioning Scheduler Placement Interfaces

At the start of kilo, the scheduler is passed a set of dictionaries across a versioned RPC interface. The dictionaries can create problems with the backwards compatibility needed for live-upgrades.

Luckily we already have the oslo.versionedobjects infrastructure we can use to model this data in a way that can be versioned across releases.

This effort is mostly focusing around the request_spec. See, for example, this spec.

Sending host and node stats to the scheduler

Periodically nova-compute updates the scheduler state stored in the database.

We need a good way to model the data that is being sent from the compute nodes into the scheduler, so over time, the scheduler can move to having its own database.

This is linked to the work on the resource tracker.

Updating the Scheduler about other data

Over time, its possible that we need to send cinder and neutron data, so the scheduler can use that data to help pick a nova-compute host.

Resource Tracker

The recent work to add support for NUMA and PCI pass through have shown we have no good pattern to extend the resource tracker. Ideally we want to keep the innovation inside the nova tree, but we also need it to be easier.

This is very related to the effort to re-think how we model resources, as covered by discussion about resource providers.

Parallelism and Concurrency

The current design of the nova-scheduler is very racy, and can lead to excessive numbers of build retries before the correct host is found. The recent NUMA features are particularly impacted by how the scheduler works. All this has lead to many people running only a single nova-scheduler process configured to use a very small greenthread pool.

The work on cells v2 will mean that we soon need the scheduler to scale for much larger problems. The current scheduler works best with less than 1k nodes but we will need the scheduler to work with at least 10k nodes.

Various ideas have been discussed to reduce races when running multiple nova-scheduler processes. One idea is to use two-phase commit style resource tracker claims. Another idea involves using incremental updates so it is more efficient to keep the schedulers state up to date, potentially using Kafka.

For more details, see the backlog spec that describes more of the details around this problem.

4.37 Services, Managers and Drivers

The responsibilities of Services, Managers, and Drivers, can be a bit confusing to people that are new to nova. This document attempts to outline the division of responsibilities to make understanding the system a little bit easier.

Currently, Managers and Drivers are specified by flags and loaded using utils.load_object(). This method allows for them to be implemented as singletons, classes, modules or objects. As long as the path specified by the flag leads to an object (or a callable that returns an object) that responds to getattr, it should work as a manager or driver.

4.37.1 The `nova.service` Module

Generic Node base class for all workers that run on hosts.

class nova.service.**Service**(*host*, *binary*, *topic*, *manager*, *report_interval=None*, *periodic_enable=None*, *periodic_fuzzy_delay=None*, *periodic_interval_max=None*, **args*, ***kwargs*)

Bases: `oslo_service.service.Service`

Service object for binaries running on hosts.

A service takes a manager and enables rpc by listening to queues based on topic. It also periodically runs tasks on the manager and reports its state to the database services table.

basic_config_check()
Perform basic config checks before starting processing.

classmethod create(*host=None*, *binary=None*, *topic=None*, *manager=None*, *report_interval=None*, *periodic_enable=None*, *periodic_fuzzy_delay=None*, *periodic_interval_max=None*)
Instantiates class and passes back application object.

Parameters

- **host** – defaults to CONF.host
- **binary** – defaults to basename of executable
- **topic** – defaults to bin_name - nova- part
- **manager** – defaults to CONF.<topic>_manager
- **report_interval** – defaults to CONF.report_interval
- **periodic_enable** – defaults to CONF.periodic_enable
- **periodic_fuzzy_delay** – defaults to CONF.periodic_fuzzy_delay
- **periodic_interval_max** – if set, the max time to wait between runs

kill()
Destroy the service object in the datastore.

NOTE: Although this method is not used anywhere else than tests, it is convenient to have it here, so the tests might easily and in clean way stop and remove the service_ref.

periodic_tasks(*raise_on_error=False*)
Tasks to be run at a periodic interval.

reset()
reset the service.

start()
Start the service.

This includes starting an RPC service, initializing periodic tasks, etc.

stop()
stop the service and clean up.

class nova.service.**WSGIService**(*name*, *loader=None*, *use_ssl=False*, *max_url_len=None*)
Bases: oslo_service.service.Service

Provides ability to launch API from a paste configuration.

reset()
Reset the following:

- server greenpool size to default
- service version cache
- cell cache holding database transaction context managers

Returns None

start()
Start serving this service using loaded configuration.

Also, retrieve updated port number in case 0 was passed in, which indicates a random port should be used.

Returns None

stop()
Stop serving this API.

Returns None

wait()
Wait for the service to stop serving this API.

Returns None

nova.service.**assert_eventlet_uses_monotonic_clock**()

nova.service.**process_launcher**()

nova.service.**serve**(*server*, *workers=None*)

nova.service.**setup_profiler**(*binary*, *host*)

nova.service.**wait**()

4.37.2 The nova.manager Module

Base Manager class.

Managers are responsible for a certain aspect of the system. It is a logical grouping of code relating to a portion of the system. In general other components should be using the manager to make changes to the components that it is responsible for.

For example, other components that need to deal with volumes in some way, should do so by calling methods on the VolumeManager instead of directly changing fields in the database. This allows us to keep all of the code relating to volumes in the same place.

We have adopted a basic strategy of Smart managers and dumb data, which means rather than attaching methods to data objects, components should call manager methods that act on the data.

Methods on managers that can be executed locally should be called directly. If a particular method must execute on a remote host, this should be done via rpc to the service that wraps the manager

Managers should be responsible for most of the db access, and non-implementation specific data. Anything implementation specific that cant be generalized should be done by the Driver.

In general, we prefer to have one manager with multiple drivers for different implementations, but sometimes it makes sense to have multiple managers. You can think of it this way: Abstract different overall strategies at the manager level(FlatNetwork vs VlanNetwork), and different implementations at the driver level(LinuxNetDriver vs CiscoNetDriver).

Managers will often provide methods for initial setup of a host or periodic tasks to a wrapping service.

This module provides Manager, a base class for managers.

class nova.manager.**Manager**(*host=None*, *service_name='undefined'*)
Bases: nova.db.base.Base, nova.manager.PeriodicTasks

cleanup_host()
Hook to do cleanup work when the service shuts down.

Child classes should override this method.

init_host()
Hook to do additional manager initialization when one requests the service be started. This is called before any service record is created.

Child classes should override this method.

periodic_tasks(*context*, *raise_on_error=False*)
Tasks to be run at a periodic interval.

post_start_hook()
Hook to provide the manager the ability to do additional start-up work immediately after a service creates RPC consumers and starts running.

Child classes should override this method.

pre_start_hook()
Hook to provide the manager the ability to do additional start-up work before any RPC queues/consumers are created. This is called after other initialization has succeeded and a service record is created.

Child classes should override this method.

reset()
Hook called on SIGHUP to signal the manager to re-read any dynamic configuration or do any reconfiguration tasks.

class nova.manager.**ManagerMeta**(*names*, *bases*, *dict_*)
Bases: nova.profiler.NoopMeta, oslo_service.periodic_task._PeriodicTasksMeta

Metaclass to trace all children of a specific class.

This metaclass wraps every public method (not starting with _ or __) of the class using it. All children classes of the class using ManagerMeta will be profiled as well.

Adding this metaclass requires that the __trace_args__ attribute be added to the class we want to modify. That attribute is a dictionary with one mandatory key: name. name defines the name of the action to be traced (for example, wsgi, rpc, db).

The OSprofiler-based tracing, although, will only happen if profiler instance was initiated somewhere before in the thread, that can only happen if profiling is enabled in nova.conf and the API call to Nova API contained specific headers.

class nova.manager.**PeriodicTasks**
Bases: oslo_service.periodic_task.PeriodicTasks

4.37.3 Implementation-Specific Drivers

A manager will generally load a driver for some of its tasks. The driver is responsible for specific implementation details. Anything running shell commands on a host, or dealing with other non-python code should probably be happening in a driver.

Drivers should not touch the database as the database management is done inside *nova-conductor*.

It usually makes sense to define an Abstract Base Class for the specific driver (i.e. VolumeDriver), to define the methods that a different driver would need to implement.

4.38 Nova Stable REST API

This document describes both the current state of the Nova REST API – as of the Pike release – and also attempts to describe how the Nova team evolved the REST APIs implementation over time and removed some of the cruft that has

crept in over the years.

4.38.1 Background

Nova used to include two distinct frameworks for exposing REST API functionality. Older code is called the v2 API and existed in the /nova/api/openstack/compute/legacy_v2/ directory. This code tree was totally removed during Netwon release time frame (14.0.0 and later). Newer code is called the v2.1 API and exists in the /nova/api/openstack/compute directory.

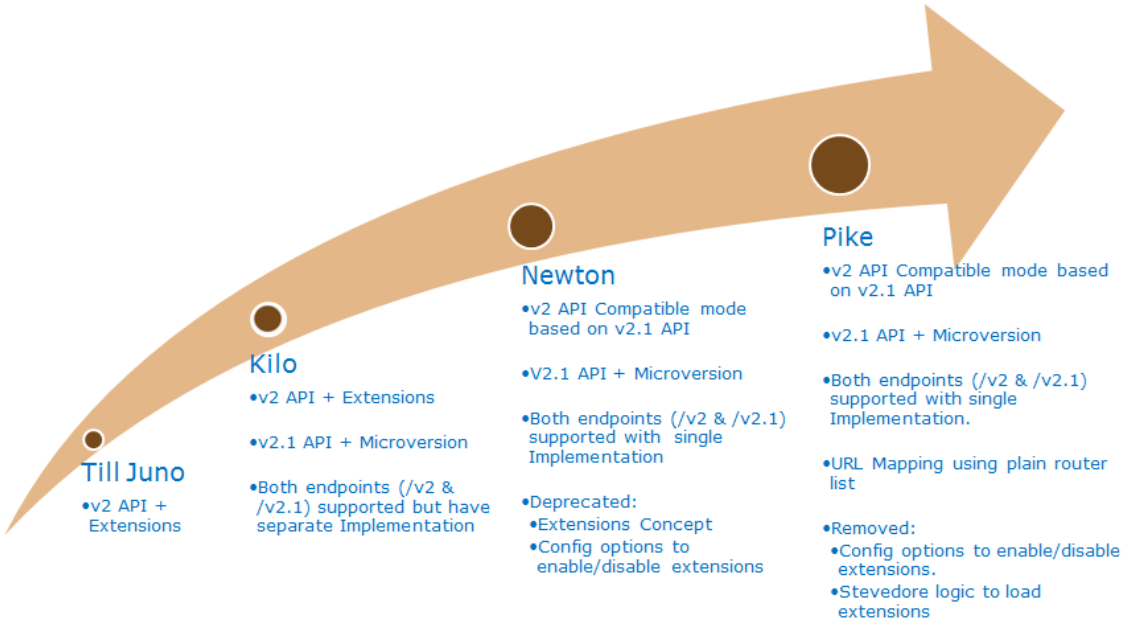
The v2 API is the old Nova REST API. It is mostly replaced by v2.1 API.

The v2.1 API is the new Nova REST API with a set of improvements which includes Microversion and standardized validation of inputs using JSON-Schema. Also the v2.1 API is totally backwards compatible with the v2 API (That is the reason we call it as v2.1 API).

4.38.2 Current Stable API

- Nova v2.1 API + Microversion (v2.1 APIs are backward-compatible with v2 API, but more strict validation)
- /v2 & /v2.1 endpoint supported
- v2 compatible mode for old v2 users

4.38.3 Evolution of Nova REST API



Nova v2 API + Extensions

Nova used to have v2 API. In v2 API, there was a concept called extension. An operator can use it to enable/disable part of Nova REST API based on requirements. An end user may query the /extensions API to discover what *API functionality* is supported by the Nova deployment.

Unfortunately, because v2 API extensions could be enabled or disabled from one deployment to another – as well as custom API extensions added to one deployment and not another – it was impossible for an end user to know what the OpenStack Compute API actually included. No two OpenStack deployments were consistent, which made cloud interoperability impossible.

In the Newton release, stevedore loading of API extension plugins was deprecated and marked for removal.

In the Newton release, v2 API code base has been removed and /v2 endpoints were directed to v2.1 code base.

v2 API compatibility mode based on v2.1 API

v2.1 API is exactly same as v2 API except strong input validation with no additional request parameter allowed and Microversion feature. Since Newton, /v2 endpoint also started using v2.1 API implementation. But to keep the backward compatibility of v2 API, /v2 endpoint should not return error on additional request parameter or any new headers for Microversion. v2 API must be same as it has been since starting.

To achieve that behavior legacy v2 compatibility mode has been introduced. v2 compatibility mode is based on v2.1 implementation with below difference:

- Skip additionalProperties checks in request body
- Ignore Microversion headers in request
- No Microversion headers in response

Nova v2.1 API + Microversion

In the Kilo release, nova v2.1 API has been released. v2.1 API is supposed to be backward compatible with v2 API with strong input validation using JSON Schema.

v2.1 API comes up with microversion concept which is a way to version the API changes. Each new feature or modification in API has to done via microversion bump.

API extensions concept was deprecated from the v2.1 API, are no longer needed to evolve the REST API, and no new API functionality should use the API extension classes to implement new functionality. Instead, new API functionality should be added via microversion concept and use the microversioning decorators to add or change the REST API.

v2.1 API had plugin framework which was using stevedore to load Nova REST API extensions instead of old V2 handcrafted extension load mechanism. There was an argument that the plugin framework supported extensibility in the Nova API to allow deployers to publish custom API resources.

In the Newton release, config options of blacklist and whitelist extensions and stevedore things were deprecated and marked for removal.

In Pike, stevedore based plugin framework has been removed and url mapping is done with plain router list. There is no more dynamic magic of detecting API implementation for url. See *Extending the API* for more information.

The /extensions API exposed the list of enabled API functions to users by GET method. However as the above, new API extensions should not be added to the list of this API. The /extensions API is frozen in Nova V2.1 API and is deprecated.

Things which are History now

As of the Pike release, many deprecated things have been removed and became history in Nova API world:

- v2 legacy framework
- API extensions concept

- stevedore magic to load the extension/plugin dynamically
- Configurable way to enable/disable APIs extensions

4.39 Threading model

All OpenStack services use *green thread* model of threading, implemented through using the Python eventlet and greenlet libraries.

Green threads use a cooperative model of threading: thread context switches can only occur when specific eventlet or greenlet library calls are made (e.g., sleep, certain I/O calls). From the operating systems point of view, each OpenStack service runs in a single thread.

The use of green threads reduces the likelihood of race conditions, but does not completely eliminate them. In some cases, you may need to use the `@lockutils.synchronized(...)` decorator to avoid races.

In addition, since there is only one operating system thread, a call that blocks that main thread will block the entire process.

4.39.1 Yielding the thread in long-running tasks

If a code path takes a long time to execute and does not contain any methods that trigger an eventlet context switch, the long-running thread will block any pending threads.

This scenario can be avoided by adding calls to the eventlet sleep method in the long-running code path. The sleep call will trigger a context switch if there are pending threads, and using an argument of 0 will avoid introducing delays in the case that there is only a single green thread:

```
from eventlet import greenthread
...
greenthread.sleep(0)
```

In current code, time.sleep(0) does the same thing as greenthread.sleep(0) if time module is patched through eventlet.monkey_patch(). To be explicit, we recommend contributors use `greenthread.sleep()` instead of `time.sleep()`.

4.39.2 MySQL access and eventlet

There are some MySQL DB API drivers for oslo.db, like PyMySQL, MySQL-python etc. PyMySQL is the default MySQL DB API driver for oslo.db, and it works well with eventlet. MySQL-python uses an external C library for accessing the MySQL database. Since eventlet cannot use monkey-patching to intercept blocking calls in a C library, so queries to the MySQL database will block the main thread of a service.

The Diablo release contained a thread-pooling implementation that did not block, but this implementation resulted in a bug and was removed.

See this mailing list thread for a discussion of this issue, including a discussion of the impact on performance.

4.40 ComputeDriver.update_provider_tree

This provides details on the `ComputeDriver` abstract method `update_provider_tree` for developers implementing this method in their own virt drivers.

4.40.1 Background

In the movement towards using placement for scheduling and resource management, the virt driver method `get_available_resource` was initially superseded by `get_inventory`, whereby the driver could specify its inventory in terms understood by placement. In Queens, a `get_traits` driver method was added. But `get_inventory` is limited to expressing only inventory (not traits or aggregates). And both of these methods are limited to the resource provider corresponding to the compute node.

Recent developments such as Nested Resource Providers necessitate the ability for the virt driver to have deeper control over what the resource tracker configures in placement on behalf of the compute node. This need is filled by the virt driver method `update_provider_tree` and its consumption by the resource tracker, allowing full control over the placement representation of the compute node and its associated providers.

4.40.2 The Method

`update_provider_tree` accepts the following parameters:

- A `nova.compute.provider_tree.ProviderTree` object representing all the providers in the tree associated with the compute node, and any sharing providers (those with the `MISC_SHARES_VIA_AGGREGATE` trait) associated via aggregate with any of those providers (but not *their* tree- or aggregate-associated providers), as currently known by placement. This object is fully owned by the `update_provider_tree` method, and can therefore be modified without locking/concurrency considerations. In other words, the parameter is passed *by reference* with the expectation that the virt driver will modify the object. Note, however, that it may contain providers not directly owned/controlled by the compute host. Care must be taken not to remove or modify such providers inadvertently. In addition, providers may be associated with traits and/or aggregates maintained by outside agents. The `update_provider_tree` method must therefore also be careful only to add/remove traits/aggregates it explicitly controls.
- String name of the compute node (i.e. `ComputeNode.hypervisor_hostname`) for which the caller is requesting updated provider information. Drivers may use this to help identify the compute node provider in the ProviderTree. Drivers managing more than one node (e.g. ironic) may also use it as a cue to indicate which node is being processed by the caller.
- Dictionary of `allocations` data of the form:

```
{ $CONSUMER_UUID: {
      # The shape of each "allocations" dict below is identical
      # to the return from GET /allocations/{consumer_uuid}
      "allocations": {
          $RP_UUID: {
              "generation": $RP_GEN,
              "resources": {
                  $RESOURCE_CLASS: $AMOUNT,
                  ...
              },
          },
          ...
      },
      "project_id": $PROJ_ID,
      "user_id": $USER_ID,
      "consumer_generation": $CONSUMER_GEN,
  },
  ...
}
```

  If `None`, and the method determines that any inventory needs to be moved (from one provider to another and/or to a different resource class), the `ReshapeNeeded` exception must be raised. Otherwise, this dict must be

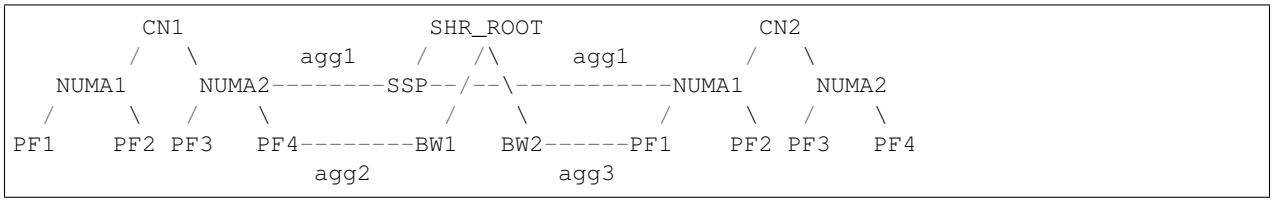
edited in place to indicate the desired final state of allocations. Drivers should *only* edit allocation records for providers whose inventories are being affected by the reshape operation. For more information about the reshape operation, refer to the spec.

The virt driver is expected to update the ProviderTree object with current resource provider and inventory information. When the method returns, the ProviderTree should represent the correct hierarchy of nested resource providers associated with this compute node, as well as the inventory, aggregates, and traits associated with those resource providers.

Note: Despite the name, a ProviderTree instance may in fact contain more than one tree. For purposes of this specification, the ProviderTree passed to `update_provider_tree` will contain:

- the entire tree associated with the compute node; and
- any sharing providers (those with the `MISC_SHARES_VIA_AGGREGATE` trait) which are associated via aggregate with any of the providers in the compute nodes tree. The sharing providers will be presented as lone roots in the ProviderTree, even if they happen to be part of a tree themselves.

Consider the example below. `SSP` is a shared storage provider and `BW1` and `BW2` are shared bandwidth providers; all three have the `MISC_SHARES_VIA_AGGREGATE` trait:



When `update_provider_tree` is invoked for `CN1`, it is passed a ProviderTree containing:

This method supersedes `get_inventory` and `get_traits`: if this method is implemented, neither `get_inventory` nor `get_traits` is used.

Driver implementations of `update_provider_tree` are expected to use public `ProviderTree` methods to effect changes to the provider tree passed in. Some of the methods which may be useful are as follows:

- `new_root`: Add a new root provider to the tree.
- `new_child`: Add a new child under an existing provider.
- `data`: Access information (name, UUID, parent, inventory, traits, aggregates) about a provider in the tree.
- `remove`: Remove a provider **and its descendants** from the tree. Use caution in multiple-ownership scenarios.
- `update_inventory`: Set the inventory for a provider.
- `add_traits`, `remove_traits`: Set/unset virt-owned traits for a provider.
- `add_aggregates`, `remove_aggregates`: Set/unset virt-owned aggregate associations for a provider.

Note: There is no supported mechanism for `update_provider_tree` to effect changes to allocations. This is intentional: in Nova, allocations are managed exclusively outside of virt. (Usually by the scheduler; sometimes - e.g.

for migrations - by the conductor.)

Porting from get_inventory

Virt driver developers wishing to move from `get_inventory` to `update_provider_tree` should use the `ProviderTree.update_inventory` method, specifying the compute node as the provider and the same inventory as returned by `get_inventory`. For example:

```
def get_inventory(self, nodename):
    inv_data = {
        'VCPU': { ... },
        'MEMORY_MB': { ... },
        'DISK_GB': { ... },
    }
    return inv_data
```

would become:

```
def update_provider_tree(self, provider_tree, nodename, allocations=None):
    inv_data = {
        'VCPU': { ... },
        'MEMORY_MB': { ... },
        'DISK_GB': { ... },
    }
    provider_tree.update_inventory(nodename, inv_data)
```

When reporting inventory for the standard resource classes `VCPU`, `MEMORY_MB` and `DISK_GB`, implementors of `update_provider_tree` may need to set the `allocation_ratio` and `reserved` values in the `inv_data` dict based on configuration to reflect changes on the compute for allocation ratios and reserved resource amounts back to the placement service.

Porting from get_traits

To replace `get_traits`, developers should use the `ProviderTree.add_traits` method, specifying the compute node as the provider and the same traits as returned by `get_traits`. For example:

```
def get_traits(self, nodename):
    traits = ['HW_CPU_X86_AVX', 'HW_CPU_X86_AVX2', 'CUSTOM_GOLD']
    return traits
```

would become:

```
def update_provider_tree(self, provider_tree, nodename, allocations=None):
    provider_tree.add_traits(
        nodename, 'HW_CPU_X86_AVX', 'HW_CPU_X86_AVX2', 'CUSTOM_GOLD')
```

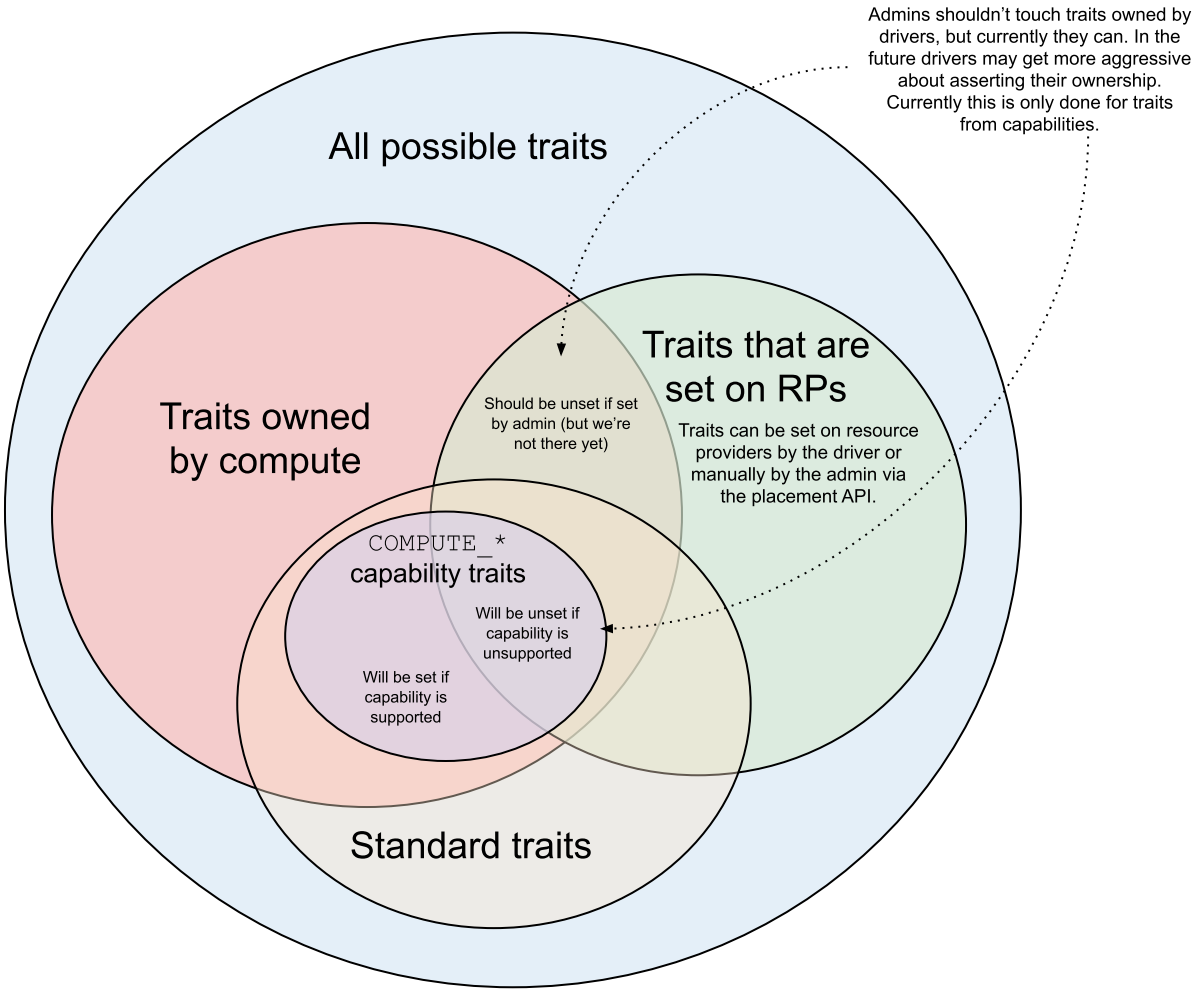
4.40.3 Taxonomy of traits and capabilities

There are various types of traits:

- Some are standard (registered in os-traits); others are custom.
- Some are owned by the compute service; others can be managed by operators.

- Some come from driver-supported capabilities, via a mechanism which was introduced to convert them to standard traits on the compute node resource provider. This mechanism is *documented in the configuration guide*.

This diagram may shed further light on how these traits relate to each other and how they are managed.



4.41 Upgrade checks

Nova provides automated *upgrade check tooling* to assist deployment tools in verifying critical parts of the deployment, especially when it comes to major changes during upgrades that require operator intervention.

This guide covers the background on novas upgrade check tooling, how it is used, and what to look for in writing new checks.

4.41.1 Background

Nova has historically supported offline database schema migrations (`nova-manage db sync`) and *online data migrations* during upgrades.

The `nova-status upgrade check` command was introduced in the 15.0.0 Ocata release to aid in the verification of two major required changes in that release, namely Placement and Cells v2.

Integration with the Placement service and deploying Cells v2 was optional starting in the 14.0.0 Newton release and made required in the Ocata release. The nova team working on these changes knew that there were required deployment changes to successfully upgrade to Ocata. In addition, the required deployment changes were not things that could simply be verified in a database migration script, e.g. a migration script should not make REST API calls to Placement.

So `nova-status upgrade check` was written to provide an automated pre-flight check to verify that required deployment steps were performed prior to upgrading to Ocata.

Reference the Ocata changes for implementation details.

4.41.2 Guidelines

- The checks should be able to run within a virtual environment or container. All that is required is a full configuration file, similar to running other `nova-manage` type administration commands. In the case of nova, this means having *`api_database`*, *`placement`*, etc sections configured.
- Candidates for automated upgrade checks are things in a projects upgrade release notes which can be verified via the database. For example, when upgrading to Cells v2 in Ocata, one required step was creating cell mappings for `cell0` and `cell1`. This can easily be verified by checking the contents of the `cell_mappings` table in the `nova_api` database.
- Checks will query the database(s) and potentially REST APIs (depending on the check) but should not expect to run RPC calls. For example, a check should not require that the `nova-compute` service is running on a particular host.
- Checks are typically meant to be run before re-starting and upgrading to new service code, which is how grenade uses them, but they can also be run as a *post-install verify step* which is how openstack-ansible also uses them. The high-level set of upgrade steps for upgrading nova in grenade is:
  - Install new code
  - Sync the database schema for new models (`nova-manage api_db sync`; `nova-manage db sync`)
  - Run the online data migrations (`nova-manage db online_data_migrations`)
  - Run the upgrade check (`nova-status upgrade check`)
  - Restart services with new code
- Checks must be idempotent so they can be run repeatedly and the results are always based on the latest data. This allows an operator to run the checks, fix any issues reported, and then iterate until the status check no longer reports any issues.
- Checks which cannot easily, or should not, be run within offline database migrations are a good candidate for these CLI-driven checks. For example, `instances` records are in the cell database and for each instance there should be a corresponding `request_specs` table entry in the `nova_api` database. A `nova-manage db online_data_migrations` routine was added in the Newton release to back-fill request specs for existing instances, and in Rocky an upgrade check was added to make sure all non-deleted instances have a request spec so compatibility code can be removed in Stein. In older releases of nova we would have added a blocker migration as part of the database schema migrations to make sure the online data migrations had been completed before the upgrade could proceed.

Note: Usage of `nova-status upgrade check` does not preclude the need for blocker migrations within a given database, but in the case of request specs the check spans multiple databases and was a better fit for the

nova-status tooling.

- All checks should have an accompanying upgrade release note.

4.41.3 Structure

There is no graph logic for checks, meaning each check is meant to be run independently of other checks in the same set. For example, a project could have five checks which run serially but that does not mean the second check in the set depends on the results of the first check in the set, or the third check depends on the second, and so on.

The base framework is fairly simple as can be seen from the initial change. Each check is registered in the `_upgrade_checks` variable and the `check` method executes each check and records the result. The most severe result is recorded for the final return code.

There are one of three possible results per check:

- `Success`: All upgrade readiness checks passed successfully and there is nothing to do.
- `Warning`: At least one check encountered an issue and requires further investigation. This is considered a warning but the upgrade may be OK.
- `Failure`: There was an upgrade status check failure that needs to be investigated. This should be considered something that stops an upgrade.

The `UpgradeCheckResult` object provides for adding details when there is a warning or failure result which generally should refer to how to resolve the failure, e.g. maybe `nova-manage db online_data_migrations` is incomplete and needs to be run again.

Using the cells v2 check as an example, there are really two checks involved:

1. Do the cell0 and cell1 mappings exist?
2. Do host mappings exist in the API database if there are compute node records in the cell database?

Failing either check results in a `Failure` status for that check and return code of `2` for the overall run.

The initial placement check provides an example of a warning response. In that check, if there are fewer resource providers in Placement than there are compute nodes in the cell database(s), the deployment may be underutilized because the `nova-scheduler` is using the Placement service to determine candidate hosts for scheduling.

Warning results are good for cases where scenarios are known to run through a rolling upgrade process, e.g. `nova-compute` being configured to report resource provider information into the Placement service. These are things that should be investigated and completed at some point, but might not cause any immediate failures.

The results feed into a standard output for the checks:

```
$ nova-status upgrade check
+-----------------------------------------------------+
| Upgrade Check Results                               |
+-----------------------------------------------------+
| Check: Cells v2                                     |
| Result: Success                                     |
| Details: None                                       |
+-----------------------------------------------------+
| Check: Placement API                                |
| Result: Failure                                     |
| Details: There is no placement-api endpoint in the  |
|          service catalog.                           |
+-----------------------------------------------------+
```

Note: Long-term the framework for upgrade checks will come from the oslo.upgradecheck library.

4.41.4 Other

Documentation

Each check should be documented in the *history section* of the CLI guide and have a release note. This is important since the checks can be run in an isolated environment apart from the actual deployed version of the code and since the checks should be idempotent, the history / change log is good for knowing what is being validated.

Backports

Sometimes upgrade checks can be backported to aid in pre-empting bugs on stable branches. For example, a check was added for bug 1759316 in Rocky which was also backported to stable/queens in case anyone upgrading from Pike to Queens would hit the same issue. Backportable checks are generally only made for latent bugs since someone who has already passed checks and upgraded to a given stable branch should not start failing after a patch release on that same branch. For this reason, any check being backported should have a release note with it.

Other projects

A community-wide goal for the Stein release is adding the same type of `$PROJECT-status upgrade check` tooling to other projects to ease in upgrading OpenStack across the board. So while the guidelines in this document are primarily specific to nova, they should apply generically to other projects wishing to incorporate the same tooling.

FAQs

1. How is the nova-status upgrade script packaged and deployed?

   There is a `console_scripts` entry for `nova-status` in the `setup.cfg` file.

2. Why are there multiple parts to the command structure, i.e. upgrade and check?

   This is an artifact of how the `nova-manage` command is structured which has categories of sub-commands, like `nova-manage db` is a sub-category made up of other sub-commands like `nova-manage db sync`. The `nova-status upgrade check` command was written in the same way for consistency and extensibility if other sub-commands need to be added later.

3. Where should the documentation live for projects other than nova?

   As part of the standard OpenStack project documentation guidelines the command should be documented under `doc/source/cli` in each project repo.

4. Why is the upgrade check command not part of the standard python-*client CLIs?

   The `nova-status` command was modeled after the `nova-manage` command which is meant to be admin-only and has direct access to the database, unlike other CLI packages like python-novaclient which requires a token and communicates with nova over the REST API. Because of this, it is also possible to write commands in `nova-manage` and `nova-status` that can work while the API service is down for maintenance.

5. Can upgrade checks only be for N-1 to N version upgrades?

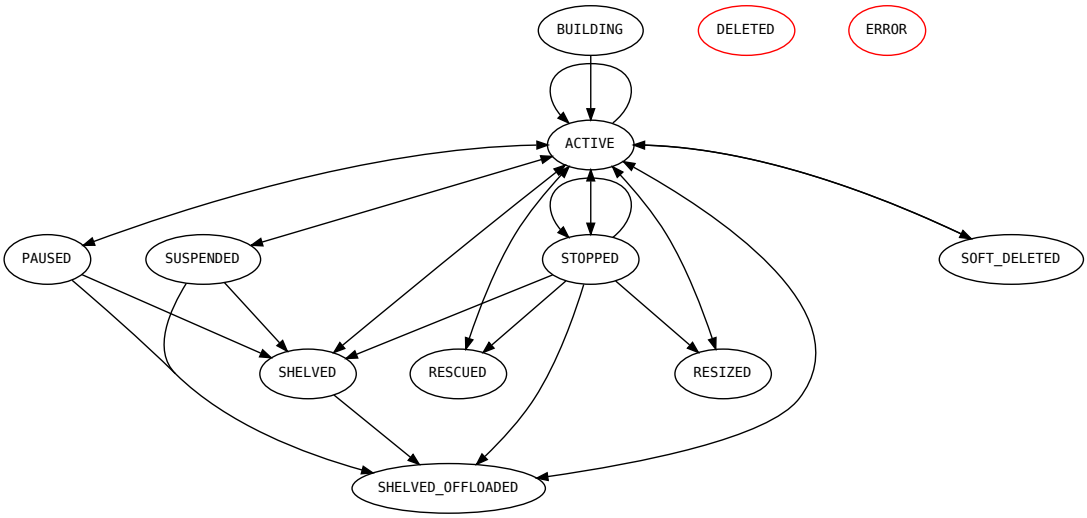
   No, not necessarily. The upgrade checks are also an essential part of fast-forward upgrades to make sure that as you roll through each release performing schema (data model) updates and data migrations that you are also

completing all of the necessary changes. For example, if you are fast forward upgrading from Ocata to Rocky, something could have been added, deprecated or removed in Pike or Queens and a pre-upgrade check is a way to make sure the necessary steps were taking while upgrading through those releases before restarting the Rocky code at the end.

4.42 Virtual Machine States and Transitions

The following diagrams and tables show the required virtual machine (VM) states and task states for various commands issued by the user.

4.42.1 Allowed State Transitions



All states are allowed to transition to DELETED and ERROR.

4.42.2 Requirements for Commands

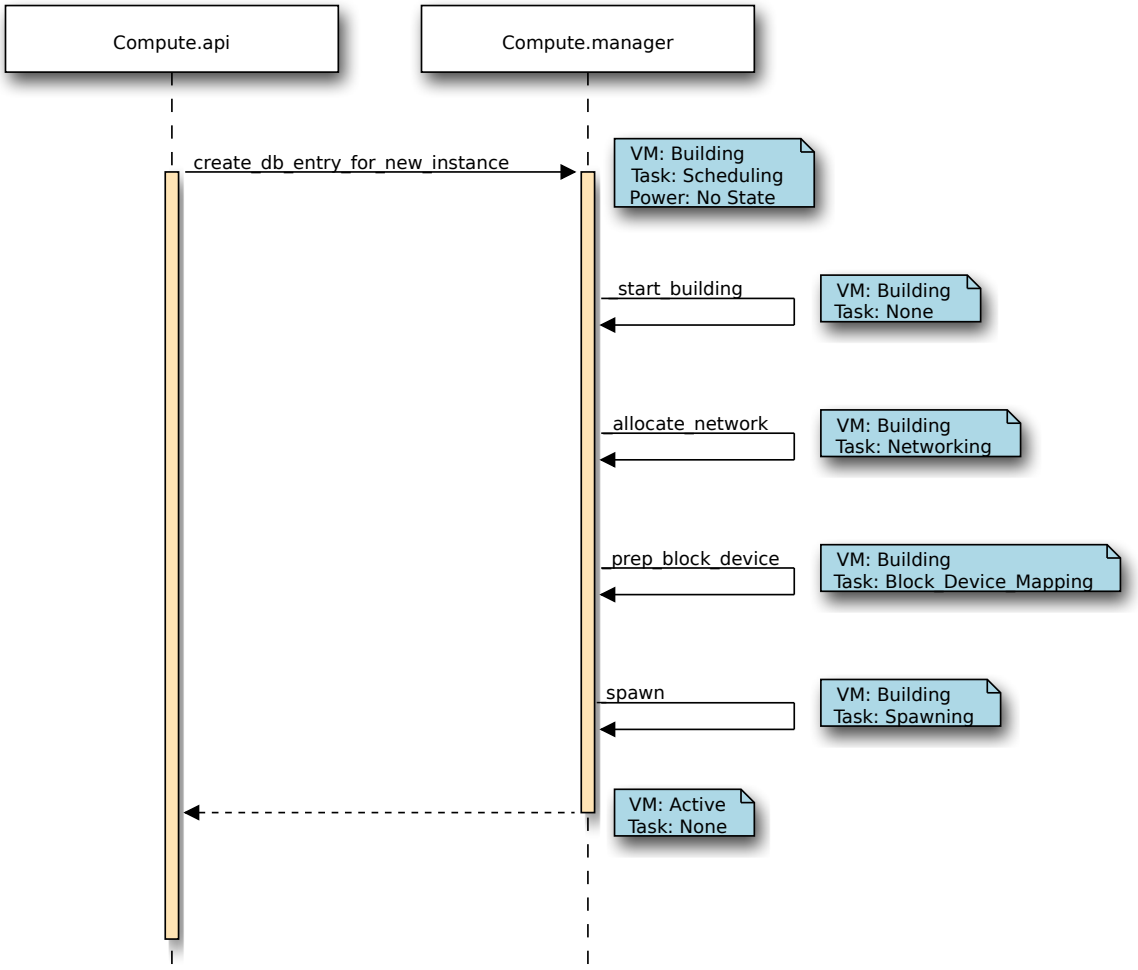
Command	Reqd VM States	Reqd Task States	Target State
pause	Active, Shutoff, Rescued	Resize Verify, unset	Paused
unpause	Paused	N/A	Active
suspend	Active, Shutoff	N/A	Suspended
resume	Suspended	N/A	Active
rescue	Active, Shutoff	Resize Verify, unset	Rescued
unrescue	Rescued	N/A	Active
set admin password	Active	N/A	Active
rebuild	Active, Shutoff	Resize Verify, unset	Active, Shutoff
force delete	Soft Deleted	N/A	Deleted
restore	Soft Deleted	N/A	Active
soft delete	Active, Shutoff, Error	N/A	Soft Deleted
delete	Active, Shutoff, Building, Rescued, Error	N/A	Deleted
backup	Active, Shutoff	N/A	Active, Shutoff
snapshot	Active, Shutoff	N/A	Active, Shutoff
start	Shutoff, Stopped	N/A	Active
stop	Active, Shutoff, Rescued	Resize Verify, unset	Stopped
reboot	Active, Shutoff, Rescued	Resize Verify, unset	Active
resize	Active, Shutoff	Resize Verify, unset	Resized
revert resize	Active, Shutoff	Resize Verify, unset	Active
confirm resize	Active, Shutoff	Resize Verify, unset	Active

4.42.3 VM states and Possible Commands

VM State	Commands
Paused	unpause
Suspended	resume
Active	set admin password, suspend, pause, rescue, rebuild, soft delete, delete, backup, snapshot, stop, reboot, resize, revert resize, confirm resize
Shutoff	suspend, pause, rescue, rebuild, soft delete, delete, backup, start, snapshot, stop, reboot, resize, revert resize, confirm resize
Rescued	unrescue, pause
Stopped	rescue, delete, start
Soft Deleted	force delete, restore
Error	soft delete, delete
Building	delete
Rescued	delete, stop, reboot

4.42.4 Create Instance States

The following diagram shows the sequence of VM states, task states, and power states when a new VM instance is created.



4.43 Scheduler hints versus flavor extra specs

People deploying and working on Nova often have questions about flavor extra specs and scheduler hints and what role they play in scheduling decisions, and which is a better choice for exposing capability to an end user of the cloud. There are several things to consider and it can get complicated. This document attempts to explain at a high level some of the major differences and drawbacks with both flavor extra specs and scheduler hints.

4.43.1 Extra Specs

In general flavor extra specs are specific to the cloud and how it is organized for capabilities, and should be abstracted from the end user. Extra specs are tied to *host aggregates* and a lot of them also define how a guest is created in the hypervisor, for example what the watchdog action is for a VM. Extra specs are also generally interchangeable with image properties when it comes to VM behavior, like the watchdog example. How that is presented to the user is via the name of the flavor, or documentation specifically for that deployment, e.g. instructions telling a user how to setup a baremetal instance.

4.43.2 Scheduler Hints

Scheduler hints, also known simply as hints, can be specified during server creation to influence the placement of the server by the scheduler depending on which scheduler filters are enabled. Hints are mapped to specific filters. For example, the `ServerGroupAntiAffinityFilter` scheduler filter is used with the `group` scheduler hint to indicate that the server being created should be a member of the specified anti-affinity group and the filter should place that server on a compute host which is different from all other current members of the group.

Hints are not more dynamic than flavor extra specs. The end user specifies a flavor and optionally a hint when creating a server, but ultimately what they can specify is static and defined by the deployment.

4.43.3 Similarities

- Both scheduler hints and flavor extra specs can be used by *scheduler filters*.
- Both are totally customizable, meaning there is no whitelist within Nova of acceptable hints or extra specs, unlike image properties[1].
- An end user cannot achieve a new behavior without deployer consent, i.e. even if the end user specifies the `group` hint, if the deployer did not configure the `ServerGroupAntiAffinityFilter` the end user cannot have the `anti-affinity` behavior.

4.43.4 Differences

- A servers host location and/or behavior can change when resized with a flavor that has different extra specs from those used to create the server. Scheduler hints can only be specified during server creation, not during resize or any other move operation, but the original hints are still applied during the move operation.
- The flavor extra specs used to create (or resize) a server can be retrieved from the compute API using the 2.47 microversion. As of the 19.0.0 Stein release, there is currently no way from the compute API to retrieve the scheduler hints used to create a server.

  Note: Exposing the hints used to create a server has been proposed[2]. Without this, it is possible to workaround the limitation by doing things such as including the scheduler hint in the server metadata so it can be retrieved via server metadata later.

- In the case of hints the end user can decide not to include a hint. On the other hand the end user cannot create a new flavor (by default policy) to avoid passing a flavor with an extra spec - the deployer controls the flavors.

4.43.5 Discoverability

When it comes to discoverability, by the default `os_compute_api:os-flavor-extra-specs:index` policy rule, flavor extra specs are more discoverable by the end user since they can list them for a flavor. However, one should not expect an average end user to understand what different extra specs mean as they are just a key/value pair. There is some documentation for some standard extra specs though[3]. However, that is not an exhaustive list and it does not include anything that different deployments would define for things like linking a flavor to a set of host aggregates, for example, when creating flavors for baremetal instances, or what the chosen *hypervisor driver* might support for flavor extra specs.

[1] https://opendev.org/openstack/nova/src/commit/fbe6f77bc1cb41f5d6cfc24ece54d3413f997aab/nova/objects/image_meta.py#L225
[2] https://review.opendev.org/#/c/440580/
[3] https://docs.openstack.org/nova/latest/user/flavors.html#extra-specs

Scheduler hints are less discoverable from an end user perspective than extra specs. There are some standard hints defined in the API request schema[4]. However:

1. Those hints are tied to scheduler filters and the scheduler filters are configurable per deployment, so for example the `JsonFilter` might not be enabled (it is not enabled by default), so the `query` hint would not do anything.
2. Scheduler hints are not restricted to just what is in that schema in the upstream nova code because of the `additionalProperties: True` entry in the schema. This allows deployments to define their own hints outside of that API request schema for their own *custom scheduler filters* which are not part of the upstream nova code.

4.43.6 Interoperability

The only way an end user can really use scheduler hints is based on documentation (or GUIs/SDKs) that a specific cloud deployment provides for their setup. So if **CloudA** defines a custom scheduler filter X and a hint for that filter in their documentation, an end user application can only run with that hint on that cloud and expect it to work as documented. If the user moves their application to **CloudB** which does not have that scheduler filter or hint, they will get different behavior.

So obviously both flavor extra specs and scheduler hints are not interoperable.

4.43.7 Which to use?

When it comes to defining a custom scheduler filter, you could use a hint or an extra spec. If you need a flavor extra spec anyway for some behavior in the hypervisor when creating the guest, or to be able to retrieve the original flavor extra specs used to create a guest later, then you might as well just use the extra spec. If you do not need that, then a scheduler hint may be an obvious choice, from an end user perspective, for exposing a certain scheduling behavior but it must be well documented and the end user should realize that hint might not be available in other clouds, and they do not have a good way of finding that out either. Long-term, flavor extra specs are likely to be more standardized than hints so ultimately extra specs are the recommended choice.

4.43.8 Footnotes

4.44 Filtering hosts by isolating aggregates

4.44.1 Background

I want to set up an aggregate `ABC` with hosts that allow you to run only certain licensed images. I could tag the aggregate with metadata such as `<LICENSED=WINDOWS>`. Then if I boot an instance with an image containing the property `<LICENSED=WINDOWS>`, it will land on one of the hosts in aggregate `ABC`. But if the user creates a new image which does not include `<LICENSED=WINDOWS>` metadata, an instance booted with that image could still land on a host in aggregate `ABC` as reported in launchpad bug 1677217. The *AggregateImagePropertiesIsolation* scheduler filter passes even though the aggregate metadata `<LICENSED=WINDOWS>` is not present in the image properties.

4.44.2 Solution

The above problem is addressed by blueprint placement-req-filter-forbidden-aggregates which was implemented in the 20.0.0 Train release.

[4] https://opendev.org/openstack/nova/src/commit/fbe6f77bc1cb41f5d6cfc24ece54d3413f997aab/nova/api/openstack/compute/schemas/scheduler_hints.py

The following example assumes you have configured aggregate ABC and added hosts HOST1 and HOST2 to it in Nova, and that you want to isolate those hosts to run only instances requiring Windows licensing.

1. Set the *scheduler.enable_isolated_aggregate_filtering* config option to true in nova.conf and restart the nova-scheduler service.

2. Add trait CUSTOM_LICENSED_WINDOWS to the resource providers for HOST1 and HOST2 in the Placement service.

   First create the CUSTOM_LICENSED_WINDOWS trait

   ```
   # openstack --os-placement-api-version 1.6 trait create CUSTOM_LICENSED_WINDOWS
   ```

   Discover the ID of HOST1, which is its resource provider UUID

   ```
   # host_id=$(openstack resource provider show <HOST1_UUID>)
   ```

   Start to build the command line by first collecting existing traits for HOST1

   ```
   # traits=$(openstack --os-placement-api-version 1.6 resource provider trait list -
   ↪f value $server_id | sed 's/^/--trait /')
   ```

   Replace HOST1s traits, adding CUSTOM_LICENSED_WINDOWS

   ```
   # openstack --os-placement-api-version 1.6 resource provider trait set $traits --
   ↪trait CUSTOM_LICENSED_WINDOWS $server_id
   ```

   Repeat the above steps for HOST2.

3. Add the trait:CUSTOM_LICENSED_WINDOWS=required metadata property to aggregate ABC.

   ```
   openstack --os-compute-api-version 2.53 aggregate set --property trait:CUSTOM_
   ↪LICENSED_WINDOWS=required ABC
   ```

As before, any instance spawned with a flavor or image containing trait:CUSTOM_LICENSED_WINDOWS=required will land on HOST1 or HOST2 because those hosts expose that trait.

However, now that the isolate_aggregates request filter is configured, any instance whose flavor or image **does not** contain trait:CUSTOM_LICENSED_WINDOWS=required will **not** land on HOST1 or HOST2 because aggregate ABC requires that trait.

The above example uses a CUSTOM_LICENSED_WINDOWS trait, but you can use any custom or standard trait in a similar fashion.

The filter supports the use of multiple traits across multiple aggregates. The combination of flavor and image metadata must require **all** of the traits configured on the aggregate in order to pass.

4.45 User Documentation

4.45.1 End user guide

Launch instances

Instances are virtual machines that run inside the cloud.

Before you can launch an instance, gather the following parameters:

- The **instance source** can be an image, snapshot, or block storage volume that contains an image or snapshot.

- A **name** for your instance.
- The **flavor** for your instance, which defines the compute, memory, and storage capacity of nova computing instances. A flavor is an available hardware configuration for a server. It defines the size of a virtual server that can be launched.
- Any **user data** files. A *user data* file is a special key in the metadata service that holds a file that cloud-aware applications in the guest instance can access. For example, one application that uses user data is the cloud-init system, which is an open-source package from Ubuntu that is available on various Linux distributions and that handles early initialization of a cloud instance.
- Access and security credentials, which include one or both of the following credentials:
  - A **key pair** for your instance, which are SSH credentials that are injected into images when they are launched. For the key pair to be successfully injected, the image must contain the `cloud-init` package. Create at least one key pair for each project. If you already have generated a key pair with an external tool, you can import it into OpenStack. You can use the key pair for multiple instances that belong to that project.
  - A **security group** that defines which incoming network traffic is forwarded to instances. Security groups hold a set of firewall policies, known as *security group rules*.
- If needed, you can assign a **floating (public) IP address** to a running instance to make it accessible from outside the cloud. See *Manage IP addresses*.
- You can also attach a block storage device, or **volume**, for persistent storage.

Note: Instances that use the default security group cannot, by default, be accessed from any IP address outside of the cloud. If you want those IP addresses to access the instances, you must modify the rules for the default security group.

After you gather the parameters that you need to launch an instance, you can launch it from an *image* or a *volume*. You can launch an instance directly from one of the available OpenStack images or from an image that you have copied to a persistent volume. The OpenStack Image service provides a pool of images that are accessible to members of different projects.

Gather parameters to launch an instance

Before you begin, source the OpenStack RC file.

1. Create a flavor.

   Creating a flavor is typically only available to administrators of a cloud because this has implications for scheduling efficiently in the cloud.

   ```
   $ openstack flavor create --ram 512 --disk 1 --vcpus 1 m1.tiny
   ```

2. List the available flavors.

   ```
   $ openstack flavor list
   ```

   Note the ID of the flavor that you want to use for your instance:

   ```
   +-----+-----------+-------+------+-----------+-------+-----------+
   | ID  | Name      |   RAM | Disk | Ephemeral | VCPUs | Is_Public |
   +-----+-----------+-------+------+-----------+-------+-----------+
   | 1   | m1.tiny   |   512 |    1 |         0 |     1 | True      |
   | 2   | m1.small  |  2048 |   20 |         0 |     1 | True      |
   ```

(continues on next page)

(continued from previous page)

```
| 3   | m1.medium |  4096 |   40 |         0 |     2 | True      |
| 4   | m1.large  |  8192 |   80 |         0 |     4 | True      |
| 5   | m1.xlarge | 16384 |  160 |         0 |     8 | True      |
+-----+-----------+-------+------+-----------+-------+-----------+
```

3. List the available images.

```
$ openstack image list
```

Note the ID of the image from which you want to boot your instance:

```
+--------------------------------------+---------------------------------+--------
↪+
| ID                                   | Name                            | Status␣
↪|
+--------------------------------------+---------------------------------+--------
↪+
| 397e713c-b95b-4186-ad46-6126863ea0a9 | cirros-0.3.5-x86_64-uec         | active␣
↪|
| df430cc2-3406-4061-b635-a51c16e488ac | cirros-0.3.5-x86_64-uec-kernel  | active␣
↪|
| 3cf852bd-2332-48f4-9ae4-7d926d50945e | cirros-0.3.5-x86_64-uec-ramdisk | active␣
↪|
+--------------------------------------+---------------------------------+--------
↪+
```

You can also filter the image list by using **grep** to find a specific image, as follows:

```
$ openstack image list | grep 'kernel'

| df430cc2-3406-4061-b635-a51c16e488ac | cirros-0.3.5-x86_64-uec-kernel  | active␣
↪|
```

4. List the available security groups.

```
$ openstack security group list
```

Note: If you are an admin user, this command will list groups for all tenants.

Note the ID of the security group that you want to use for your instance:

```
+--------------------------------------+---------+------------------------+-------
↪---------------------------+
| ID                                   | Name    | Description            |␣
↪Project                           |
+--------------------------------------+---------+------------------------+-------
↪---------------------------+
| b0d78827-0981-45ef-8561-93aee39bbd9f | default | Default security group |␣
↪5669caad86a04256994cdf755df4d3c1 |
| ec02e79e-83e1-48a5-86ad-14ab9a8c375f | default | Default security group |␣
↪1eaaf6ede7a24e78859591444abf314a |
+--------------------------------------+---------+------------------------+-------
↪---------------------------+
```

If you have not created any security groups, you can assign the instance to only the default security group.

You can view rules for a specified security group:

```
$ openstack security group rule list default
```

5. List the available key pairs, and note the key pair name that you use for SSH access.

```
$ openstack keypair list
```

Launch an instance

You can launch an instance from various sources.

Launch an instance from an image

Follow the steps below to launch an instance from an image.

1. After you gather required parameters, run the following command to launch an instance. Specify the server name, flavor ID, and image ID.

```
$ openstack server create --flavor FLAVOR_ID --image IMAGE_ID --key-name KEY_NAME␣
↪\
  --user-data USER_DATA_FILE --security-group SEC_GROUP_NAME --property KEY=VALUE␣
↪\
  INSTANCE_NAME
```

Optionally, you can provide a key name for access control and a security group for security. You can also include metadata key and value pairs. For example, you can add a description for your server by providing the `--property description="My Server"` parameter.

You can pass *user data* in a local file at instance launch by using the `--user-data USER-DATA-FILE` parameter.

Important: If you boot an instance with an INSTANCE_NAME greater than 63 characters, Compute truncates it automatically when turning it into a host name to ensure the correct work of dnsmasq. The corresponding warning is written into the `neutron-dnsmasq.log` file.

The following command launches the `MyCirrosServer` instance with the `m1.small` flavor (ID of `1`), `cirros-0.3.2-x86_64-uec` image (ID of `397e713c-b95b-4186-ad46-6126863ea0a9`), `default` security group, `KeyPair01` key, and a user data file called `cloudinit.file`:

```
$ openstack server create --flavor 1 --image 397e713c-b95b-4186-ad46-6126863ea0a9␣
↪\
  --security-group default --key-name KeyPair01 --user-data cloudinit.file \
  myCirrosServer
```

Depending on the parameters that you provide, the command returns a list of server properties.

```
+--------------------------------------+--------------------------------------------
↪-----+
| Field                                | Value                                      ␣
↪     |
+--------------------------------------+--------------------------------------------
↪-----+
```

(continues on next page)

(continued from previous page)

```
| OS-DCF:diskConfig                    | MANUAL                                 |
| OS-EXT-AZ:availability_zone          |                                        |
| OS-EXT-SRV-ATTR:host                 | None                                   |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | None                                   |
| OS-EXT-SRV-ATTR:instance_name        |                                        |
| OS-EXT-STS:power_state               | NOSTATE                                |
| OS-EXT-STS:task_state                | scheduling                             |
| OS-EXT-STS:vm_state                  | building                               |
| OS-SRV-USG:launched_at               | None                                   |
| OS-SRV-USG:terminated_at             | None                                   |
| accessIPv4                           |                                        |
| accessIPv6                           |                                        |
| addresses                            |                                        |
| adminPass                            | E4Ksozt4Efi8                           |
| config_drive                         |                                        |
| created                              | 2016-11-30T14:48:05Z                   |
| flavor                               | m1.tiny                                |
| hostId                               |                                        |
| id                                   | 89015cc9-bdf1-458a-8518-fdca2b4a5785   |
| image                                | cirros (397e713c-b95b-4186-ad46-
↪6126863ea0a9) |
| key_name                             | KeyPair01                              |
| name                                 | myCirrosServer                         |
| os-extended-volumes:volumes_attached | []                                     |
| progress                             | 0                                      |
| project_id                           | 5669caad86a04256994cdf755df4d3c1       |
| properties                           |                                        |
| security_groups                      | [{u'name': u'default'}]                |
| status                               | BUILD                                  |
| updated                              | 2016-11-30T14:48:05Z                   |
```

(continues on next page)

(continued from previous page)

```
| user_id                              | c36cec73b0e44876a4478b1e6cd749bb
↪      |
| metadata                             | {u'KEY': u'VALUE'}
↪      |
+--------------------------------------+------------------------------------------
↪-----+
```

A status of `BUILD` indicates that the instance has started, but is not yet online.

A status of `ACTIVE` indicates that the instance is active.

2. Copy the server ID value from the `id` field in the output. Use the ID to get server details or to delete your server.
3. Copy the administrative password value from the `adminPass` field. Use the password to log in to your server.
4. Check if the instance is online.

```
$ openstack server list
```

The list shows the ID, name, status, and private (and if assigned, public) IP addresses for all instances in the project to which you belong:

```
+--------------+----------------------+--------+------------+-------------+--------
↪----------+------------+
| ID           | Name                 | Status | Task State | Power State |
↪Networks          | Image Name |
+--------------+----------------------+--------+------------+-------------+--------
↪----------+------------+
| 84c6e57d...  | myCirrosServer       | ACTIVE | None       | Running     |
↪private=10.0.0.3 | cirros     |
| 8a99547e...  | myInstanceFromVolume | ACTIVE | None       | Running     |
↪private=10.0.0.4 | centos     |
+--------------+----------------------+--------+------------+-------------+--------
↪----------+------------+
```

If the status for the instance is ACTIVE, the instance is online.

5. To view the available options for the **`openstack server list`** command, run the following command:

```
$ openstack help server list
```

Note: If you did not provide a key pair, security groups, or rules, you can access the instance only from inside the cloud through VNC. Even pinging the instance is not possible.

Launch an instance from a volume

You can boot instances from a volume instead of an image.

To complete these tasks, use these parameters on the **`nova boot`** command:

Task	nova boot parameter	Information
Boot an instance from an image and attach a non-bootable volume.	`--block-device`	*Boot instance from image and attach non-bootable volume*
Create a volume from an image and boot an instance from that volume.	`--block-device`	*Create volume from image and boot instance*
Boot from an existing source image, volume, or snapshot.	`--block-device`	*Create volume from image and boot instance*
Attach a swap disk to an instance.	`--swap`	*Attach swap or ephemeral disk to an instance*
Attach an ephemeral disk to an instance.	`--ephemeral`	*Attach swap or ephemeral disk to an instance*

Note: To attach a volume to a running instance, refer to the Cinder documentation.

Note: The maximum limit on the number of disk devices allowed to attach to a single server is configurable with the option *compute.max_disk_devices_to_attach*.

Boot instance from image and attach non-bootable volume

Create a non-bootable volume and attach that volume to an instance that you boot from an image.

To create a non-bootable volume, do not create it from an image. The volume must be entirely empty with no partition table and no file system.

1. Create a non-bootable volume.

```
$ openstack volume create --size 8 my-volume
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| attachments         | []                                   |
| availability_zone   | nova                                 |
| bootable            | false                                |
| consistencygroup_id | None                                 |
| created_at          | 2016-11-25T10:37:08.850997           |
| description         | None                                 |
| encrypted           | False                                |
| id                  | b8f7bbec-6274-4cd7-90e7-60916a5e75d4 |
| migration_status    | None                                 |
| multiattach         | False                                |
| name                | my-volume                            |
| properties          |                                      |
| replication_status  | disabled                             |
| size                | 8                                    |
| snapshot_id         | None                                 |
| source_volid        | None                                 |
| status              | creating                             |
| type                | None                                 |
| updated_at          | None                                 |
| user_id             | 0678735e449149b0a42076e12dd54e28     |
+---------------------+--------------------------------------+
```

2. List volumes.

```
$ openstack volume list
+--------------------------------------+--------------+-----------+------+--------
↪-----+
| ID                                   | Name         | Status    | Size |
↪Attached to |
+--------------------------------------+--------------+-----------+------+--------
↪-----+
| b8f7bbec-6274-4cd7-90e7-60916a5e75d4 | my-volume    | available |    8 |
↪     |
+--------------------------------------+--------------+-----------+------+--------
↪-----+
```

3. Boot an instance from an image and attach the empty volume to the instance.

```
$ nova boot --flavor 2 --image 98901246-af91-43d8-b5e6-a4506aa8f369 \
  --block-device source=volume,id=d620d971-b160-4c4e-8652-2513d74e2080,
↪dest=volume,shutdown=preserve \
  myInstanceWithVolume
+--------------------------------------+----------------------------------------
↪--+
| Property                             | Value
↪  |
+--------------------------------------+----------------------------------------
↪--+
| OS-DCF:diskConfig                    | MANUAL
↪  |
| OS-EXT-AZ:availability_zone          | nova
↪  |
| OS-EXT-SRV-ATTR:host                 | -
↪  |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | -
↪  |
| OS-EXT-SRV-ATTR:instance_name        | instance-00000004
↪  |
| OS-EXT-STS:power_state               | 0
↪  |
| OS-EXT-STS:task_state                | scheduling
↪  |
| OS-EXT-STS:vm_state                  | building
↪  |
| OS-SRV-USG:launched_at               | -
↪  |
| OS-SRV-USG:terminated_at             | -
↪  |
| accessIPv4                           |
↪  |
| accessIPv6                           |
↪  |
| adminPass                            | ZaiYeC8iucgU
↪  |
| config_drive                         |
↪  |
| created                              | 2014-05-09T16:34:50Z
↪  |
| flavor                               | m1.small (2)
↪  |
| hostId                               |
↪  |
```

(continues on next page)

(continued from previous page)

```
| id                                   | 1e1797f3-1662-49ff-ae8c-a77e82ee1571
↪  |
| image                                | cirros-0.3.5-x86_64-uec (98901246-af91-..
↪. |
| key_name                             | -
↪  |
| metadata                             | {}
↪  |
| name                                 | myInstanceWithVolume
↪  |
| os-extended-volumes:volumes_attached | [{"id": "d620d971-b160-4c4e-8652-2513d7..
↪. |
| progress                             | 0
↪  |
| security_groups                      | default
↪  |
| status                               | BUILD
↪  |
| tenant_id                            | ccef9e62b1e645df98728fb2b3076f27
↪  |
| updated                              | 2014-05-09T16:34:51Z
↪  |
| user_id                              | fef060ae7bfd4024b3edb97dff59017a
↪  |
+--------------------------------------+------------------------------------------
↪--+
```

Create volume from image and boot instance

You can create a volume from an existing image, volume, or snapshot. This procedure shows you how to create a volume from an image, and use the volume to boot an instance.

1. List the available images.

```
$ openstack image list
+------------------+---------------------------------+--------+
| ID               | Name                            | Status |
+------------------+---------------------------------+--------+
| 484e05af-a14...  | Fedora-x86_64-20-20131211.1-sda | active |
| 98901246-af9...  | cirros-0.3.5-x86_64-uec         | active |
| b6e95589-7eb...  | cirros-0.3.5-x86_64-uec-kernel  | active |
| c90893ea-e73...  | cirros-0.3.5-x86_64-uec-ramdisk | active |
+------------------+---------------------------------+--------+
```

Note the ID of the image that you want to use to create a volume.

If you want to create a volume to a specific storage backend, you need to use an image which has *cinder_img_volume_type* property. In this case, a new volume will be created as *storage_backend1* volume type.

```
$ openstack image show 98901246-af9d-4b61-bea8-09cc6dc41829
+------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | ee1eca47dc88f4879d8a229cc70a07c6                     |
| container_format | bare                                                 |
```

(continues on next page)

(continued from previous page)

```
| created_at       | 2016-10-08T14:59:05Z                                          |
| disk_format      | qcow2                                                         |
| file             | /v2/images/9fef3b2d-c35d-4b61-bea8-09cc6dc41829/file          |
| id               | 98901246-af9d-4b61-bea8-09cc6dc41829                          |
| min_disk         | 0                                                             |
| min_ram          | 0                                                             |
| name             | cirros-0.3.5-x86_64-uec                                       |
| owner            | 8d8ef3cdf2b54c25831cbb409ad9ae86                              |
| protected        | False                                                         |
| schema           | /v2/schemas/image                                             |
| size             | 13287936                                                      |
| status           | active                                                        |
| tags             |                                                               |
| updated_at       | 2016-10-19T09:12:52Z                                          |
| virtual_size     | None                                                          |
| visibility       | public                                                        |
+------------------+---------------------------------------------------------------+
```

2. List the available flavors.

```
$ openstack flavor list
+-----+-----------+-------+------+-----------+-------+-----------+
| ID  | Name      |   RAM | Disk | Ephemeral | VCPUs | Is_Public |
+-----+-----------+-------+------+-----------+-------+-----------+
| 1   | m1.tiny   |   512 |    1 |         0 |     1 | True      |
| 2   | m1.small  |  2048 |   20 |         0 |     1 | True      |
| 3   | m1.medium |  4096 |   40 |         0 |     2 | True      |
| 4   | m1.large  |  8192 |   80 |         0 |     4 | True      |
| 5   | m1.xlarge | 16384 |  160 |         0 |     8 | True      |
+-----+-----------+-------+------+-----------+-------+-----------+
```

Note the ID of the flavor that you want to use to create a volume.

3. To create a bootable volume from an image and launch an instance from this volume, use the `--block-device` parameter with the `nova boot` command.

For example:

```
$ nova boot --flavor FLAVOR --block-device \
  source=SOURCE,id=ID,dest=DEST,size=SIZE,shutdown=PRESERVE,bootindex=INDEX \
  NAME
```

The parameters are:

- `--flavor` The flavor ID or name.
- `--block-device` source=SOURCE,id=ID,dest=DEST,size=SIZE,shutdown=PRESERVE,bootindex=INDEX

  source=SOURCE The type of object used to create the block device. Valid values are `volume`, `snapshot`, `image`, and `blank`.

  id=ID The ID of the source object.

  dest=DEST The type of the target virtual device. Valid values are `volume` and `local`.

  size=SIZE The size of the volume that is created.

  shutdown={preserve|remove} What to do with the volume when the instance is deleted. `preserve` does not delete the volume. `remove` deletes the volume.

  bootindex=INDEX Orders the boot disks. Use `0` to boot from this volume.

- `NAME`. The name for the server.

See the nova boot command documentation and *Block Device Mapping in Nova* for more details on these parameters.

Note: As of the Stein release, the `openstack server create` command does not support creating a volume-backed server from a source image like the `nova boot` command. The next steps will show how to create a bootable volume from an image and then create a server from that boot volume using the `openstack server create` command.

---

4. Create a bootable volume from an image. Cinder makes a volume bootable when `--image` parameter is passed.

```
$ openstack volume create --image IMAGE_ID --size SIZE_IN_GB bootable_volume
```

Note: A bootable encrypted volume can also be created by adding the *–type ENCRYPTED_VOLUME_TYPE* parameter to the volume create command:

```
$ openstack volume create --type ENCRYPTED_VOLUME_TYPE --image IMAGE_ID --size␣
↪SIZE_IN_GB bootable_volume
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| attachments         | []                                   |
| availability_zone   | nova                                 |
| bootable            | false                                |
| consistencygroup_id | None                                 |
| created_at          | 2017-06-13T18:59:57.626872           |
| description         | None                                 |
| encrypted           | True                                 |
| id                  | ded57a86-5b51-43ab-b70e-9bc0f91ef4ab |
| multiattach         | False                                |
| name                | bootable_volume                      |
| properties          |                                      |
| replication_status  | None                                 |
| size                | 1                                    |
| snapshot_id         | None                                 |
| source_volid        | None                                 |
| status              | creating                             |
| type                | LUKS                                 |
| updated_at          | None                                 |
| user_id             | 459ae34ffcd94edab0c128ed616bb19f     |
+---------------------+--------------------------------------+
```

This requires an encrypted volume type, which must be created ahead of time by an admin. Refer to horizonadmin/manage-volumes.html#create-an-encrypted-volume-type. in the OpenStack Horizon Administration Guide.

5. Create a VM from previously created bootable volume. The volume is not deleted when the instance is terminated.

Note: The example here uses the `--volume` option for simplicity. The `--block-device-mapping` option could also be used for more granular control over the parameters. See the openstack server create documentation for details.

```
$ openstack server create --flavor 2 --volume VOLUME_ID myInstanceFromVolume
+--------------------------------------+--------------------------------+
| Field                                | Value                          |
+--------------------------------------+--------------------------------+
| OS-EXT-STS:task_state                | scheduling                     |
| image                                | Attempt to boot from volume    |
|                                      | - no image supplied            |
| OS-EXT-STS:vm_state                  | building                       |
| OS-EXT-SRV-ATTR:instance_name        | instance-00000003              |
| OS-SRV-USG:launched_at               | None                           |
| flavor                               | m1.small                       |
| id                                   | 2e65c854-dba9-4f68-8f08-fe3... |
| security_groups                      | [{u'name': u'default'}]        |
| user_id                              | 352b37f5c89144d4ad053413926... |
| OS-DCF:diskConfig                    | MANUAL                         |
| accessIPv4                           |                                |
| accessIPv6                           |                                |
| progress                             | 0                              |
| OS-EXT-STS:power_state               | 0                              |
| OS-EXT-AZ:availability_zone          | nova                           |
| config_drive                         |                                |
| status                               | BUILD                          |
| updated                              | 2014-02-02T13:29:54Z           |
| hostId                               |                                |
| OS-EXT-SRV-ATTR:host                 | None                           |
| OS-SRV-USG:terminated_at             | None                           |
| key_name                             | None                           |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | None                           |
| name                                 | myInstanceFromVolume           |
| adminPass                            | TzjqyGsRcJo9                   |
| tenant_id                            | f7ac731cc11f40efbc03a9f9e1d... |
| created                              | 2014-02-02T13:29:53Z           |
| os-extended-volumes:volumes_attached | [{"id": "2fff50ab..."}]        |
| metadata                             | {}                             |
+--------------------------------------+--------------------------------+
```

6. List volumes to see the bootable volume and its attached `myInstanceFromVolume` instance.

```
$ openstack volume list
+--------------------+-----------------+--------+------+-------------------------
↪--------+
| ID                 | Name            | Status | Size | Attached to             ␣
↪        |
+--------------------+-----------------+--------+------+-------------------------
↪--------+
| c612f739-8592-44c4- | bootable_volume | in-use |  10  | Attached to␣
↪myInstanceFromVolume|
| b7d4-0fee2fe1da0c   |                 |        |      | on /dev/vda             ␣
↪        |
+--------------------+-----------------+--------+------+-------------------------
↪--------+
```

Attach swap or ephemeral disk to an instance

Use the `nova boot --swap` parameter to attach a swap disk on boot or the `nova boot --ephemeral` parameter to attach an ephemeral disk on boot. When you terminate the instance, both disks are deleted.

Boot an instance with a 512 MB swap disk and 2 GB ephemeral disk.

```
$ nova boot --flavor FLAVOR --image IMAGE_ID --swap 512 \
  --ephemeral size=2 NAME
```

Note: The flavor defines the maximum swap and ephemeral disk size. You cannot exceed these maximum values.

Launch an instance using ISO image

Boot an instance from an ISO image

OpenStack supports booting instances using ISO images. But before you make such instances functional, use the **`openstack server create`** command with the following parameters to boot an instance:

```
$ openstack server create --image ubuntu-14.04.2-server-amd64.iso \
  --nic net-id = NETWORK_UUID \
  --flavor 2 INSTANCE_NAME
+--------------------------------------+--------------------------------------------+
| Field                                | Value                                      |
+--------------------------------------+--------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                     |
| OS-EXT-AZ:availability_zone          | nova                                       |
| OS-EXT-SRV-ATTR:host                 | -                                          |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | -                                          |
| OS-EXT-SRV-ATTR:instance_name        | instance-00000004                          |
| OS-EXT-STS:power_state               | 0                                          |
| OS-EXT-STS:task_state                | scheduling                                 |
| OS-EXT-STS:vm_state                  | building                                   |
| OS-SRV-USG:launched_at               | -                                          |
| OS-SRV-USG:terminated_at             | -                                          |
| accessIPv4                           |                                            |
| accessIPv6                           |                                            |
| adminPass                            | ZaiYeC8iucgU                               |
| config_drive                         |                                            |
| created                              | 2015-06-01T16:34:50Z                       |
| flavor                               | m1.small (2)                               |
| hostId                               |                                            |
| id                                   | 1e1797f3-1662-49ff-ae8c-a77e82ee1571       |
| image                                | ubuntu-14.04.2-server-amd64.iso            |
| key_name                             | -                                          |
| metadata                             | {}                                         |
| name                                 | INSTANCE_NAME                              |
| os-extended-volumes:volumes_attached | []                                         |
| progress                             | 0                                          |
| security_groups                      | default                                    |
| status                               | BUILD                                      |
| tenant_id                            | ccef9e62b1e645df98728fb2b3076f27           |
| updated                              | 2014-05-09T16:34:51Z                       |
| user_id                              | fef060ae7bfd4024b3edb97dff59017a           |
+--------------------------------------+--------------------------------------------+
```

In this command, `ubuntu-14.04.2-server-amd64.iso` is the ISO image, and `INSTANCE_NAME` is the name of the new instance. `NETWORK_UUID` is a valid network id in your system.

Create a bootable volume for the instance to reside on after shutdown.

1. Create the volume:

```
$ openstack volume create \
  --size <SIZE_IN_GB> \
  --bootable VOLUME_NAME
```

2. Attach the instance to the volume:

```
$ openstack server add volume
  INSTANCE_NAME \
  VOLUME_NAME \
  --device /dev/vda
```

Note: You need the Block Storage service to preserve the instance after shutdown. The `--block-device` argument, used with the legacy **`nova boot`**, will not work with the OpenStack **`openstack server create`** command. Instead, the **`openstack volume create`** and **`openstack server add volume`** commands create persistent storage.

After the instance is successfully launched, connect to the instance using a remote console and follow the instructions to install the system as using ISO images on regular computers. When the installation is finished and system is rebooted, the instance asks you again to install the operating system, which means your instance is not usable. If you have problems with image creation, please check the Virtual Machine Image Guide for reference.

Make the instances booted from ISO image functional

Now complete the following steps to make your instances created using ISO image actually functional.

1. Delete the instance using the following command.

```
$ openstack server delete INSTANCE_NAME
```

2. After you delete the instance, the system you have just installed using your ISO image remains, because the parameter `shutdown=preserve` was set, so run the following command.

```
$ openstack volume list
+--------------------------+--------------------------+-----------+------+---------
↪----+
| ID                       | Name                     | Status    | Size |␣
↪Attached to |
+--------------------------+--------------------------+-----------+------+---------
↪----+
| 8edd7c97-1276-47a5-9563- |dc01d873-d0f1-40b6-bfcc- | available |   10 |        ␣
↪    |
| 1025f4264e4f             | 26a8d955a1d9-blank-vol  |           |      |        ␣
↪    |
+--------------------------+--------------------------+-----------+------+---------
↪----+
```

You get a list with all the volumes in your system. In this list, you can find the volume that is attached to your ISO created instance, with the false bootable property.

3. Upload the volume to glance.

```
$ openstack image create --volume SOURCE_VOLUME IMAGE_NAME
$ openstack image list
+--------------------+------------+--------+
| ID                 | Name       | Status |
+--------------------+------------+--------+
| 74303284-f802-... | IMAGE_NAME | active |
+--------------------+------------+--------+
```

The `SOURCE_VOLUME` is the UUID or a name of the volume that is attached to your ISO created instance, and the `IMAGE_NAME` is the name that you give to your new image.

4. After the image is successfully uploaded, you can use the new image to boot instances.

   The instances launched using this image contain the system that you have just installed using the ISO image.

Metadata

Nova presents configuration information to instances it starts via a mechanism called metadata. These mechanisms are widely used via helpers such as cloud-init to specify things like the root password the instance should use.

This metadata is made available via either a *config drive* or the *metadata service* and can be somewhat customised by the user using the *user data* feature. This guide provides an overview of these features along with a summary of the types of metadata available.

Types of metadata

There are three separate groups of users who need to be able to specify metadata for an instance.

User provided data

The user who booted the instance can pass metadata to the instance in several ways. For authentication keypairs, the keypairs functionality of the nova API can be used to upload a key and then specify that key during the nova boot API request. For less structured data, a small opaque blob of data may be passed via the *user data* feature of the nova API. Examples of such unstructured data would be the puppet role that the instance should use, or the HTTP address of a server from which to fetch post-boot configuration information.

Nova provided data

Nova itself needs to pass information to the instance via its internal implementation of the metadata system. Such information includes the requested hostname for the instance and the availability zone the instance is in. This happens by default and requires no configuration by the user or deployer.

Nova provides both an *OpenStack metadata API* and an *EC2-compatible API*. Both the OpenStack metadata and EC2-compatible APIs are versioned by date. These are described later.

Deployer provided data

A deployer of OpenStack may need to pass data to an instance. It is also possible that this data is not known to the user starting the instance. An example might be a cryptographic token to be used to register the instance with Active Directory post boot – the user starting the instance should not have access to Active Directory to create this token, but the nova deployment might have permissions to generate the token on the users behalf. This is possible using the *vendordata* feature, which must be configured by your cloud operator.

The metadata service

Note: This section provides end user information about the metadata service. For deployment information about the metadata service, refer to the *admin guide*.

The *metadata service* provides a way for instances to retrieve instance-specific data via a REST API. Instances access this service at `169.254.169.254` and all types of metadata, be it user-, nova- or vendor-provided, can be accessed via this service.

Using the metadata service

To retrieve a list of supported versions for the *OpenStack metadata API*, make a GET request to `http://169.254.169.254/openstack`, which will return a list of directories:

```
$ curl http://169.254.169.254/openstack
2012-08-10
2013-04-04
2013-10-17
2015-10-15
2016-06-30
2016-10-06
2017-02-22
2018-08-27
latest
```

Refer to *OpenStack format metadata* for information on the contents and structure of these directories.

To list supported versions for the *EC2-compatible metadata API*, make a GET request to `http://169.254.169.254`, which will, once again, return a list of directories:

```
$ curl http://169.254.169.254
1.0
2007-01-19
2007-03-01
2007-08-29
2007-10-10
2007-12-15
2008-02-01
2008-09-01
2009-04-04
latest
```

Refer to *EC2-compatible metadata* for information on the contents and structure of these directories.

Config drives

Note: This section provides end user information about config drives. For deployment information about the config drive feature, refer to the *admin guide*.

Config drives are special drives that are attached to an instance when it boots. The instance can mount this drive and read files from it to get information that is normally available through the metadata service.

One use case for using the config drive is to pass a networking configuration when you do not use DHCP to assign IP addresses to instances. For example, you might pass the IP address configuration for the instance through the config drive, which the instance can mount and access before you configure the network settings for the instance.

Using the config drive

To enable the config drive for an instance, pass the `--config-drive true` parameter to the **`openstack server create`** command.

The following example enables the config drive and passes a user data file and two key/value metadata pairs, all of which are accessible from the config drive:

```
$ openstack server create --config-drive true --image my-image-name \
    --flavor 1 --key-name mykey --user-data ./my-user-data.txt \
    --property role=webservers --property essential=false MYINSTANCE
```

Note: The Compute service can be configured to always create a config drive. For more information, refer to *the admin guide*.

If your guest operating system supports accessing disk by label, you can mount the config drive as the `/dev/disk/by-label/configurationDriveVolumeLabel` device. In the following example, the config drive has the `config-2` volume label:

```
# mkdir -p /mnt/config
# mount /dev/disk/by-label/config-2 /mnt/config
```

If your guest operating system does not use `udev`, the `/dev/disk/by-label` directory is not present. You can use the **`blkid`** command to identify the block device that corresponds to the config drive. For example:

```
# blkid -t LABEL="config-2" -odevice
/dev/vdb
```

Once identified, you can mount the device:

```
# mkdir -p /mnt/config
# mount /dev/vdb /mnt/config
```

Once mounted, you can examine the contents of the config drive:

```
$ cd /mnt/config
$ find . -maxdepth 2
.
./ec2
./ec2/2009-04-04
./ec2/latest
./openstack
./openstack/2012-08-10
./openstack/2013-04-04
./openstack/2013-10-17
./openstack/2015-10-15
./openstack/2016-06-30
./openstack/2016-10-06
./openstack/2017-02-22
./openstack/latest
```

The files that appear on the config drive depend on the arguments that you pass to the **openstack server create** command. The format of this directory is the same as that provided by the *metadata service*, with the exception that the EC2-compatible metadata is now located in the `ec2` directory instead of the root (`/`) directory. Refer to the *OpenStack format metadata* and *EC2-compatible metadata* sections for information about the format of the files and subdirectories within these directories.

Nova metadata

As noted previously, nova provides its metadata in two formats: OpenStack format and EC2-compatible format.

OpenStack format metadata

Changed in version 12.0.0: Support for network metadata was added in the Liberty release.

Metadata from the OpenStack API is distributed in JSON format. There are two files provided for each version: `meta_data.json` and `network_data.json`. The `meta_data.json` file contains nova-specific information, while the `network_data.json` file contains information retrieved from neutron. For example:

```
$ curl http://169.254.169.254/openstack/2018-08-27/meta_data.json
```

```
{
   "random_seed": "yu5ZnkqF2CqnDZVAfZgarGLoFubhcK5wHG4fcNfVZEtie/bTV8k2dDXK\
                   C7krP2cjp9A7g9LIWe5+WSaZ3zpvQ03hp/4mMNy9V1U/mnRMZyQ3W4Fn\
                   Nex7UP/0Smjb9rVzfUb2HrVUCN61Yo4jHySTd7UeEasF0nxBrx6NFY6e\
                   KRoELGPPr1S6+ZDcDT1Sp7pRoHqwVbzyJZc80ICndqxGkZOuvwDgVKZD\
                   B6O3kFSLuqOfNRaL8y79gJizw/MHI7YjOxtPMr6g0upIBHFl8Vt1VKjR\
                   s3zB+c3WkC6JsopjcToHeR4tPK0RtdIp6G2Bbls5cblQUAc/zG0a8BAm\
                   p6Pream9XRpaQBDk4iXtjIn8Bf56SCANOFfeI5BgBeTwfdDGoM0Ptml6\
                   BJQiyFtc3APfXVVswrCq2SuJop+spgrpiKXOzXvve+gEWVhyfbigI52e\
                   l1VyMoyZ7/pbdnX0LCGHOdAU8KRnBoo99ZOErv+p7sROEIN4Yywq/U/C\
                   xXtQ5BNCtae389+3yT5ZCV7fYzLYChgDMJSZ9ds9fDFIWKmsRu3N+wUg\
                   eL4klxAjRgzQ7MMlap5kppnIYRxXVy0a5j1qOaBAzJB5LLJ7r3/Om38x\
                   Z4+XGWjqd6KbSwhUVs1aqzxpep1Sp3nTurQCuYjgMchjslt0O5oJjh5Z\
                   hbCZT3YUc8M=\n",
   "uuid": "d8e02d56-2648-49a3-bf97-6be8f1204f38",
   "availability_zone": "nova",
   "keys": [
       {
         "data": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQDYVEprvtYJXVOBN0XNKV\
                  VRNCRX6BlnNbI+USLGais1sUWPwtSg7z9K9vhbYAPUZcq8c/s5S9dg5vTH\
                  bsiyPCIDOKyeHba4MUJq8Oh5b2i71/3BISpyxTBH/uZDHdslW2a+SrPDCe\
                  uMMoss9NFhBdKtDkdG9zyi0ibmCP6yMdEX8Q== Generated by Nova\n",
         "type": "ssh",
         "name": "mykey"
       }
   ],
   "hostname": "test.novalocal",
   "launch_index": 0,
   "meta": {
      "priority": "low",
      "role": "webserver"
   },
   "devices": [
       {
         "type": "nic",
```

(continues on next page)

(continued from previous page)

```
          "bus": "pci",
          "address": "0000:00:02.0",
          "mac": "00:11:22:33:44:55",
          "tags": ["trusted"]
        },
        {
          "type": "disk",
          "bus": "ide",
          "address": "0:0",
          "serial": "disk-vol-2352423",
          "path": "/dev/sda",
          "tags": ["baz"]
        }
    ],
    "project_id": "f7ac731cc11f40efbc03a9f9e1d1d21f",
    "public_keys": {
        "mykey": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQDYVEprvtYJXVOBN0XNKV\
                  VRNCRX6BlnNbI+USLGais1sUWPwtSg7z9K9vhbYAPUZcq8c/s5S9dg5vTH\
                  bsiyPCIDOKyeHba4MUJq8Oh5b2i71/3BISpyxTBH/uZDHdslW2a+SrPDCe\
                  uMMoss9NFhBdKtDkdG9zyi0ibmCP6yMdEX8Q== Generated by Nova\n"
    },
    "name": "test"
}
```

```
$ curl http://169.254.169.254/openstack/2018-08-27/network_data.json
```

```
{
    "links": [
        {
            "ethernet_mac_address": "fa:16:3e:9c:bf:3d",
            "id": "tapcd9f6d46-4a",
            "mtu": null,
            "type": "bridge",
            "vif_id": "cd9f6d46-4a3a-43ab-a466-994af9db96fc"
        }
    ],
    "networks": [
        {
            "id": "network0",
            "link": "tapcd9f6d46-4a",
            "network_id": "99e88329-f20d-4741-9593-25bf07847b16",
            "type": "ipv4_dhcp"
        }
    ],
    "services": [
        {
            "address": "8.8.8.8",
            "type": "dns"
        }
    ]
}
```

EC2-compatible metadata

The EC2-compatible API is compatible with version 2009-04-04 of the Amazon EC2 metadata service This means that virtual machine images designed for EC2 will work properly with OpenStack.

The EC2 API exposes a separate URL for each metadata element. Retrieve a listing of these elements by making a GET query to `http://169.254.169.254/2009-04-04/meta-data/`. For example:

```
$ curl http://169.254.169.254/2009-04-04/meta-data/
ami-id
ami-launch-index
ami-manifest-path
block-device-mapping/
hostname
instance-action
instance-id
instance-type
kernel-id
local-hostname
local-ipv4
placement/
public-hostname
public-ipv4
public-keys/
ramdisk-id
reservation-id
security-groups
```

```
$ curl http://169.254.169.254/2009-04-04/meta-data/block-device-mapping/
ami
```

```
$ curl http://169.254.169.254/2009-04-04/meta-data/placement/
availability-zone
```

```
$ curl http://169.254.169.254/2009-04-04/meta-data/public-keys/
0=mykey
```

Instances can retrieve the public SSH key (identified by keypair name when a user requests a new instance) by making a GET request to `http://169.254.169.254/2009-04-04/meta-data/public-keys/0/openssh-key`:

```
$ curl http://169.254.169.254/2009-04-04/meta-data/public-keys/0/openssh-key
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQDYVEprvtYJXVOBN0XNKVVRNCRX6BlnNbI+US\
LGais1sUWPwtSg7z9K9vhbYAPUZcq8c/s5S9dg5vTHbsiyPCIDOKyeHba4MUJq8Oh5b2i71/3B\
ISpyxTBH/uZDHdslW2a+SrPDCeuMMoss9NFhBdKtDkdG9zyi0ibmCP6yMdEX8Q== Generated\
by Nova
```

User data

User data is a blob of data that the user can specify when they launch an instance. The instance can access this data through the metadata service or config drive. Commonly used to pass a shell script that the instance runs on boot.

For example, one application that uses user data is the cloud-init system, which is an open-source package from Ubuntu that is available on various Linux distributions and which handles early initialization of a cloud instance.

You can place user data in a local file and pass it through the `--user-data <user-data-file>` parameter at instance creation.

```
$ openstack server create --image ubuntu-cloudimage --flavor 1 \
    --user-data mydata.file VM_INSTANCE
```

Note: The provided user data must be base64 encoded and is restricted to 65535 bytes.

Once booted, you can access this data from the instance using either the metadata service or the config drive. To access it via the metadata service, make a GET request to either `http://169.254.169.254/openstack/{version}/user_data` (OpenStack API) or `http://169.254.169.254/{version}/user-data` (EC2-compatible API). For example:

```
$ curl http://169.254.169.254/openstack/2018-08-27/user_data
```

```
#!/bin/bash
echo 'Extra user data here'
```

Vendordata

Note: This section provides end user information about the vendordata feature. For deployment information about this feature, refer to the *admin guide*.

Changed in version 14.0.0: Support for dynamic vendor data was added in the Newton release.

Where configured, instances can retrieve vendor-specific data from the metadata service or config drive. To access it via 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`, depending on the deployment. For example:

```
$ curl http://169.254.169.254/openstack/2018-08-27/vendor_data2.json
```

```
{
    "testing": {
        "value1": 1,
        "value2": 2,
        "value3": "three"
    }
}
```

Note: The presence and contents of this file will vary from deployment to deployment.

General guidelines

- Do not rely on the presence of the EC2 metadata in the metadata API or config drive, because this content might be removed in a future release. For example, do not rely on files in the `ec2` directory.
- When you create images that access metadata service or config drive data and multiple directories are under the `openstack` directory, always select the highest API version by date that your consumer supports. For

example, if your guest image supports the `2012-03-05`, `2012-08-05`, and `2013-04-13` versions, try `2013-04-13` first and fall back to a previous version if `2013-04-13` is not present.

Image Signature Certificate Validation

Nova can determine if the certificate used to generate and verify the signature of a signed image (see Glance Image Signature Verification documentation) is trusted by the user. This feature is called certificate validation and can be applied to the creation or rebuild of an instance.

Certificate validation is meant to be performed jointly with image signature verification but each feature has its own Nova configuration option, to be specified in the `[glance]` section of the `nova.conf` configuration file. To enable certificate validation, set *`glance.enable_certificate_validation`* to True. To enable signature validation, set *`glance.verify_glance_signatures`* to True. Conversely, to disable either of these features, set their option to False or do not include the option in the Nova configurations at all.

Certificate validation operates in concert with signature validation in Cursive. It takes in a list of trusted certificate IDs and verifies that the certificate used to sign the image being booted is cryptographically linked to at least one of the provided trusted certificates. This provides the user with confidence in the identity and integrity of the image being booted.

Certificate validation will only be performed if image signature validation is enabled. However, the presence of trusted certificate IDs overrides the `enable_certificate_validation` and `verify_glance_signatures` settings. In other words, if a list of trusted certificate IDs is provided to the instance create or rebuild commands, signature verification and certificate validation will be performed, regardless of their settings in the Nova configurations. See *Using Signature Verification* for details.

Note: Certificate validation configuration options must be specified in the Nova configuration file that controls the `nova-osapi_compute` and `nova-compute` services, as opposed to other Nova services (conductor, scheduler, etc.).

Requirements

Key manager that is a backend to the Castellan Interface. Possible key managers are:

- Barbican
- Vault

Limitations

- As of the 18.0.0 Rocky release, only the libvirt compute driver supports trusted image certification validation. The feature is not, however, driver specific so other drivers should be able to support this feature over time. See the feature support matrix for information on which drivers support the feature at any given release.
- As of the 18.0.0 Rocky release, trusted image certification validation is not supported with volume-backed (boot from volume) instances. The block storage service support may be available in a future release:

  https://blueprints.launchpad.net/cinder/+spec/certificate-validate

- Trusted image certification support can be controlled via policy configuration if it needs to be disabled. See the `os_compute_api:servers:create:trusted_certs` and `os_compute_api:servers:rebuild:trusted_certs` policy rules.

Configuration

Nova will use the key manager defined by the Castellan key manager interface, which is the Barbican key manager by default. To use a different key manager, update the `backend` value in the `[key_manager]` group of the nova configuration file. For example:

```
[key_manager]
backend = barbican
```

Note: If these lines do not exist, then simply add them to the end of the file.

Using Signature Verification

An image will need a few properties for signature verification to be enabled:

`img_signature` Signature of your image. Signature restrictions are:

- 255 character limit

`img_signature_hash_method` Method used to hash your signature. Possible hash methods are:

- SHA-224
- SHA-256
- SHA-384
- SHA-512

`img_signature_key_type` Key type used for your image. Possible key types are:

- RSA-PSS
- DSA
- ECC-CURVES
  - SECT571K1
  - SECT409K1
  - SECT571R1
  - SECT409R1
  - SECP521R1
  - SECP384R1

`img_signature_certificate_uuid` UUID of the certificate that you uploaded to the key manager. Possible certificate types are:

- X_509

Using Certificate Validation

Certificate validation is triggered by one of two ways:

1. The Nova configuration options `verify_glance_signatures` and `enable_certificate_validation` are both set to True:

```
[glance]
verify_glance_signatures = True
enable_certificate_validation = True
```

2. A list of trusted certificate IDs is provided by one of three ways:

Note: The command line support is pending changes https://review.opendev.org/#/c/500396/ and https://review.opendev.org/#/c/501926/ to python-novaclient and python-openstackclient, respectively.

Environment Variable Use the environment variable `OS_TRUSTED_IMAGE_CERTIFICATE_IDS` to define a comma-delimited list of trusted certificate IDs. For example:

```
$ export OS_TRUSTED_IMAGE_CERTIFICATE_IDS=79a6ad17-3298-4e55-8b3a-
↪1672dd93c40f,b20f5600-3c9d-4af5-8f37-3110df3533a0
```

Command-Line Flag If booting or rebuilding an instance using the **`nova`** commands, use the `--trusted-image-certificate-id` flag to define a single trusted certificate ID. The flag may be used multiple times to specify multiple trusted certificate IDs. For example:

```
$ nova boot myInstanceName \
    --flavor 1 \
    --image myImageId \
    --trusted-image-certificate-id 79a6ad17-3298-4e55-8b3a-1672dd93c40f \
    --trusted-image-certificate-id b20f5600-3c9d-4af5-8f37-3110df3533a0
```

If booting or rebuilding an instance using the **`openstack server`** commands, use the `--trusted-image-certificate-id` flag to define a single trusted certificate ID. The flag may be used multiple times to specify multiple trusted certificate IDs. For example:

```
$ openstack --os-compute-api-version=2.63 server create myInstanceName \
    --flavor 1 \
    --image myImageId \
    --nic net-id=fd25c0b2-b36b-45a8-82e4-ab52516289e5 \
    --trusted-image-certificate-id 79a6ad17-3298-4e55-8b3a-1672dd93c40f \
    --trusted-image-certificate-id b20f5600-3c9d-4af5-8f37-3110df3533a0
```

Nova Configuration Option Use the Nova configuration option *`glance.default_trusted_certificate_ids`* to define a comma-delimited list of trusted certificate IDs. This configuration value is only used if `verify_glance_signatures` and `enable_certificate_validation` options are set to True, and the trusted certificate IDs are not specified anywhere else. For example:

```
[glance]
default_trusted_certificate_ids=79a6ad17-3298-4e55-8b3a-1672dd93c40f,b20f5600-
↪3c9d-4af5-8f37-3110df3533a0
```

Example Usage

For these instructions, we will construct a 4-certificate chain to illustrate that it is possible to have a single trusted root certificate. We will upload all four certificates to Barbican. Then, we will sign an image and upload it to Glance, which will illustrate image signature verification. Finally, we will boot the signed image from Glance to show that certificate validation is enforced.

Enable certificate validation

Enable image signature verification and certificate validation by setting both of their Nova configuration options to True:

```
[glance]
verify_glance_signatures = True
enable_certificate_validation = True
```

Create a certificate chain

As mentioned above, we will construct a 4-certificate chain to illustrate that it is possible to have a single trusted root certificate. Before we begin to build our certificate chain, we must first create files for OpenSSL to use for indexing and serial number tracking:

```
$ touch index.txt
$ echo '01' > serial.txt
```

Create a certificate configuration file

For these instructions, we will create a single configuration file called `ca.conf`, which contains various sections that we can specify for use on the command-line during certificate requests and generation.

Note that this certificate will be able to sign other certificates because it is a certificate authority. Also note the root CAs unique common name (root). The intermediate certificates common names will be specified on the command-line when generating the corresponding certificate requests.

`ca.conf`:

```
[ req ]
prompt             = no
distinguished_name = dn-param
x509_extensions    = ca_cert_extensions

[ ca ]
default_ca = ca_default

[ dn-param ]
C  = US
CN = Root CA

[ ca_cert_extensions ]
keyUsage         = keyCertSign, digitalSignature
basicConstraints = CA:TRUE, pathlen:2

[ ca_default ]
new_certs_dir = .              # Location for new certs after signing
database      = ./index.txt    # Database index file
serial        = ./serial.txt   # The current serial number

default_days  = 1000
default_md    = sha256

policy        = signing_policy
```

(continues on next page)

(continued from previous page)

```
email_in_dn    = no

[ intermediate_cert_extensions ]
keyUsage          = keyCertSign, digitalSignature
basicConstraints = CA:TRUE, pathlen:1

[client_cert_extensions]
keyUsage          = keyCertSign, digitalSignature
basicConstraints = CA:FALSE

[ signing_policy ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName = optional
commonName              = supplied
emailAddress            = optional
```

Generate the certificate authority (CA) and corresponding private key

For these instructions, we will save the certificate as `cert_ca.pem` and the private key as `key_ca.pem`. This certificate will be a self-signed root certificate authority (CA) that can sign other CAs and non-CA certificates.

```
$ openssl req \
    -x509 \
    -nodes \
    -newkey rsa:1024 \
    -config ca.conf \
    -keyout key_ca.pem \
    -out cert_ca.pem

Generating a 1024 bit RSA private key
..........................++++++
...++++++
writing new private key to 'key_ca.pem'
-----
```

Create the first intermediate certificate

Create a certificate request for the first intermediate certificate. For these instructions, we will save the certificate request as `cert_intermeidate_a.csr` and the private key as `key_intermediate_a.pem`.

```
$ openssl req \
    -nodes \
    -newkey rsa:2048 \
    -subj '/CN=First Intermediate Certificate' \
    -keyout key_intermediate_a.pem \
    -out cert_intermediate_a.csr

Generating a 2048 bit RSA private key
..................................................................................
↪.....................+++
```

(continues on next page)

(continued from previous page)

```
.....+++
writing new private key to 'key_intermediate_a.pem'
-----
```

Generate the first intermediate certificate by signing its certificate request with the CA. For these instructions we will save the certificate as `cert_intermediate_a.pem`.

```
$ openssl ca \
    -config ca.conf \
    -extensions intermediate_cert_extensions \
    -cert cert_ca.pem \
    -keyfile key_ca.pem \
    -out cert_intermediate_a.pem \
    -infiles cert_intermediate_a.csr

Using configuration from ca.conf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'First Intermediate Certificate'
Certificate is to be certified until Nov 15 16:24:21 2020 GMT (1000 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Create the second intermediate certificate

Create a certificate request for the second intermediate certificate. For these instructions, we will save the certificate request as `cert_intermeidate_b.csr` and the private key as `key_intermediate_b.pem`.

```
$ openssl req \
    -nodes \
    -newkey rsa:2048 \
    -subj '/CN=Second Intermediate Certificate' \
    -keyout key_intermediate_b.pem \
    -out cert_intermediate_b.csr

Generating a 2048 bit RSA private key
..........+++
.........................................+++
writing new private key to 'key_intermediate_b.pem'
-----
```

Generate the second intermediate certificate by signing its certificate request with the first intermediate certificate. For these instructions we will save the certificate as `cert_intermediate_b.pem`.

```
$ openssl ca \
    -config ca.conf \
    -extensions intermediate_cert_extensions \
    -cert cert_intermediate_a.pem \
    -keyfile key_intermediate_a.pem \
```

(continues on next page)

(continued from previous page)

```
    -out cert_intermediate_b.pem \
    -infiles cert_intermediate_b.csr

Using configuration from ca.conf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'Second Intermediate Certificate'
Certificate is to be certified until Nov 15 16:25:42 2020 GMT (1000 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Create the client certificate

Create a certificate request for the client certificate. For these instructions, we will save the certificate request as `cert_client.csr` and the private key as `key_client.pem`.

```
$ openssl req \
    -nodes \
    -newkey rsa:2048 \
    -subj '/CN=Client Certificate' \
    -keyout key_client.pem \
    -out cert_client.csr

Generating a 2048 bit RSA private key
.....................................................................................
↪.....................................+++
.....................................................................................
↪........+++
writing new private key to 'key_client.pem'
-----
```

Generate the client certificate by signing its certificate request with the second intermediate certificate. For these instructions we will save the certificate as `cert_client.pem`.

```
$ openssl ca \
    -config ca.conf \
    -extensions client_cert_extensions \
    -cert cert_intermediate_b.pem \
    -keyfile key_intermediate_b.pem \
    -out cert_client.pem \
    -infiles cert_client.csr

Using configuration from ca.conf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'Client Certificate'
Certificate is to be certified until Nov 15 16:26:46 2020 GMT (1000 days)
Sign the certificate? [y/n]:y
```

(continues on next page)

(continued from previous page)

```
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Upload the generated certificates to the key manager

In order interact with the key manager, the user needs to have a *creator* role.

To list all users with a *creator* role, run the following command as an admin:

```
$ openstack role assignment list --role creator --names

+---------+-----------------------------+-------+------------------+--------+--------
↪---+
| Role    | User                        | Group | Project          | Domain |␣
↪Inherited |
+---------+-----------------------------+-------+------------------+--------+--------
↪---+
| creator | project_a_creator_2@Default |       | project_a@Default |        | False ␣
↪   |
| creator | project_b_creator@Default   |       | project_b@Default |        | False ␣
↪   |
| creator | project_a_creator@Default   |       | project_a@Default |        | False ␣
↪   |
+---------+-----------------------------+-------+------------------+--------+--------
↪---+
```

To give the *demo* user a *creator* role in the *demo* project, run the following command as an admin:

```
$ openstack role add --user demo --project demo creator
```

Note: This command provides no output. If the command fails, the user will see a 4xx Client error indicating that Secret creation attempt not allowed and to please review your user/project privileges.

Note: The following openstack secret commands require that the python-barbicanclient package is installed.

```
$ openstack secret store \
    --name CA \
    --algorithm RSA \
    --expiration 2018-06-29 \
    --secret-type certificate \
    --payload-content-type "application/octet-stream" \
    --payload-content-encoding base64 \
    --payload "$(base64 cert_ca.pem)"

$ openstack secret store \
    --name IntermediateA \
    --algorithm RSA \
    --expiration 2018-06-29 \
    --secret-type certificate \
```

(continues on next page)

(continued from previous page)

```
    --payload-content-type "application/octet-stream" \
    --payload-content-encoding base64 \
    --payload "$(base64 cert_intermediate_a.pem)"

$ openstack secret store \
    --name IntermediateB \
    --algorithm RSA \
    --expiration 2018-06-29 \
    --secret-type certificate \
    --payload-content-type "application/octet-stream" \
    --payload-content-encoding base64 \
    --payload "$(base64 cert_intermediate_b.pem)"

$ openstack secret store \
    --name Client \
    --algorithm RSA \
    --expiration 2018-06-29 \
    --secret-type certificate \
    --payload-content-type "application/octet-stream" \
    --payload-content-encoding base64 \
    --payload "$(base64 cert_client.pem)"
```

The responses should look something like this:

```
+---------------+------------------------------------------------------------------------
↪---------+
| Field         | Value                                                                  ␣
↪         |
+---------------+------------------------------------------------------------------------
↪---------+
| Secret href   | http://127.0.0.1/key-manager/v1/secrets/8fbcce5d-d646-4295-ba8a-
↪269fc9451eeb |
| Name          | CA                                                                     ␣
↪         |
| Created       | None                                                                   ␣
↪         |
| Status        | None                                                                   ␣
↪         |
| Content types | {u'default': u'application/octet-stream'}                              ␣
↪         |
| Algorithm     | RSA                                                                    ␣
↪         |
| Bit length    | 256                                                                    ␣
↪         |
| Secret type   | certificate                                                            ␣
↪         |
| Mode          | cbc                                                                    ␣
↪         |
| Expiration    | 2018-06-29T00:00:00+00:00                                              ␣
↪         |
+---------------+------------------------------------------------------------------------
↪---------+
```

Save off the certificate UUIDs (found in the secret href):

```
$ cert_ca_uuid=8fbcce5d-d646-4295-ba8a-269fc9451eeb
$ cert_intermediate_a_uuid=0b5d2c72-12cc-4ba6-a8d7-3ff5cc1d8cb8
```

(continues on next page)

(continued from previous page)

```
$ cert_intermediate_b_uuid=674736e3-f25c-405c-8362-bbf991e0ce0a
$ cert_client_uuid=125e6199-2de4-46e3-b091-8e2401ef0d63
```

Create a signed image

For these instructions, we will download a small CirrOS image:

```
$ wget -nc -O cirros.tar.gz http://download.cirros-cloud.net/0.3.5/cirros-0.3.5-
↪source.tar.gz

--2018-02-19 11:37:52--  http://download.cirros-cloud.net/0.3.5/cirros-0.3.5-source.
↪tar.gz
Resolving download.cirros-cloud.net (download.cirros-cloud.net)... 64.90.42.85
Connecting to download.cirros-cloud.net (download.cirros-cloud.net)|64.90.42.85|:80...
↪ connected.
HTTP request sent, awaiting response... 200 OK
Length: 434333 (424K) [application/x-tar]
Saving to: cirros.tar.gz

cirros.tar.gz       100%[===================>] 424.15K  --.-KB/s    in 0.1s

2018-02-19 11:37:54 (3.79 MB/s) - cirros.tar.gz saved [434333/434333]
```

Sign the image with the generated client private key:

```
$ openssl dgst \
    -sha256 \
    -sign key_client.pem \
    -sigopt rsa_padding_mode:pss \
    -out cirros.self_signed.signature \
    cirros.tar.gz
```

Note: This command provides no output.

Save off the base64 encoded signature:

```
$ base64_signature=$(base64 -w 0 cirros.self_signed.signature)
```

Upload the signed image to Glance:

```
$ openstack image create \
    --public \
    --container-format bare \
    --disk-format qcow2 \
    --property img_signature="$base64_signature" \
    --property img_signature_certificate_uuid="$cert_client_uuid" \
    --property img_signature_hash_method='SHA-256' \
    --property img_signature_key_type='RSA-PSS' \
    --file cirros.tar.gz \
    cirros_client_signedImage

+------------------+------------------------------------------------------------------
↪------+
```

(continues on next page)

(continued from previous page)

```
| Field            | Value                                                                    ␣
↪       |
+------------------+--------------------------------------------------------------------------
↪------+
| checksum         | d41d8cd98f00b204e9800998ecf8427e                                         ␣
↪       |
| container_format | bare                                                                     ␣
↪       |
| created_at       | 2019-02-06T06:29:56Z                                                     ␣
↪       |
| disk_format      | qcow2                                                                    ␣
↪       |
| file             | /v2/images/17f48a6c-e592-446e-9c91-00fbc436d47e/file                     ␣
↪       |
| id               | 17f48a6c-e592-446e-9c91-00fbc436d47e                                     ␣
↪       |
| min_disk         | 0                                                                        ␣
↪       |
| min_ram          | 0                                                                        ␣
↪       |
| name             | cirros_client_signedImage                                                ␣
↪       |
| owner            | 45e13e63606f40d6b23275c3cd91aec2                                         ␣
↪       |
| properties       | img_signature='swA/
↪hZi3WaNh35VMGlnfGnBWuXMlUbdO8h306uG7W3nwOyZP6dGRJ3   |
|                  | Xoi/
↪07Bo2dMUB9saFowqVhdlW5EywQAK6vgDsi9O5aItHM4u7zUPw+2e8eeaIoHlGhTks   |
|                  | kmW9isLy0mYA9nAfs3coChOIPXW4V8VgVXEfb6VYGHWm0nShiAP1e0do9WwitsE/
↪TVKoS   |
|                  | QnWjhggIYij5hmUZ628KAygPnXklxVhqPpY/dFzL+tTzNRD0nWAtsc5wrl6/
↪8HcNzZsaP   |
|                  |␣
↪oexAysXJtcFzDrf6UQu66D3UvFBVucRYL8S3W56It3Xqu0+InLGaXJJpNagVQBb476zB2   |
|                  | ZzZ5RJ/4Zyxw==',                                                         ␣
↪       |
|                  | img_signature_certificate_uuid='125e6199-2de4-46e3-b091-
↪8e2401ef0d63', |
|                  | img_signature_hash_method='SHA-256',                                     ␣
↪       |
|                  | img_signature_key_type='RSA-PSS',                                        ␣
↪       |
|                  | os_hash_algo='sha512',                                                   ␣
↪       |
|                  | os_hash_value=
↪'cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a92 |
|                  |␣
↪1d36ce9ce47d0d13c5d85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927d |
|                  | a3e',                                                                    ␣
↪       |
|                  | os_hidden='False'                                                        ␣
↪       |
| protected        | False                                                                    ␣
↪       |
| schema           | /v2/schemas/image                                                        ␣
↪       |
| size             | 0                                                                        ␣
↪       |
```

(continues on next page)

(continued from previous page)

```
| status           | active                                                                 ␣
↪       |
| tags             |                                                                        ␣
↪       |
| updated_at       | 2019-02-06T06:29:56Z                                                   ␣
↪       |
| virtual_size     | None                                                                   ␣
↪       |
| visibility       | public                                                                 ␣
↪       |
+------------------+------------------------------------------------------------------------
↪------+
```

Note: Creating the image can fail if validation does not succeed. This will cause the image to be deleted and the Glance log to report that Signature verification failed for the given image ID.

Boot the signed image

Boot the signed image without specifying trusted certificate IDs:

```
$ nova boot myInstance \
    --flavor m1.tiny \
    --image cirros_client_signedImage
```

Note: The instance should fail to boot because certificate validation fails when the feature is enabled but no trusted image certificates are provided. The Nova log output should indicate that Image signature certificate validation failed because Certificate chain building failed.

Boot the signed image with trusted certificate IDs:

```
$ nova boot myInstance \
    --flavor m1.tiny \
    --image cirros_client_signedImage \
    --trusted-image-certificate-id $cert_ca_uuid,$cert_intermediate_a_uuid \
    --trusted-image-certificate-id $cert_intermediate_b_uuid
```

Note: The instance should successfully boot and certificate validation should succeed. The Nova log output should indicate that Image signature certificate validation succeeded.

Other Links

- https://etherpad.openstack.org/p/mitaka-glance-image-signing-instructions
- https://etherpad.openstack.org/p/queens-nova-certificate-validation
- https://wiki.openstack.org/wiki/OpsGuide/User-Facing_Operations
- http://specs.openstack.org/openstack/nova-specs/specs/rocky/approved/nova-validate-certificates.html

Resize an instance

You can change the size of an instance by changing its flavor. This rebuilds the instance and therefore results in a restart.

To resize an instance, use the **`openstack server resize`** command:

```
$ openstack server resize --flavor FLAVOR SERVER
```

Note: By default, the **`openstack server resize`** command gives the guest operating system a chance to perform a controlled shutdown before the instance is powered off and the instance is resized. The shutdown behavior is configured by the *`shutdown_timeout`* parameter that can be set in the `nova.conf` file. Its value stands for the overall period (in seconds) a guest operating system is allowed to complete the shutdown. The default timeout is 60 seconds.

The timeout value can be overridden on a per image basis by means of `os_shutdown_timeout` that is an image metadata setting allowing different types of operating systems to specify how much time they need to shut down cleanly. See Useful image properties for details.

Resizing can take some time. During this time, the instance status will be `RESIZE`:

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

When the resize completes, the instance status will be `VERIFY_RESIZE`. You can now confirm the resize to change the status to `ACTIVE`:

```
$ openstack server resize --confirm SERVER
```

Note: The resized server may be automatically confirmed based on the administrators configuration of the deployment.

If the resize fails or does not work as expected, you can revert the resize. This will revert the instance to the old flavor and change the status to `ACTIVE`:

```
$ openstack server resize --revert SERVER
```

Reboot an instance

You can soft or hard reboot a running instance. A soft reboot attempts a graceful shut down and restart of the instance. A hard reboot power cycles the instance.

To reboot a server, use the **`openstack server reboot`** command:

```
$ openstack server reboot SERVER
```

By default, when you reboot an instance it is a soft reboot. To perform a hard reboot, pass the `--hard` parameter as follows:

```
$ openstack server reboot --hard SERVER
```

It is also possible to reboot a running instance into rescue mode. For example, this operation may be required if a filesystem of an instance becomes corrupted with prolonged use.

Note: Pause, suspend, and stop operations are not allowed when an instance is running in rescue mode, as triggering these actions causes the loss of the original instance state and makes it impossible to unrescue the instance.

Rescue mode provides a mechanism for access, even if an image renders the instance inaccessible. By default, it starts an instance from the initial image attaching the current boot disk as a secondary one.

To perform an instance reboot into rescue mode, use the **`openstack server rescue`** command:

```
$ openstack server rescue SERVER
```

Note: On running the **`openstack server rescue`** command, an instance performs a soft shutdown first. This means that the guest operating system has a chance to perform a controlled shutdown before the instance is powered off. The shutdown behavior is configured by the *`shutdown_timeout`* parameter that can be set in the `nova.conf` file. Its value stands for the overall period (in seconds) a guest operating system is allowed to complete the shutdown.

The timeout value can be overridden on a per image basis by means of `os_shutdown_timeout` that is an image metadata setting allowing different types of operating systems to specify how much time they need to shut down cleanly.

To restart the instance from the normal boot disk, run the following command:

```
$ openstack server unrescue SERVER
```

If you want to rescue an instance with a specific image, rather than the default one, use the `--image` parameter:

```
$ openstack server rescue --image IMAGE_ID SERVER
```

Todo: The rest of this document should probably move to the admin guide.

4.45.2 Architecture Overview

- *Nova architecture*: An overview of how all the parts in nova fit together.
- *Block Device Mapping*: One of the more complicated parts to understand is the Block Device Mapping parameters used to connect specific block devices to computes. This deserves its own deep dive.

See the *reference guide* for details about more internal subsystems.

4.45.3 Deployment Considerations

There is information you might want to consider before doing your deployment, especially if it is going to be a larger deployment. For smaller deployments the defaults from the *install guide* will be sufficient.

- **Compute Driver Features Supported**: While the majority of nova deployments use libvirt/kvm, you can use nova with other compute drivers. Nova attempts to provide a unified feature set across these, however, not all features are implemented on all backends, and not all features are equally well tested.
  - *Feature Support by Use Case*: A view of what features each driver supports based on whats important to some large use cases (General Purpose Cloud, NFV Cloud, HPC Cloud).
  - *Feature Support full list*: A detailed dive through features in each compute driver backend.
- *Cells v2 Planning*: For large deployments, Cells v2 allows sharding of your compute environment. Upfront planning is key to a successful Cells v2 layout.
- Placement service: Overview of the placement service, including how it fits in with the rest of nova.
- *Running nova-api on wsgi*: Considerations for using a real WSGI container instead of the baked-in eventlet web server.

4.45.4 Maintenance

Once you are running nova, the following information is extremely useful.

- *Admin Guide*: A collection of guides for administrating nova.
- *Upgrades*: How nova is designed to be upgraded for minimal service impact, and the order you should do them in.
- *Quotas*: Managing project quotas in nova.
- *Aggregates*: Aggregates are a useful way of grouping hosts together for scheduling purposes.
- *Filter Scheduler*: How the filter scheduler is configured, and how that will impact where compute instances land in your environment. If you are seeing unexpected distribution of compute instances in your hosts, youll want to dive into this configuration.
- *Exposing custom metadata to compute instances*: How and when you might want to extend the basic metadata exposed to compute instances (either via metadata server or config drive) for your specific purposes.

4.46 Host Aggregates

Host aggregates can be regarded as a mechanism to further partition an availability zone; while availability zones are visible to users, host aggregates are only visible to administrators. Host aggregates started out as a way to use Xen hypervisor resource pools, but have been generalized to provide a mechanism to allow administrators to assign key-value pairs to groups of machines. Each node can have multiple aggregates, each aggregate can have multiple key-value pairs, and the same key-value pair can be assigned to multiple aggregates. This information can be used in the scheduler to enable advanced scheduling, to set up Xen hypervisor resource pools or to define logical groups for migration. For more information, including an example of associating a group of hosts to a flavor, see *Host aggregates and availability zones*.

4.46.1 Availability Zones (AZs)

Availability Zones are the end-user visible logical abstraction for partitioning a cloud without knowing the physical infrastructure. That abstraction doesnt come up in Nova with an actual database model since the availability zone is

actually a specific metadata information attached to an aggregate. Adding that specific metadata to an aggregate makes the aggregate visible from an end-user perspective and consequently allows to schedule upon a specific set of hosts (the ones belonging to the aggregate).

That said, there are a few rules to know that diverge from an API perspective between aggregates and availability zones:

- one host can be in multiple aggregates, but it can only be in one availability zone
- by default a host is part of a default availability zone even if it doesnt belong to an aggregate (the configuration option is named `default_availability_zone`)

Warning: That last rule can be very error-prone. Since the user can see the list of availability zones, they have no way to know whether the default availability zone name (currently *nova*) is provided because an host belongs to an aggregate whose AZ metadata key is set to *nova*, or because there is at least one host not belonging to any aggregate. Consequently, it is highly recommended for users to never ever ask for booting an instance by specifying an explicit AZ named *nova* and for operators to never set the AZ metadata for an aggregate to *nova*. That leads to some problems due to the fact that the instance AZ information is explicitly attached to *nova* which could break further move operations when either the host is moved to another aggregate or when the user would like to migrate the instance.

Note: Availability zone name must NOT contain : since it is used by admin users to specify hosts where instances are launched in server creation. See *Select hosts where instances are launched* for more detail.

There is a nice educational video about availability zones from the Rocky summit which can be found here: https://www.openstack.org/videos/vancouver-2018/curse-your-bones-availability-zones-1

Implications for moving servers

There are several ways to move a server to another host: evacuate, resize, cold migrate, live migrate, and unshelve. Move operations typically go through the scheduler to pick the target host *unless* a target host is specified and the request forces the server to that host by bypassing the scheduler. Only evacuate and live migrate can forcefully bypass the scheduler and move a server to a specified host and even then it is highly recommended to *not* force and bypass the scheduler.

With respect to availability zones, a server is restricted to a zone if:

1. The server was created in a specific zone with the `POST /servers` request containing the `availability_zone` parameter.
2. If the server create request did not contain the `availability_zone` parameter but the API service is configured for `default_schedule_zone` then by default the server will be scheduled to that zone.
3. The shelved offloaded server was unshelved by specifying the `availability_zone` with the `POST /servers/{server_id}/action` request using microversion 2.77 or greater.

If the server was not created in a specific zone then it is free to be moved to other zones, i.e. the *AvailabilityZoneFilter* is a no-op.

Knowing this, it is dangerous to force a server to another host with evacuate or live migrate if the server is restricted to a zone and is then forced to move to a host in another zone, because that will create an inconsistency in the internal tracking of where that server should live and may require manually updating the database for that server. For example, if a user creates a server in zone A and then the admin force live migrates the server to zone B, and then the user resizes the server, the scheduler will try to move it back to zone A which may or may not work, e.g. if the admin deleted or renamed zone A in the interim.

Resource affinity

The *cinder.cross_az_attach* configuration option can be used to restrict servers and the volumes attached to servers to the same availability zone.

A typical use case for setting `cross_az_attach=False` is to enforce compute and block storage affinity, for example in a High Performance Compute cluster.

By default `cross_az_attach` is True meaning that the volumes attached to a server can be in a different availability zone than the server. If set to False, then when creating a server with pre-existing volumes or attaching a volume to a server, the server and volume zone must match otherwise the request will fail. In addition, if the nova-compute service creates the volumes to attach to the server during server create, it will request that those volumes are created in the same availability zone as the server, which must exist in the block storage (cinder) service.

As noted in the *Implications for moving servers* section, forcefully moving a server to another zone could also break affinity with attached volumes.

Note: `cross_az_attach=False` is not widely used nor tested extensively and thus suffers from some known issues:

- Bug 1694844
- Bug 1781421

4.46.2 Design

The OSAPI Admin API is extended to support the following operations:

- Aggregates
  - list aggregates: returns a list of all the host-aggregates
  - create aggregate: creates an aggregate, takes a friendly name, etc. returns an id
  - show aggregate: shows the details of an aggregate (id, name, availability_zone, hosts and metadata)
  - update aggregate: updates the name and availability zone of an aggregate
  - set metadata: sets the metadata on an aggregate to the values supplied
  - delete aggregate: deletes an aggregate, it fails if the aggregate is not empty
  - add host: adds a host to the aggregate
  - remove host: removes a host from the aggregate
- Hosts
  - list all hosts by service
    - It has been deprecated since microversion 2.43. Use *list hypervisors* instead.
  - start host maintenance (or evacuate-host): disallow a host to serve API requests and migrate instances to other hosts of the aggregate
    - It has been deprecated since microversion 2.43. Use *disable service* instead.
  - stop host maintenance (or rebalance-host): put the host back into operational mode, migrating instances back onto that host
    - It has been deprecated since microversion 2.43. Use *enable service* instead.

- Hypervisors
  - list hypervisors: list hypervisors with hypervisor hostname
- Compute services
  - enable service
  - disable service

4.46.3 Using the Nova CLI

Using the nova command you can create, delete and manage aggregates. The following section outlines the list of available commands.

Usage

```
* aggregate-list                                                    Print a list of␣
↪all aggregates.
* aggregate-create         <name> [<availability_zone>]             Create a new␣
↪aggregate with the specified details.
* aggregate-delete         <aggregate>                              Delete the␣
↪aggregate by its ID or name.
* aggregate-show           <aggregate>                              Show details of␣
↪the aggregate specified by its ID or name.
* aggregate-add-host       <aggregate> <host>                       Add the host to␣
↪the aggregate specified by its ID or name.
* aggregate-remove-host    <aggregate> <host>                       Remove the␣
↪specified host from the aggregate specified by its ID or name.
* aggregate-set-metadata   <aggregate> <key=value> [<key=value> ...]
                                                                    Update the␣
↪metadata associated with the aggregate specified by its ID or name.
* aggregate-update         [--name <name>] [--availability-zone <availability-zone>]
↪<aggregate>
                                                                    Update the␣
↪aggregate's name or availability zone.

* host-list                                                         List all hosts by␣
↪service.
* hypervisor-list          [--matching <hostname>] [--marker <marker>] [--limit
↪<limit>]
                                                                    List hypervisors.

* host-update              [--status <enable|disable>] [--maintenance <enable|disable>
↪] <hostname>
                                                                    Put/resume host␣
↪into/from maintenance.
* service-enable           <id>                                     Enable the␣
↪service.
* service-disable          [--reason <reason>] <id>                 Disable the␣
↪service.
```

4.47 Nova System Architecture

Nova comprises multiple server processes, each performing different functions. The user-facing interface is a REST API, while internally Nova components communicate via an RPC message passing mechanism.

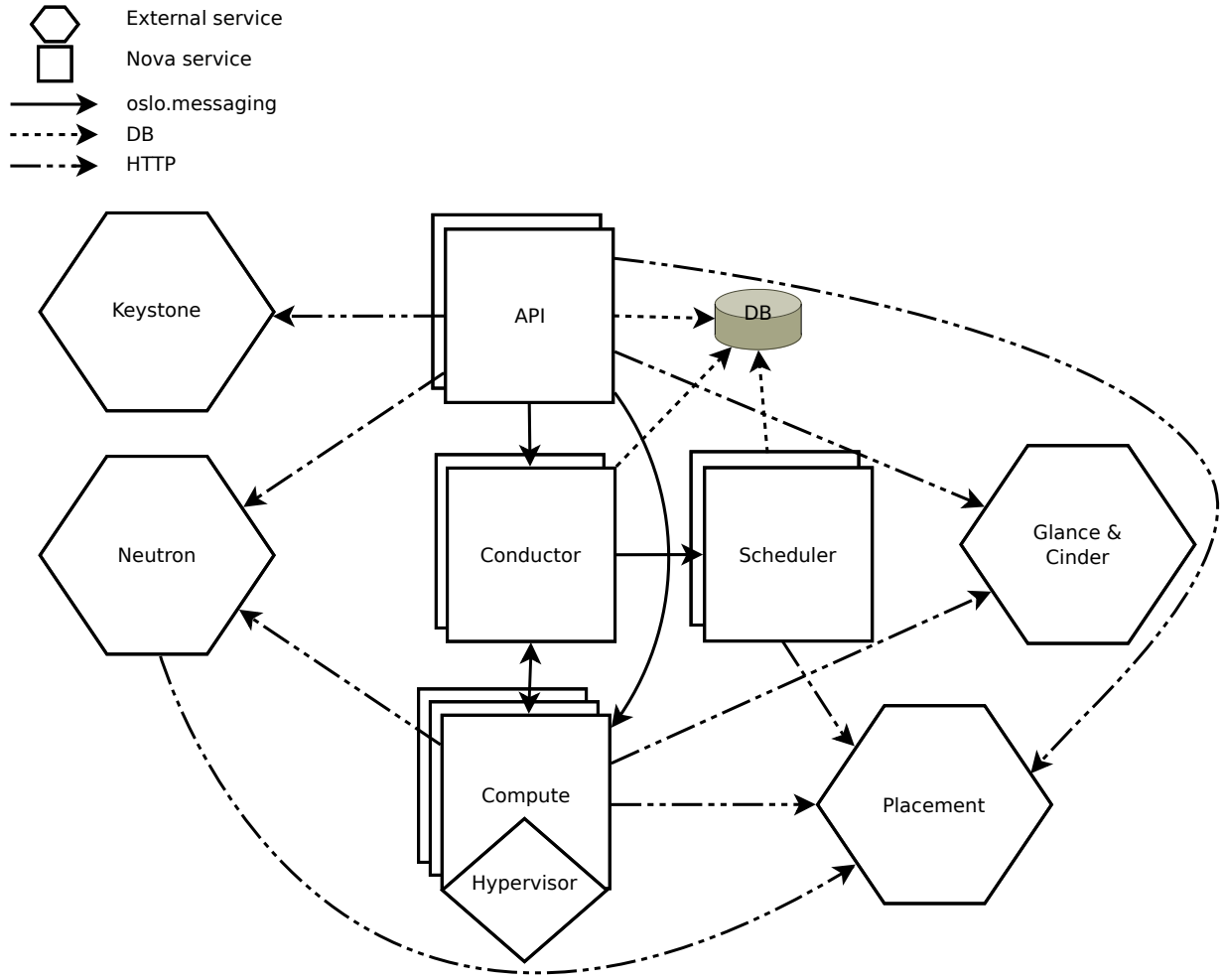
The API servers process REST requests, which typically involve database reads/writes, optionally sending RPC messages to other Nova services, and generating responses to the REST calls. RPC messaging is done via the **oslo.messaging** library, an abstraction on top of message queues. Most of the major nova components can be run on multiple servers, and have a manager that is listening for RPC messages. The one major exception is `nova-compute`, where a single process runs on the hypervisor it is managing (except when using the VMware or Ironic drivers). The manager also, optionally, has periodic tasks. For more details on our RPC system, please see: *AMQP and Nova*

Nova also uses a central database that is (logically) shared between all components. However, to aid upgrade, the DB is accessed through an object layer that ensures an upgraded control plane can still communicate with a `nova-compute` running the previous release. To make this possible nova-compute proxies DB requests over RPC to a central manager called `nova-conductor`.

To horizontally expand Nova deployments, we have a deployment sharding concept called cells. For more information please see: *Cells*

4.47.1 Components

Below you will find a helpful explanation of the key components of a typical Nova deployment.



- DB: sql database for data storage.
- API: component that receives HTTP requests, converts commands and communicates with other components via the **oslo.messaging** queue or HTTP.
- Scheduler: decides which host gets each instance.
- Compute: manages communication with hypervisor and virtual machines.
- Conductor: handles requests that need coordination (build/resize), acts as a database proxy, or handles object conversions.
- Placement: tracks resource provider inventories and usages.

While all services are designed to be horizontally scalable, you should have significantly more computes than anything else.

4.48 Block Device Mapping in Nova

Nova has a concept of block devices that can be exposed to cloud instances. There are several types of block devices an instance can have (we will go into more details about this later in this document), and which ones are available

depends on a particular deployment and the usage limitations set for tenants and users. Block device mapping is a way to organize and keep data about all of the block devices an instance has.

When we talk about block device mapping, we usually refer to one of two things

1. API/CLI structure and syntax for specifying block devices for an instance boot request
2. The data structure internal to Nova that is used for recording and keeping, which is ultimately persisted in the block_device_mapping table. However, Nova internally has several slightly different formats for representing the same data. All of them are documented in the code and or presented by a distinct set of classes, but not knowing that they exist might trip up people reading the code. So in addition to BlockDeviceMapping[1] objects that mirror the database schema, we have:

   2.1 The API format - this is the set of raw key-value pairs received from the API client, and is almost immediately transformed into the object; however, some validations are done using this format. We will refer to this format as the API BDMs from now on.

   2.2 The virt driver format - this is the format defined by the classes in :mod: *nova.virt.block_device*. This format is used and expected by the code in the various virt drivers. These classes, in addition to exposing a different format (mimicking the Python dict interface), also provide a place to bundle some functionality common to certain types of block devices (for example attaching volumes which has to interact with both Cinder and the virt driver code). We will refer to this format as Driver BDMs from now on.

Note: The maximum limit on the number of disk devices allowed to attach to a single server is configurable with the option `compute.max_disk_devices_to_attach`.

4.48.1 Data format and its history

In the early days of Nova, block device mapping general structure closely mirrored that of the EC2 API. During the Havana release of Nova, block device handling code, and in turn the block device mapping structure, had work done on improving the generality and usefulness. These improvements included exposing additional details and features in the API. In order to facilitate this, a new extension was added to the v2 API called *BlockDeviceMappingV2Boot*[2], that added an additional *block_device_mapping_v2* field to the instance boot API request.

Block device mapping v1 (aka legacy)

This was the original format that supported only cinder volumes (similar to how EC2 block devices support only EBS volumes). Every entry was keyed by device name (we will discuss why this was problematic in its own section later on this page), and would accept only:

- UUID of the Cinder volume or snapshot
- Type field - used only to distinguish between volumes and Cinder volume snapshots
- Optional size field
- Optional *delete_on_termination* flag

While all of Nova internal code only uses and stores the new data structure, we still need to handle API requests that use the legacy format. This is handled by the Nova API service on every request. As we will see later, since block device mapping information can also be stored in the image metadata in Glance, this is another place where we need to handle the v1 format. The code to handle legacy conversions is part of the :mod: *nova.block_device* module.

[1] In addition to the BlockDeviceMapping Nova object, we also have the BlockDeviceDict class in :mod: *nova.block_device* module. This class handles transforming and validating the API BDM format.

[2] This work predates API microversions and thus the only way to add it was by means of an API extension.

Intermezzo - problem with device names

Using device names as the primary per-instance identifier, and exposing them in the API, is problematic for Nova mostly because several hypervisors Nova supports with its drivers cant guarantee that the device names the guest OS assigns are the ones the user requested from Nova. Exposing such a detail in the public API of Nova is obviously not ideal, but it needed to stay for backwards compatibility. It is also required for some (slightly obscure) features around overloading a block device in a Glance image when booting an instance[3].

The plan for fixing this was to allow users to not specify the device name of a block device, and Nova will determine it (with the help of the virt driver), so that it can still be discovered through the API and used when necessary, like for the features mentioned above (and preferably only then).

Another use for specifying the device name was to allow the boot from volume functionality, by specifying a device name that matches the root device name for the instance (usually */dev/vda*).

Currently (mid Liberty) users are discouraged from specifying device names for all calls requiring or allowing block device mapping, except when trying to override the image block device mapping on instance boot, and it will likely remain like that in the future. Libvirt device driver will outright override any device names passed with its own values.

Block device mapping v2

New format was introduced in an attempt to solve issues with the original block device mapping format discussed above, and also to allow for more flexibility and addition of features that were not possible with the simple format we had.

New block device mapping is a list of dictionaries containing the following fields (in addition to the ones that were already there):

- source_type - this can have one of the following values:
  - *image*
  - *volume*
  - *snapshot*
  - *blank*
- destination_type - this can have one of the following values:
  - *local*
  - *volume*
- guest_format - Tells Nova how/if to format the device prior to attaching, should be only used with blank local images. Denotes a swap disk if the value is *swap*.
- device_name - See the previous section for a more in depth explanation of this - currently best left empty (not specified that is), unless the user wants to override the existing device specified in the image metadata. In case of Libvirt, even when passed in with the purpose of overriding the existing image metadata, final set of device names for the instance may still get changed by the driver.
- disk_bus and device_type - low level details that some hypervisors (currently only libvirt) may support. Some example disk_bus values can be: *ide*, *usb*, *virtio*, *scsi*, while device_type may be *disk*, *cdrom*, *floppy*, *lun*. This is not an exhaustive list as it depends on the virtualization driver, and may change as more support is added. Leaving these empty is the most common thing to do.

[3] This is a feature that the EC2 API offers as well and has been in Nova for a long time, although it has been broken in several releases. More info can be found on *this bug <https://launchpad.net/bugs/1370250>*

- boot_index - Defines the order in which a hypervisor will try devices when attempting to boot the guest from storage. Each device which is capable of being used as boot device should be given a unique boot index, starting from 0 in ascending order. Some hypervisors may not support booting from multiple devices, so will only consider the device with boot index of 0. Some hypervisors will support booting from multiple devices, but only if they are of different types - eg a disk and CD-ROM. Setting a negative value or None indicates that the device should not be used for booting. The simplest usage is to set it to 0 for the boot device and leave it as None for any other devices.
- volume_type - Added in microversion 2.67 to the servers create API to support specifying volume type when booting instances. When we snapshot a volume-backed server, the block_device_mapping_v2 image metadata will include the volume_type from the BDM record so if the user then creates another server from that snapshot, the volume that nova creates from that snapshot will use the same volume_type. If a user wishes to change that volume type in the image metadata, they can do so via the image API.

Valid source / destination combinations

Combination of the `source_type` and `destination_type` will define the kind of block device the entry is referring to. The following combinations are supported:

- *image* -> *local* - this is only currently reserved for the entry referring to the Glance image that the instance is being booted with (it should also be marked as a boot device). It is also worth noting that an API request that specifies this, also has to provide the same Glance uuid as the *image_ref* parameter to the boot request (this is done for backwards compatibility and may be changed in the future). This functionality might be extended to specify additional Glance images to be attached to an instance after boot (similar to kernel/ramdisk images) but this functionality is not supported by any of the current drivers.
- *volume* -> *volume* - this is just a Cinder volume to be attached to the instance. It can be marked as a boot device.
- *snapshot* -> *volume* - this works exactly as passing *type=snap* does. It would create a volume from a Cinder volume snapshot and attach that volume to the instance. Can be marked bootable.
- *image* -> *volume* - As one would imagine, this would download a Glance image to a cinder volume and attach it to an instance. Can also be marked as bootable. This is really only a shortcut for creating a volume out of an image before booting an instance with the newly created volume.
- *blank* -> *volume* - Creates a blank Cinder volume and attaches it. This will also require the volume size to be set.
- *blank* -> *local* - Depending on the guest_format field (see below), this will either mean an ephemeral blank disk on hypervisor local storage, or a swap disk (instances can have only one of those).

Nova will not allow mixing of BDMv1 and BDMv2 in a single request, and will do basic validation to make sure that the requested block device mapping is valid before accepting a boot request.

4.48.2 FAQs

1. Is it possible to configure nova to automatically use cinder to back all root disks with volumes?

   No, there is nothing automatic within nova that converts a non-boot-from-volume request to convert the image to a root volume. Several ideas have been discussed over time which are captured in the spec for volume-backed flavors. However, if you wish to force users to always create volume-backed servers, you can configure the API service by setting `max_local_block_devices` to 0. This will result in any non-boot-from-volume server create request to fail with a 400 response.

4.49 Cells

Before reading further, there is a nice overview presentation that Andrew Laski gave at the Austin (Newton) summit which is worth watching.

4.49.1 Cells V2

- Newton Summit Video - Nova Cells V2: Whats Going On?
- Pike Summit Video - Scaling Nova: How CellsV2 Affects Your Deployment
- Queens Summit Video - Add Cellsv2 to your existing Nova deployment
- Rocky Summit Video - Moving from CellsV1 to CellsV2 at CERN
- Stein Summit Video - Scaling Nova with CellsV2: The Nova Developer and the CERN Operator perspective

Manifesto

Proposal

Right now, when a request hits the Nova API for a particular instance, the instance information is fetched from the database, which contains the hostname of the compute node on which the instance currently lives. If the request needs to take action on the instance (which is most of them), the hostname is used to calculate the name of a queue, and a message is written there which finds its way to the proper compute node.

The meat of this proposal is changing the above hostname lookup into two parts that yield three pieces of information instead of one. Basically, instead of merely looking up the *name* of the compute node on which an instance lives, we will also obtain database and queue connection information. Thus, when asked to take action on instance $foo, we will:

1. Lookup the three-tuple of (database, queue, hostname) for that instance
2. Connect to that database and fetch the instance record
3. Connect to the queue and send the message to the proper hostname queue

The above differs from the current organization in two ways. First, we need to do two database lookups before we know where the instance lives. Second, we need to demand-connect to the appropriate database and queue. Both of these have performance implications, but we believe we can mitigate the impacts through the use of things like a memcache of instance mapping information and pooling of connections to database and queue systems. The number of cells will always be much smaller than the number of instances.

There are availability implications with this change since something like a nova list which might query multiple cells could end up with a partial result if there is a database failure in a cell. See *CellsV2 Management* for knowing more about the recommended practices under such situations. A database failure within a cell would cause larger issues than a partial list result so the expectation is that it would be addressed quickly and cellsv2 will handle it by indicating in the response that the data may not be complete.

Since this is very similar to what we have with current cells, in terms of organization of resources, we have decided to call this cellsv2 for disambiguation.

After this work is complete there will no longer be a no cells deployment. The default installation of Nova will be a single cell setup.

Benefits

The benefits of this new organization are:

- Native sharding of the database and queue as a first-class-feature in nova. All of the code paths will go through the lookup procedure and thus we wont have the same feature parity issues as we do with current cells.
- No high-level replication of all the cell databases at the top. The API will need a database of its own for things like the instance index, but it will not need to replicate all the data at the top level.
- It draws a clear line between global and local data elements. Things like flavors and keypairs are clearly global concepts that need only live at the top level. Providing this separation allows compute nodes to become even more stateless and insulated from things like deleted/changed global data.
- Existing non-cells users will suddenly gain the ability to spawn a new cell from their existing deployment without changing their architecture. Simply adding information about the new database and queue systems to the new index will allow them to consume those resources.
- Existing cells users will need to fill out the cells mapping index, shutdown their existing cells synchronization service, and ultimately clean up their top level database. However, since the high-level organization is not substantially different, they will not have to re-architect their systems to move to cellsv2.
- Adding new sets of hosts as a new cell allows them to be plugged into a deployment and tested before allowing builds to be scheduled to them.

Database split

As mentioned above there is a split between global data and data that is local to a cell.

The following is a breakdown of what data can uncontroversially considered global versus local to a cell. Missing data will be filled in as consensus is reached on the data that is more difficult to cleanly place. The missing data is mostly concerned with scheduling and networking.

Global (API-level) Tables

instance_types instance_type_projects instance_type_extra_specs quotas project_user_quotas quota_classes quota_usages security_groups security_group_rules security_group_default_rules provider_fw_rules key_pairs migrations networks tags

Cell-level Tables

instances instance_info_caches instance_extra instance_metadata instance_system_metadata instance_faults instance_actions instance_actions_events instance_id_mappings pci_devices block_device_mapping virtual_interfaces

4.49.2 Setup of Cells V2

Overview

As more of the CellsV2 implementation is finished, all operators are required to make changes to their deployment. For all deployments (even those that only intend to have one cell), these changes are configuration-related, both in the main nova configuration file as well as some extra records in the databases.

All nova deployments must now have the following databases available and configured:

1. The API database
2. One special cell database called cell0
3. One (or eventually more) cell databases

Thus, a small nova deployment will have an API database, a cell0, and what we will call here a cell1 database. High-level tracking information is kept in the API database. Instances that are never scheduled are relegated to the cell0 database, which is effectively a graveyard of instances that failed to start. All successful/running instances are stored in cell1.

Note: Since Nova services make use of both configuration file and some databases records, starting or restarting those services with an incomplete configuration could lead to an incorrect deployment. Please only restart the services once you are done with the described steps below.

First Time Setup

Since there is only one API database, the connection information for it is stored in the nova.conf file.

```
[api_database]
connection = mysql+pymysql://root:secretmysql@dbserver/nova_api?charset=utf8
```

Since there may be multiple cell databases (and in fact everyone will have cell0 and cell1 at a minimum), connection info for these is stored in the API database. Thus, you must have connection information in your config file for the API database before continuing to the steps below, so that *nova-manage* can find your other databases.

The following examples show the full expanded command line usage of the setup commands. This is to make it easier to visualize which of the various URLs are used by each of the commands. However, you should be able to put all of that in the config file and *nova-manage* will use those values. If need be, you can create separate config files and pass them as *nova-manage –config-file foo.conf* to control the behavior without specifying things on the command lines.

The commands below use the API database so remember to run *nova-manage api_db sync* first.

First we will create the necessary records for the cell0 database. To do that we use *nova-manage* like this:

```
nova-manage cell_v2 map_cell0 --database_connection \
  mysql+pymysql://root:secretmysql@dbserver/nova_cell0?charset=utf8
```

Note: If you dont specify *–database_connection* then *nova-manage* will use the *[database]/connection* value from your config file, and mangle the database name to have a *_cell0* suffix.

Warning: If your databases are on separate hosts then you should specify *–database_connection* or make certain that the nova.conf being used has the *[database]/connection* value pointing to the same user/password/host that will work for the cell0 database. If the cell0 mapping was created incorrectly, it can be deleted using the *nova-manage cell_v2 delete_cell* command and then run *map_cell0* again with the proper database connection value.

Since no hosts are ever in cell0, nothing further is required for its setup. Note that all deployments only ever have one cell0, as it is special, so once you have done this step you never need to do it again, even if you add more regular cells.

Now, we must create another cell which will be our first regular cell, which has actual compute hosts in it, and to which instances can actually be scheduled. First, we create the cell record like this:

```
nova-manage cell_v2 create_cell --verbose --name cell1 \
  --database_connection  mysql+pymysql://root:secretmysql@127.0.0.1/nova?charset=utf8
  --transport-url rabbit://stackrabbit:secretrabbit@mqserver:5672/
```

Note: If you dont specify the database and transport urls then *nova-manage* will use the *[database]/connection* and *[DEFAULT]/transport_url* values from the config file.

Note: At this point, the API database can now find the cell database, and further commands will attempt to look inside. If this is a completely fresh database (such as if youre adding a cell, or if this is a new deployment), then you will need to run *nova-manage db sync* on it to initialize the schema.

The *nova-manage cell_v2 create_cell* command will print the UUID of the newly-created cell if *–verbose* is passed, which is useful if you need to run commands like *discover_hosts* targeted at a specific cell.

Now we have a cell, but no hosts are in it which means the scheduler will never actually place instances there. The next step is to scan the database for compute node records and add them into the cell we just created. For this step, you must have had a compute node started such that it registers itself as a running service. Once that has happened, you can scan and add it to the cell:

```
nova-manage cell_v2 discover_hosts
```

This command will connect to any databases for which you have created cells (as above), look for hosts that have registered themselves there, and map those hosts in the API database so that they are visible to the scheduler as available targets for instances. Any time you add more compute hosts to a cell, you need to re-run this command to map them from the top-level so they can be utilized.

Template URLs in Cell Mappings

Starting in the Rocky release, the URLs provided in the cell mappings for `--database_connection` and `--transport-url` can contain variables which are evaluated each time they are loaded from the database, and the values of which are taken from the corresponding base options in the hosts configuration file. The base URL is parsed and the following elements may be substituted into the cell mapping URL (using `rabbit://bob:s3kret@myhost:123/nova?sync=true#extra`):

Table 169: Cell Mapping URL Variables

Variable	Meaning	Part of example URL
`scheme`	The part before the *://*	`rabbit`
`username`	The username part of the credentials	`bob`
`password`	The password part of the credentials	`s3kret`
`hostname`	The hostname or address	`myhost`
`port`	The port number (must be specified)	`123`
`path`	The path part of the URL (without leading slash)	`nova`
`query`	The full query string arguments (without leading question mark)	`sync=true`
`fragment`	Everything after the first hash mark	`extra`

Variables are provided in curly brackets, like `{username}`. A simple template of `rabbit://{username}:{password}@otherhost/{path}` will generate a full URL of `rabbit://bob:s3kret@otherhost/nova` when used with the above example.

Note: The `[database]/connection` and `[DEFAULT]/transport_url` values are not reloaded from the configuration file during a SIGHUP, which means that a full service restart will be required to notice changes in a cell mapping record if variables are changed.

Note: The `[DEFAULT]/transport_url` option can contain an extended syntax for the netloc part of the url (i.e. *userA:passwordA@hostA:portA,userB:passwordB:hostB:portB*). In this case, substitions of the form `username1`, `username2`, etc will be honored and can be used in the template URL.

The templating of these URLs may be helpful in order to provide each service host with its own credentials for, say, the database. Without templating, all hosts will use the same URL (and thus credentials) for accessing services like the database and message queue. By using a URL with a template that results in the credentials being taken from the host-local configuration file, each host will use different values for those connections.

Assuming you have two service hosts that are normally configured with the cell0 database as their primary connection, their (abbreviated) configurations would look like this:

```
[database]
connection = mysql+pymysql://service1:foo@myapidbhost/nova_cell0
```

and:

```
[database]
connection = mysql+pymysql://service2:bar@myapidbhost/nova_cell0
```

Without cell mapping template URLs, they would still use the same credentials (as stored in the mapping) to connect to the cell databases. However, consider template URLs like the following:

```
mysql+pymysql://{username}:{password}@mycell1dbhost/nova
```

and:

```
mysql+pymysql://{username}:{password}@mycell2dbhost/nova
```

Using the first service and cell1 mapping, the calculated URL that will actually be used for connecting to that database will be:

```
mysql+pymysql://service1:foo@mycell1dbhost/nova
```

References

- *nova-manage man page*

4.49.3 Step-By-Step for Common Use Cases

The following are step-by-step examples for common use cases setting up Cells V2. This is intended as a quick reference that puts together everything explained in *Setup of Cells V2*. It is assumed that you have followed all other install steps for Nova and are setting up Cells V2 specifically at this point.

Fresh Install

You are installing Nova for the first time and have no compute hosts in the database yet. This will set up a single cell Nova deployment.

1. Reminder: You should have already created and synced the Nova API database by creating a database, configuring its connection in the `[api_database]/connection` setting in the Nova configuration file, and running `nova-manage api_db sync`.
2. Create a database for cell0. If you are going to pass the database connection url on the command line in step 3, you can name the cell0 database whatever you want. If you are not going to pass the database url on the command line in step 3, you need to name the cell0 database based on the name of your existing Nova database: <Nova database name>_cell0. For example, if your Nova database is named `nova`, then your cell0 database should be named `nova_cell0`.
3. Run the `map_cell0` command to create and map cell0:

```
nova-manage cell_v2 map_cell0 \
  --database_connection <database connection url>
```

   The database connection url is generated based on the `[database]/connection` setting in the Nova configuration file if not specified on the command line.

4. Run `nova-manage db sync` to populate the cell0 database with a schema. The `db sync` command reads the database connection for cell0 that was created in step 3.
5. Run the `create_cell` command to create the single cell which will contain your compute hosts:

```
nova-manage cell_v2 create_cell --name <name> \
  --transport-url <transport url for message queue> \
  --database_connection <database connection url>
```

   The transport url is taken from the `[DEFAULT]/transport_url` setting in the Nova configuration file if not specified on the command line. The database url is taken from the `[database]/connection` setting in the Nova configuration file if not specified on the command line.

6. Configure your compute host(s), making sure `[DEFAULT]/transport_url` matches the transport URL for the cell created in step 5, and start the nova-compute service. Before step 7, make sure you have compute hosts in the database by running:

```
nova service-list --binary nova-compute
```

7. Run the `discover_hosts` command to map compute hosts to the single cell by running:

```
nova-manage cell_v2 discover_hosts
```

   The command will search for compute hosts in the database of the cell you created in step 5 and map them to the cell. You can also configure a periodic task to have Nova discover new hosts automatically by setting the `[scheduler]/discover_hosts_in_cells_interval` to a time interval in seconds. The periodic task is run by the nova-scheduler service, so you must be sure to configure it on all of your nova-scheduler hosts.

Note: Remember: In the future, whenever you add new compute hosts, you will need to run the `discover_hosts` command after starting them to map them to the cell if you did not configure the automatic host discovery in step 7.

Upgrade (minimal)

You are upgrading an existing Nova install and have compute hosts in the database. This will set up a single cell Nova deployment.

1. If you havent already created a cell0 database in a prior release, create a database for cell0 with a name based on the name of your Nova database: <Nova database name>_cell0. If your Nova database is named `nova`, then your cell0 database should be named `nova_cell0`.

Warning: In Newton, the `simple_cell_setup` command expects the name of the cell0 database to be based on the name of the Nova API database: <Nova API database name>_cell0 and the database connection url is taken from the `[api_database]/connection` setting in the Nova configuration file.

2. Run the `simple_cell_setup` command to create and map cell0, create and map the single cell, and map existing compute hosts and instances to the single cell:

```
nova-manage cell_v2 simple_cell_setup \
  --transport-url <transport url for message queue>
```

The transport url is taken from the `[DEFAULT]/transport_url` setting in the Nova configuration file if not specified on the command line. The database connection url will be generated based on the `[database]/connection` setting in the Nova configuration file.

Note: Remember: In the future, whenever you add new compute hosts, you will need to run the `discover_hosts` command after starting them to map them to the cell. You can also configure a periodic task to have Nova discover new hosts automatically by setting the `[scheduler]/discover_hosts_in_cells_interval` to a time interval in seconds. The periodic task is run by the nova-scheduler service, so you must be sure to configure it on all of your nova-scheduler hosts.

Upgrade with Cells V1

Todo: This needs to be removed but *Adding a new cell to an existing deployment* is still using it.

You are upgrading an existing Nova install that has Cells V1 enabled and have compute hosts in your databases. This will set up a multiple cell Nova deployment. At this time, it is recommended to keep Cells V1 enabled during and after the upgrade as multiple Cells V2 cell support is not fully finished and may not work properly in all scenarios. These upgrade steps will help ensure a simple cutover from Cells V1 to Cells V2 in the future.

Note: There is a Rocky summit video from CERN about how they did their upgrade from cells v1 to v2 here:

https://www.openstack.org/videos/vancouver-2018/moving-from-cellsv1-to-cellsv2-at-cern

1. If you havent already created a cell0 database in a prior release, create a database for cell0. If you are going to pass the database connection url on the command line in step 2, you can name the cell0 database whatever you want. If you are not going to pass the database url on the command line in step 2, you need to name the cell0 database based on the name of your existing Nova database: <Nova database name>_cell0. For example, if your Nova database is named `nova`, then your cell0 database should be named `nova_cell0`.
2. Run the `map_cell0` command to create and map cell0:

```
nova-manage cell_v2 map_cell0 \
  --database_connection <database connection url>
```

The database connection url is generated based on the `[database]/connection` setting in the Nova configuration file if not specified on the command line.

3. Run `nova-manage db sync` to populate the cell0 database with a schema. The `db sync` command reads the database connection for cell0 that was created in step 2.

4. Run the `create_cell` command to create cells which will contain your compute hosts:

```
nova-manage cell_v2 create_cell --name <cell name> \
  --transport-url <transport url for message queue> \
  --database_connection <database connection url>
```

You will need to repeat this step for each cell in your deployment. Your existing cell database will be re-used – this simply informs the top-level API database about your existing cell databases.

It is a good idea to specify a name for the new cell you create so you can easily look up cell uuids with the `list_cells` command later if needed.

The transport url is taken from the `[DEFAULT]/transport_url` setting in the Nova configuration file if not specified on the command line. The database url is taken from the `[database]/connection` setting in the Nova configuration file if not specified on the command line. If you are not going to specify `--database_connection` and `--transport-url` on the command line, be sure to specify your existing cell Nova configuration file:

```
nova-manage --config-file <cell nova.conf> cell_v2 create_cell \
  --name <cell name>
```

5. Run the `discover_hosts` command to map compute hosts to cells:

```
nova-manage cell_v2 discover_hosts --cell_uuid <cell uuid>
```

You will need to repeat this step for each cell in your deployment unless you omit the `--cell_uuid` option. If the cell uuid is not specified on the command line, `discover_hosts` will search for compute hosts in each cell database and map them to the corresponding cell. You can use the `list_cells` command to look up cell uuids if you are going to specify `--cell_uuid`.

You can also configure a periodic task to have Nova discover new hosts automatically by setting the `[scheduler]/discover_hosts_in_cells_interval` to a time interval in seconds. The periodic task is run by the nova-scheduler service, so you must be sure to configure it on all of your nova-scheduler hosts.

6. Run the `map_instances` command to map instances to cells:

```
nova-manage cell_v2 map_instances --cell_uuid <cell uuid> \
  --max-count <max count>
```

You will need to repeat this step for each cell in your deployment. You can use the `list_cells` command to look up cell uuids.

The `--max-count` option can be specified if you would like to limit the number of instances to map in a single run. If `--max-count` is not specified, all instances will be mapped. Repeated runs of the command will start from where the last run finished so it is not necessary to increase `--max-count` to finish. An exit code of 0 indicates that all instances have been mapped. An exit code of 1 indicates that there are remaining instances that need to be mapped.

Note: Remember: In the future, whenever you add new compute hosts, you will need to run the `discover_hosts` command after starting them to map them to a cell if you did not configure the automatic host discovery in step 5.

Adding a new cell to an existing deployment

To expand your deployment with a new cell, first follow the usual steps for standing up a new Cells V1 cell. After that is finished, follow step 4 in *Upgrade with Cells V1* to create a new Cells V2 cell for it. If you have added new compute hosts for the new cell, you will also need to run the `discover_hosts` command after starting them to map them to the new cell if you did not configure the automatic host discovery as described in step 5 in *Upgrade with Cells V1*.

References

- *nova-manage man page*

4.49.4 FAQs

1. How do I find out which hosts are bound to which cell?

   There are a couple of ways to do this.

   1. Run `nova-manage --config-file <cell config> host list`. This will only lists hosts in the provided cell nova.conf.

      Deprecated since version 16.0.0: The `nova-manage host list` command is deprecated as of the 16.0.0 Pike release.

   2. Run `nova-manage cell_v2 discover_hosts --verbose`. This does not produce a report but if you are trying to determine if a host is in a cell you can run this and it will report any hosts that are not yet mapped to a cell and map them. This command is idempotent.

   3. Run `nova-manage cell_v2 list_hosts`. This will list hosts in all cells. If you want to list hosts in a specific cell, you can run `nova-manage cell_v2 list_hosts --cell_uuid <cell_uuid>`.

2. I updated the database_connection and/or transport_url in a cell using the `nova-manage cell_v2 update_cell` command but the API is still trying to use the old settings.

   The cell mappings are cached in the nova-api service worker so you will need to restart the worker process to rebuild the cache. Note that there is another global cache tied to request contexts, which is used in the nova-conductor and nova-scheduler services, so you might need to do the same if you are having the same issue in those services. As of the 16.0.0 Pike release there is no timer on the cache or hook to refresh the cache using a SIGHUP to the service.

3. I have upgraded from Newton to Ocata and I can list instances but I get a 404 NotFound error when I try to get details on a specific instance.

   Instances need to be mapped to cells so the API knows which cell an instance lives in. When upgrading, the `nova-manage cell_v2 simple_cell_setup` command will automatically map the instances to the single cell which is backed by the existing nova database. If you have already upgraded and did not use the `simple_cell_setup` command, you can run the `nova-manage cell_v2 map_instances --cell_uuid <cell_uuid>` command to map all instances in the given cell. See the *Nova Cells v2* man page for details on command usage.

4. Should I change any of the `[cells]` configuration options for Cells v2?

   NO. Those options are for Cells v1 usage only and are not used at all for Cells v2. That includes the `nova-cells` service - it has nothing to do with Cells v2.

5. Can I create a cell but have it disabled from scheduling?

   Yes. It is possible to create a pre-disabled cell such that it does not become a candidate for scheduling new VMs. This can be done by running the `nova-manage cell_v2 create_cell` command with the `--disabled` option.

6. How can I disable a cell so that the new server create requests do not go to it while I perform maintenance?

   Existing cells can be disabled by running `nova-manage cell_v2 update_cell --cell_uuid <cell_uuid> --disable` and can be re-enabled once the maintenance period is over by running `nova-manage cell_v2 update_cell --cell_uuid <cell_uuid> --enable`

7. I disabled (or enabled) a cell using the `nova-manage cell_v2 update_cell` or I created a new (pre-disabled) cell(mapping) using the `nova-manage cell_v2 create_cell` command but the scheduler is still using the old settings.

   The cell mappings are cached in the scheduler worker so you will either need to restart the scheduler process to refresh the cache, or send a SIGHUP signal to the scheduler by which it will automatically refresh the cells cache and the changes will take effect.

8. Why was the cells REST API not implemented for CellsV2? Why are there no CRUD operations for cells in the API?

   One of the deployment challenges that CellsV1 had was the requirement for the API and control services to be up before a new cell could be deployed. This was not a problem for large-scale public clouds that never shut down, but is not a reasonable requirement for smaller clouds that do offline upgrades and/or clouds which could be taken completely offline by something like a power outage. Initial devstack and gate testing for CellsV1 was delayed by the need to engineer a solution for bringing the services partially online in order to deploy the rest, and this continues to be a gap for other deployment tools. Consider also the FFU case where the control plane needs to be down for a multi-release upgrade window where changes to cell records have to be made. This would be quite a bit harder if the way those changes are made is via the API, which must remain down during the process.

   Further, there is a long-term goal to move cell configuration (i.e. cell_mappings and the associated URLs and credentials) into config and get away from the need to store and provision those things in the database. Obviously a CRUD interface in the API would prevent us from making that move.

9. Why are cells not exposed as a grouping mechanism in the API for listing services, instances, and other resources?

   Early in the design of CellsV2 we set a goal to not let the cell concept leak out of the API, even for operators. Aggregates are the way nova supports grouping of hosts for a variety of reasons, and aggregates can cut across cells, and/or be aligned with them if desired. If we were to support cells as another grouping mechanism, we would likely end up having to implement many of the same features for them as aggregates, such as scheduler features, metadata, and other searching/filtering operations. Since aggregates are how Nova supports grouping, we expect operators to use aggregates any time they need to refer to a cell as a group of hosts from the API, and leave actual cells as a purely architectural detail.

   The need to filter instances by cell in the API can and should be solved by adding a generic by-aggregate filter, which would allow listing instances on hosts contained within any aggregate, including one that matches the cell boundaries if so desired.

10. Why are the API responses for `GET /servers`, `GET /servers/detail`, `GET /servers/{server_id}` and `GET /os-services` missing some information for certain cells at certain times? Why do I see the status as UNKNOWN for the servers in those cells at those times when I run `openstack server list` or `openstack server show`?

Starting from microversion 2.69 the API responses of `GET /servers`, `GET /servers/detail`, `GET /servers/{server_id}` and `GET /os-services` may contain missing keys during down cell situations. See the Handling Down Cells section of the Compute API guide for more information on the partial constructs.

For administrative considerations, see *Handling cell failures*.

4.50 Cells Layout (v2)

This document describes the layout of a deployment with Cells version 2, including deployment considerations for security and scale. It is focused on code present in Pike and later, and while it is geared towards people who want to have multiple cells for whatever reason, the nature of the cellsv2 support in Nova means that it applies in some way to all deployments.

4.50.1 Concepts

A basic Nova system consists of the following components:

- The nova-api service which provides the external REST API to users.
- The nova-scheduler and placement services which are responsible for tracking resources and deciding which compute node instances should be on.
- An API database that is used primarily by nova-api and nova-scheduler (called *API-level services* below) to track location information about instances, as well as a temporary location for instances being built but not yet scheduled.
- The nova-conductor service which offloads long-running tasks for the API-level service, as well as insulates compute nodes from direct database access
- The nova-compute service which manages the virt driver and hypervisor host.
- A cell database which is used by API, conductor and compute services, and which houses the majority of the information about instances.
- A cell0 database which is just like the cell database, but contains only instances that failed to be scheduled.
- A message queue which allows the services to communicate with each other via RPC.

All deployments have at least the above components. Small deployments likely have a single message queue that all services share, and a single database server which hosts the API database, a single cell database, as well as the required cell0 database. This is considered a single-cell deployment because it only has one real cell. The cell0 database mimics a regular cell, but has no compute nodes and is used only as a place to put instances that fail to land on a real compute node (and thus a real cell).

The purpose of the cells functionality in nova is specifically to allow larger deployments to shard their many compute nodes into cells, each of which has a database and message queue. The API database is always and only global, but there can be many cell databases (where the bulk of the instance information lives), each with a portion of the instances for the entire deployment within.

All of the nova services use a configuration file, all of which will at a minimum specify a message queue endpoint (i.e. `[DEFAULT]/transport_url`). Most of the services also require configuration of database connection information (i.e. `[database]/connection`). API-level services that need access to the global routing and placement information will also be configured to reach the API database (i.e. `[api_database]/connection`).

Note: The pair of `transport_url` and `[database]/connection` configured for a service defines what cell a service lives in.

API-level services need to be able to contact other services in all of the cells. Since they only have one configured `transport_url` and `[database]/connection` they look up the information for the other cells in the API database, with records called *cell mappings*.

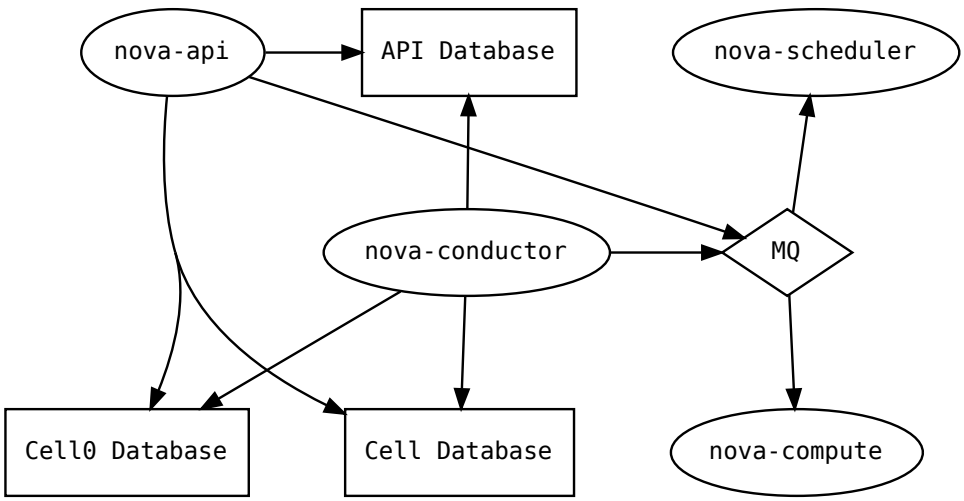
Note: The API database must have cell mapping records that match the `transport_url` and `[database]/connection` configuration elements of the lower-level services. See the `nova-manage` *Nova Cells v2* commands for more information about how to create and examine these records.

4.50.2 Service Layout

The services generally have a well-defined communication pattern that dictates their layout in a deployment. In a small/simple scenario, the rules do not have much of an impact as all the services can communicate with each other on a single message bus and in a single cell database. However, as the deployment grows, scaling and security concerns may drive separation and isolation of the services.

Simple

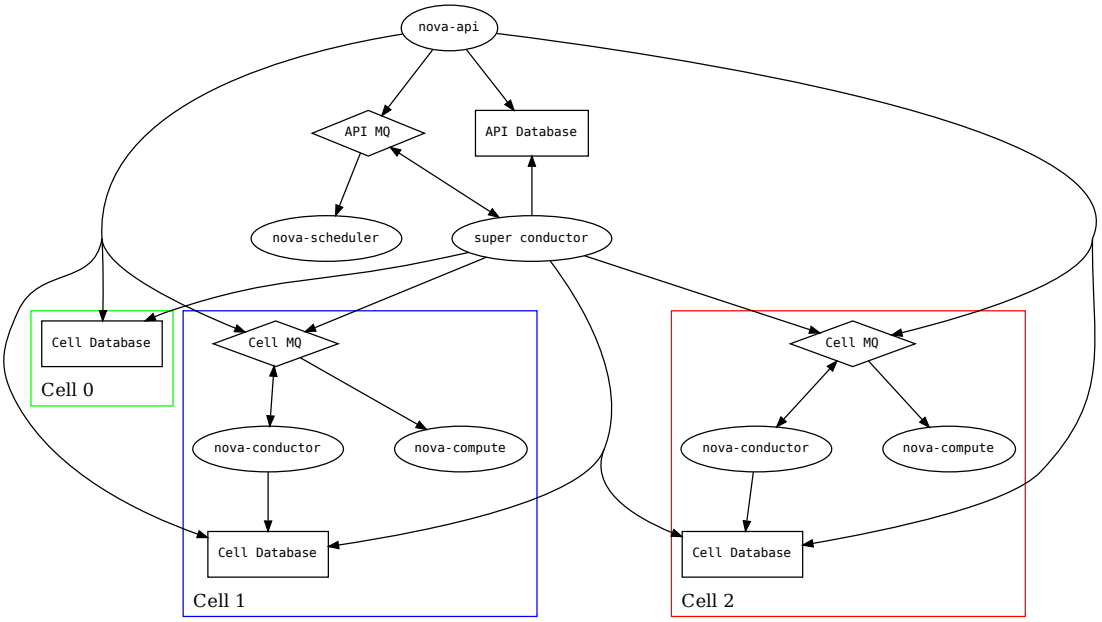
This is a diagram of the basic services that a simple (single-cell) deployment would have, as well as the relationships (i.e. communication paths) between them:



All of the services are configured to talk to each other over the same message bus, and there is only one cell database where live instance data resides. The cell0 database is present (and required) but as no compute nodes are connected to it, this is still a single cell deployment.

Multiple Cells

In order to shard the services into multiple cells, a number of things must happen. First, the message bus must be split into pieces along the same lines as the cell database. Second, a dedicated conductor must be run for the API-level services, with access to the API database and a dedicated message queue. We call this *super conductor* to distinguish its place and purpose from the per-cell conductor nodes.



It is important to note that services in the lower cell boxes only have the ability to call back to the placement API but cannot access any other API-layer services via RPC, nor do they have access to the API database for global visibility of resources across the cloud. This is intentional and provides security and failure domain isolation benefits, but also has impacts on some things that would otherwise require this any-to-any communication style. Check the release notes for the version of Nova you are using for the most up-to-date information about any caveats that may be present due to this limitation.

Caveats of a Multi-Cell deployment

Note: This information is correct as of the Pike release. Where improvements have been made or issues fixed, they are noted per item.

Cross-cell instance migrations

Currently it is not possible to migrate an instance from a host in one cell to a host in another cell. This may be possible in the future, but it is currently unsupported. This impacts cold migration, resizes, live migrations, evacuate, and unshelve operations.

Quota-related quirks

Quotas are now calculated live at the point at which an operation would consume more resource, instead of being kept statically in the database. This means that a multi-cell environment may incorrectly calculate the usage of a tenant if one of the cells is unreachable, as those resources cannot be counted. In this case, the tenant may be able to consume more resource from one of the available cells, putting them far over quota when the unreachable cell returns.

Note: Starting in the Train (20.0.0) release, it is possible to configure counting of quota usage from the placement service and API database to make quota usage calculations resilient to down or poor-performing cells in a multi-cell environment. See the *quotas documentation* for more details.

Performance of listing instances

Note: This has been resolved in the Queens release[1].

With multiple cells, the instance list operation may not sort and paginate results properly when crossing multiple cell boundaries. Further, the performance of a sorted list operation will be considerably slower than with a single cell.

Notifications

With a multi-cell environment with multiple message queues, it is likely that operators will want to configure a separate connection to a unified queue for notifications. This can be done in the configuration file of all nodes. Refer to the oslo.messaging configuration documentation for more details.

Nova Metadata API service

Starting from the Stein release, the *nova metadata API service* can be run either globally or per cell using the `api.local_metadata_per_cell` configuration option.

Global

If you have networks that span cells, you might need to run Nova metadata API globally. When running globally, it should be configured as an API-level service with access to the `api_database.connection` information. The nova metadata API service **must not** be run as a standalone service, using the **`nova-api-metadata`** service, in this case.

Local per cell

Running Nova metadata API per cell can have better performance and data isolation in a multi-cell deployment. If your networks are segmented along cell boundaries, then you can run Nova metadata API service per cell. If you choose to run it per cell, you should also configure each neutron-metadata-agent service to point to the corresponding **`nova-api-metadata`**. The nova metadata API service **must** be run as a standalone service, using the **`nova-api-metadata`** service, in this case.

[1] https://blueprints.launchpad.net/nova/+spec/efficient-multi-cell-instance-list-and-sort

Console proxies

Starting from the Rocky release, console proxies must be run per cell because console token authorizations are stored in cell databases. This means that each console proxy server must have access to the *database.connection* information for the cell database containing the instances for which it is proxying console access.

Operations Requiring upcalls

If you deploy multiple cells with a superconductor as described above, computes and cell-based conductors will not have the ability to speak to the scheduler as they are not connected to the same MQ. This is by design for isolation, but currently the processes are not in place to implement some features without such connectivity. Thus, anything that requires a so-called upcall will not function. This impacts the following:

1. Instance reschedules during boot and resize (part 1)

   Note: This has been resolved in the Queens release[2].

2. Instance affinity reporting from the compute nodes to scheduler
3. The late anti-affinity check during server create and evacuate
4. Querying host aggregates from the cell

   Note: This has been resolved in the Rocky release[3].

5. Attaching a volume and `[cinder]/cross_az_attach=False`
6. Instance reschedules during boot and resize (part 2)

The first is simple: if you boot an instance, it gets scheduled to a compute node, fails, it would normally be re-scheduled to another node. That requires scheduler intervention and thus it will not work in Pike with a multi-cell layout. If you do not rely on reschedules for covering up transient compute-node failures, then this will not affect you. To ensure you do not make futile attempts at rescheduling, you should set `[scheduler]/max_attempts=1` in `nova.conf`.

The second two are related. The summary is that some of the facilities that Nova has for ensuring that affinity/anti-affinity is preserved between instances does not function in Pike with a multi-cell layout. If you dont use affinity operations, then this will not affect you. To make sure you dont make futile attempts at the affinity check, you should set `[workarounds]/disable_group_policy_check_upcall=True` and `[filter_scheduler]/track_instance_changes=False` in `nova.conf`.

The fourth is currently only a problem when performing live migrations using the XenAPI driver and not specifying `--block-migrate`. The driver will attempt to figure out if block migration should be performed based on source and destination hosts being in the same aggregate. Since aggregates data has migrated to the API database, the cell conductor will not be able to access the aggregate information and will fail.

The fifth is a problem because when a volume is attached to an instance in the *nova-compute* service, and `[cinder]/cross_az_attach=False` in nova.conf, we attempt to look up the availability zone that the instance is in which includes getting any host aggregates that the `instance.host` is in. Since the aggregates are in the API database and the cell conductor cannot access that information, so this will fail. In the future this check could be moved to the *nova-api* service such that the availability zone between the instance and the volume is checked before we reach the cell, except in the case of boot from volume where the *nova-compute* service itself creates the volume and must tell

[2] https://specs.openstack.org/openstack/nova-specs/specs/queens/approved/return-alternate-hosts.html
[3] https://blueprints.launchpad.net/nova/+spec/live-migration-in-xapi-pool

Cinder in which availability zone to create the volume. Long-term, volume creation during boot from volume should be moved to the top-level superconductor which would eliminate this AZ up-call check problem.

The sixth is detailed in bug 1781286 and similar to the first issue. The issue is that servers created without a specific availability zone will have their AZ calculated during a reschedule based on the alternate host selected. Determining the AZ for the alternate host requires an up call to the API DB.

4.51 Feature Classification

This document presents a matrix that describes which features are ready to be used and which features are works in progress. It includes links to relevant documentation and functional tests.

> **Warning:** Please note: this is a work in progress!

4.51.1 Aims

Users want reliable, long-term solutions for their use cases. The feature classification matrix identifies which features are complete and ready to use, and which should be used with caution.

The matrix also benefits developers by providing a list of features that require further work to be considered complete.

Below is a matrix for a selection of important verticals:

- *General Purpose Cloud Features*
- *NFV Cloud Features*
- *HPC Cloud Features*

For more details on the concepts in each matrix, please see *Notes on Concepts*.

4.51.2 General Purpose Cloud Features

This is a summary of the key features dev/test clouds, and other similar general purpose clouds need, and it describes their current state.

Below there are sections on NFV and HPC specific features. These look at specific features and scenarios that are important to those more specific sets of use cases. Summary

Feature	*Maturity*	Hyper-V CI	Ironic CI	libvirt+kvm (x86 & ppc64)	libvirt+kvm (s390x)	libvirt+virtuozzo CT	libvirt+virtuozzo VM	libvirt+xen	IBM PowerVM CI	VMware CI	XenServer CI	IBM zVM CI
Create Server and Delete Server	complete	✓	*?*	✓	*?*	✓	✓	✓	✓	✓	✓	✓
Snapshot Server	complete	*?*	*?*	✓	*?*	✓	✓	✓	✓	*?*	✓	✓
Server power ops	complete	✓	*?*	✓	*?*	✓	✓	✓	✓	✓	✓	✓
Rebuild Server	complete	✓	*?*	✓	*?*	✓	✓	✓		✓	✓	
Resize Server	complete	✓	*?*	✓	*?*	✓	✓	✓		✓	✓	
Volume Operations	complete	✓		✓	*?*	✓	✓	✓	✓	✓	✓	
Custom disk configurations on boot	complete	✓ *n*		✓	*?*		✓	✓		✓	✓	
Custom neutron configurations on boot	complete	✓		✓	*?*	*?*	*?*	✓	✓	✓	✓	✓
Pause a Server	complete	✓		✓	*?*		✓	✓		✓	✓	✓
Suspend a Server	complete	✓		✓	*?*	✓	✓	✓		✓	✓	
Server console output	complete	✓		✓	*?*	*?*	*?*	✓	✓	✓	✓	✓
Server Rescue	complete	✓		✓	*?*	✓	✓	✓		✓	✓	
Server Config Drive	complete	✓	✓	✓	*?*		✓	✓	✓	✓	✓	✓
Server Change Password	experimental	✓		✓	*?*						✓	
Server Shelve and Unshelve	complete	✓		✓	*?*		✓	✓	✓		✓	

Details

- **Create Server and Delete Server** This includes creating a server, and deleting a server. Specifically this is about booting a server from a glance image using the default disk and network configuration.

  info:

  - **Maturity:** `complete`

  - **API Docs:** https://docs.openstack.org/api-ref/compute/#servers-servers
  - **Admin Docs:** https://docs.openstack.org/nova/latest/user/launch-instances.html
  - **Tempest tests:** 9a438d88-10c6-4bcd-8b5b-5b6e25e1346f, 585e934c-448e-43c4-acbf-d06a9b899997

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `complete`
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+virtuozzo CT:** `partial`
  - **libvirt+virtuozzo VM:** `partial`
  - **libvirt+xen:** `complete`
  - **XenServer CI:** `complete`
  - **VMware CI:** `complete`
  - **Hyper-V CI:** `complete`
  - **Ironic CI:** `unknown`
  - **IBM PowerVM CI:** `complete`
  - **IBM zVM CI:** `complete`

- **Snapshot Server** This is creating a glance image from the currently running server.

  info:

  - **Maturity:** `complete`
  - **API Docs:** https://docs.openstack.org/api-ref/compute/?expanded=#servers-run-an-action-servers-action
  - **Admin Docs:** https://docs.openstack.org/glance/latest/admin/troubleshooting.html
  - **Tempest tests:** aaacd1d0-55a2-4ce8-818a-b5439df8adc9

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `complete`
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+virtuozzo CT:** `partial`
  - **libvirt+virtuozzo VM:** `partial`
  - **libvirt+xen:** `complete`
  - **XenServer CI:** `complete`
  - **VMware CI:** `unknown`
  - **Hyper-V CI:** `unknown`
  - **Ironic CI:** `unknown`
  - **IBM PowerVM CI:** `complete`
  - **IBM zVM CI:** `complete`

- **Server power ops** This includes reboot, shutdown and start.

  info:

  - **Maturity:** `complete`

  - **API Docs:** https://docs.openstack.org/api-ref/compute/?expanded=#servers-run-an-action-servers-action
  - **Admin Docs:**
  - **Tempest tests:** 2cb1baf6-ac8d-4429-bf0d-ba8a0ba53e32, af8eafd4-38a7-4a4b-bdbc-75145a580560

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `complete`
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+virtuozzo CT:** `partial`
  - **libvirt+virtuozzo VM:** `partial`
  - **libvirt+xen:** `complete`
  - **XenServer CI:** `complete`
  - **VMware CI:** `complete`
  - **Hyper-V CI:** `complete`
  - **Ironic CI:** `unknown`
  - **IBM PowerVM CI:** `complete`
  - **IBM zVM CI:** `complete`

- **Rebuild Server** You can rebuild a server, optionally specifying the glance image to use.

  info:

  - **Maturity:** `complete`
  - **API Docs:** https://docs.openstack.org/api-ref/compute/?expanded=#servers-run-an-action-servers-action
  - **Admin Docs:**
  - **Tempest tests:** aaa6cdf3-55a7-461a-add9-1c8596b9a07c

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `complete`
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+virtuozzo CT:** `partial`
  - **libvirt+virtuozzo VM:** `partial`
  - **libvirt+xen:** `complete`
  - **XenServer CI:** `complete`
  - **VMware CI:** `complete`
  - **Hyper-V CI:** `complete`
  - **Ironic CI:** `unknown`
  - **IBM PowerVM CI:** `missing`
  - **IBM zVM CI:** `missing`

- **Resize Server** You resize a server to a new flavor, then confirm or revert that operation.

  info:

  - **Maturity:** `complete`

  - **API Docs:** https://docs.openstack.org/api-ref/compute/?expanded=#servers-run-an-action-servers-action
  - **Admin Docs:**
  - **Tempest tests:** 1499262a-9328-4eda-9068-db1ac57498d2

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `complete`
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+virtuozzo CT:** `complete`
  - **libvirt+virtuozzo VM:** `partial`
  - **libvirt+xen:** `complete`
  - **XenServer CI:** `complete`
  - **VMware CI:** `complete`
  - **Hyper-V CI:** `complete`
  - **Ironic CI:** `unknown`
  - **IBM PowerVM CI:** `missing`
  - **IBM zVM CI:** `missing`

- **Volume Operations** This is about attaching volumes, detaching volumes.

  info:

  - **Maturity:** `complete`
  - **API Docs:** https://docs.openstack.org/api-ref/compute/#servers-with-volume-attachments-servers-os-volume-attachments
  - **Admin Docs:** https://docs.openstack.org/cinder/latest/admin/blockstorage-manage-volumes.html
  - **Tempest tests:** fff42874-7db5-4487-a8e1-ddda5fb5288d

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `complete`
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+virtuozzo CT:** `complete`
  - **libvirt+virtuozzo VM:** `complete`
  - **libvirt+xen:** `complete`
  - **XenServer CI:** `complete`
  - **VMware CI:** `complete`
  - **Hyper-V CI:** `complete`
  - **Ironic CI:** `missing`
  - **IBM PowerVM CI:** `complete`
  - **IBM zVM CI:** `missing`

- **Custom disk configurations on boot** This is about supporting all the features of BDMv2. This includes booting from a volume, in various ways, and specifying a custom set of ephemeral disks. Note some drivers only supports part of what the API allows.

  info:

  - **Maturity:** `complete`
  - **API Docs:** https://docs.openstack.org/api-ref/compute/?expanded=create-image-createimage-action-detail#create-server
  - **Admin Docs:** https://docs.openstack.org/nova/latest/user/block-device-mapping.html
  - **Tempest tests:** 557cd2c2-4eb8-4dce-98be-f86765ff311b, 36c34c67-7b54-4b59-b188-02a2f458a63b

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `complete`
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+virtuozzo CT:** `missing`
  - **libvirt+virtuozzo VM:** `complete`
  - **libvirt+xen:** `complete`
  - **XenServer CI:** `partial`
  - **VMware CI:** `partial`
  - **Hyper-V CI:** `complete` (updated in N release)
  - **Ironic CI:** `missing`
  - **IBM PowerVM CI:** `missing`
  - **IBM zVM CI:** `missing`

- **Custom neutron configurations on boot** This is about supporting booting from one or more neutron ports, or all the related short cuts such as booting a specified network. This does not include SR-IOV or similar, just simple neutron ports.

  info:

  - **Maturity:** `complete`
  - **API Docs:** https://docs.openstack.org/api-ref/compute/?&expanded=create-server-detail
  - **Admin Docs:**
  - **Tempest tests:** 2f3a0127-95c7-4977-92d2-bc5aec602fb4

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `complete`
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+virtuozzo CT:** `unknown`
  - **libvirt+virtuozzo VM:** `unknown`
  - **libvirt+xen:** `partial`
  - **XenServer CI:** `partial`
  - **VMware CI:** `partial`
  - **Hyper-V CI:** `partial`
  - **Ironic CI:** `missing`
  - **IBM PowerVM CI:** `complete`
  - **IBM zVM CI:** `partial`

- **Pause a Server** This is pause and unpause a server, where the state is held in memory.

  info:

  - **Maturity:** `complete`
  - **API Docs:** https://docs.openstack.org/api-ref/compute/?#pause-server-pause-action
  - **Admin Docs:**
  - **Tempest tests:** bd61a9fd-062f-4670-972b-2d6c3e3b9e73

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `complete`
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+virtuozzo CT:** `missing`
  - **libvirt+virtuozzo VM:** `partial`
  - **libvirt+xen:** `complete`
  - **XenServer CI:** `complete`
  - **VMware CI:** `partial`
  - **Hyper-V CI:** `complete`
  - **Ironic CI:** `missing`
  - **IBM PowerVM CI:** `missing`
  - **IBM zVM CI:** `complete`
- **Suspend a Server** This suspend and resume a server, where the state is held on disk.

  info:

  - **Maturity:** `complete`
  - **API Docs:** https://docs.openstack.org/api-ref/compute/?expanded=suspend-server-suspend-action-detail
  - **Admin Docs:**
  - **Tempest tests:** 0d8ee21e-b749-462d-83da-b85b41c86c7f

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `complete`
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+virtuozzo CT:** `partial`
  - **libvirt+virtuozzo VM:** `partial`
  - **libvirt+xen:** `complete`
  - **XenServer CI:** `complete`
  - **VMware CI:** `complete`
  - **Hyper-V CI:** `complete`
  - **Ironic CI:** `missing`
  - **IBM PowerVM CI:** `missing`
  - **IBM zVM CI:** `missing`

- **Server console output** This gets the current server console output.

  info:

  - **Maturity:** `complete`
  - **API Docs:** https://docs.openstack.org/api-ref/compute/#show-console-output-os-getconsoleoutput-action
  - **Admin Docs:**
  - **Tempest tests:** 4b8867e6-fffa-4d54-b1d1-6fdda57be2f3

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `complete`
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+virtuozzo CT:** `unknown`
  - **libvirt+virtuozzo VM:** `unknown`
  - **libvirt+xen:** `complete`
  - **XenServer CI:** `complete`
  - **VMware CI:** `partial`
  - **Hyper-V CI:** `partial`
  - **Ironic CI:** `missing`
  - **IBM PowerVM CI:** `complete`
  - **IBM zVM CI:** `complete`

- **Server Rescue** This boots a server with a new root disk from the specified glance image to allow a user to fix a boot partition configuration, or similar.

  info:

  - **Maturity:** `complete`
  - **API Docs:** https://docs.openstack.org/api-ref/compute/#rescue-server-rescue-action
  - **Admin Docs:**
  - **Tempest tests:** fd032140-714c-42e4-a8fd-adcd8df06be6, 70cdb8a1-89f8-437d-9448-8844fd82bf46

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `complete`
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+virtuozzo CT:** `partial`
  - **libvirt+virtuozzo VM:** `complete`
  - **libvirt+xen:** `complete`
  - **XenServer CI:** `complete`
  - **VMware CI:** `complete`
  - **Hyper-V CI:** `partial`
  - **Ironic CI:** `missing`
  - **IBM PowerVM CI:** `missing`

  - **IBM zVM CI:** missing

- **Server Config Drive** This ensures the user data provided by the user when booting a server is available in one of the expected config drive locations.

  info:

  - **Maturity:** complete
  - **API Docs:** https://docs.openstack.org/api-ref/compute/#create-server
  - **Admin Docs:** https://docs.openstack.org/nova/latest/admin/config-drive.html
  - **Tempest tests:** 7fff3fb3-91d8-4fd0-bd7d-0204f1f180ba

  drivers:

  - **libvirt+kvm (x86 & ppc64):** complete
  - **libvirt+kvm (s390x):** unknown
  - **libvirt+virtuozzo CT:** missing
  - **libvirt+virtuozzo VM:** partial
  - **libvirt+xen:** complete
  - **XenServer CI:** complete
  - **VMware CI:** complete
  - **Hyper-V CI:** complete
  - **Ironic CI:** partial
  - **IBM PowerVM CI:** complete
  - **IBM zVM CI:** complete

- **Server Change Password** The ability to reset the password of a user within the server.

  info:

  - **Maturity:** experimental
  - **API Docs:** https://docs.openstack.org/api-ref/compute/#change-administrative-password-changepassword-action
  - **Admin Docs:**
  - **Tempest tests:** 6158df09-4b82-4ab3-af6d-29cf36af858d

  drivers:

  - **libvirt+kvm (x86 & ppc64):** partial
  - **libvirt+kvm (s390x):** unknown
  - **libvirt+virtuozzo CT:** missing
  - **libvirt+virtuozzo VM:** missing
  - **libvirt+xen:** missing
  - **XenServer CI:** partial
  - **VMware CI:** missing
  - **Hyper-V CI:** partial
  - **Ironic CI:** missing

  - **IBM PowerVM CI:** missing
  - **IBM zVM CI:** missing

- **Server Shelve and Unshelve** The ability to keep a server logically alive, but not using any cloud resources. For local disk based instances, this involves taking a snapshot, called offloading.

  info:

  - **Maturity:** complete
  - **API Docs:** https://docs.openstack.org/api-ref/compute/#shelve-server-shelve-action
  - **Admin Docs:**
  - **Tempest tests:** 1164e700-0af0-4a4c-8792-35909a88743c,c1b6318c-b9da-490b-9c67-9339b627271f

  drivers:

  - **libvirt+kvm (x86 & ppc64):** complete
  - **libvirt+kvm (s390x):** unknown
  - **libvirt+virtuozzo CT:** missing
  - **libvirt+virtuozzo VM:** complete
  - **libvirt+xen:** complete
  - **XenServer CI:** complete
  - **VMware CI:** missing
  - **Hyper-V CI:** complete
  - **Ironic CI:** missing
  - **IBM PowerVM CI:** complete
  - **IBM zVM CI:** missing

4.51.3 NFV Cloud Features

Network Function Virtualization (NFV) is about virtualizing network node functions into building blocks that may connect, or chain together to create a particular service. It is common for this workloads needing bare metal like performance, i.e. low latency and close to line speed performance.

Important: In deployments older than Train, or in mixed Stein/Train deployments with a rolling upgrade in progress, unless *specifically enabled*, live migration is not possible for instances with a NUMA topology when using the libvirt driver. A NUMA topology may be specified explicitly or can be added implicitly due to the use of CPU pinning or huge pages. Refer to bug #1289064 for more information. As of Train, live migration of instances with a NUMA topology when using the libvirt driver is fully supported.

Summary

Feature	*Maturity*	**libvirt+kvm (x86 & ppc64)**	**libvirt+kvm (s390x)**	**libvirt+xen**
NUMA Placement	experimental	✓	*?*	
CPU Pinning Policy	experimental	✓	*?*	
CPU Pinning Thread Policy	experimental	✓	*?*	

Details

- **NUMA Placement** Configure placement of instance vCPUs and memory across host NUMA nodes

  info:

  - **Maturity:** `experimental`
  - **API Docs:** https://docs.openstack.org/api-ref/compute/#create-server
  - **Admin Docs:** https://docs.openstack.org/nova/latest/admin/cpu-topologies.html#customizing-instance-cpu-pinning-policies
  - **Tempest tests:** 9a438d88-10c6-4bcd-8b5b-5b6e25e1346f, 585e934c-448e-43c4-acbf-d06a9b899997

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `partial`
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+xen:** `missing`

- **CPU Pinning Policy** Enable/disable binding of instance vCPUs to host CPUs

  info:

  - **Maturity:** `experimental`
  - **API Docs:** https://docs.openstack.org/api-ref/compute/#create-server
  - **Admin Docs:** https://docs.openstack.org/nova/latest/admin/cpu-topologies.html#customizing-instance-cpu-pinning-policies
  - **Tempest tests:**

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `partial`
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+xen:** `missing`

- **CPU Pinning Thread Policy** Configure usage of host hardware threads when pinning is used

  info:

  - **Maturity:** `experimental`
  - **API Docs:** https://docs.openstack.org/api-ref/compute/#create-server
  - **Admin Docs:** https://docs.openstack.org/nova/latest/admin/cpu-topologies.html#customizing-instance-cpu-pinning-policies
  - **Tempest tests:**

drivers:

- **libvirt+kvm (x86 & ppc64):** `partial`
- **libvirt+kvm (s390x):** `unknown`
- **libvirt+xen:** `missing`

4.51.4 HPC Cloud Features

High Performance Compute (HPC) cloud have some specific needs that are covered in this set of features. Summary

Feature	*Maturity*	Hyper-V CI	Ironic	libvirt+kvm (x86 & ppc64)	libvirt+kvm (s390x)	libvirt+virtuozzo CT	libvirt+virtuozzo VM	libvirt+xen	PowerVM CI	VMware CI	XenServer CI
GPU Passthrough	experimental		*?*	✓ *l*	*?*	✓	✓				✓ *k*
Virtual GPUs	experimental			✓ *queens*	*?*	*?*	*?*	*?*			✓ *queens*

Details

- **GPU Passthrough** The PCI passthrough feature in OpenStack allows full access and direct control of a physical PCI device in guests. This mechanism is generic for any devices that can be attached to a PCI bus. Correct driver installation is the only requirement for the guest to properly use the devices.

  info:

  - **Maturity:** `experimental`
  - **API Docs:** https://docs.openstack.org/api-ref/compute/#create-server
  - **Admin Docs:** https://docs.openstack.org/nova/latest/admin/pci-passthrough.html
  - **Tempest tests:** 9a438d88-10c6-4bcd-8b5b-5b6e25e1346f, 585e934c-448e-43c4-acbf-d06a9b899997

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `complete` (updated in L release)
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+virtuozzo CT:** `partial`
  - **libvirt+virtuozzo VM:** `partial`
  - **libvirt+xen:** `missing`
  - **XenServer CI:** `partial` (updated in K release)
  - **VMware CI:** `missing`
  - **Hyper-V CI:** `missing`
  - **Ironic:** `unknown`
  - **PowerVM CI:** `missing`

- **Virtual GPUs** Attach a virtual GPU to an instance at server creation time

  info:

  - **Maturity:** `experimental`
  - **API Docs:** https://docs.openstack.org/api-ref/compute/#create-server
  - **Admin Docs:** https://docs.openstack.org/nova/latest/admin/virtual-gpu.html
  - **Tempest tests:**

  drivers:

  - **libvirt+kvm (x86 & ppc64):** `partial` (updated in QUEENS release)
  - **libvirt+kvm (s390x):** `unknown`
  - **libvirt+virtuozzo CT:** `unknown`
  - **libvirt+virtuozzo VM:** `unknown`
  - **libvirt+xen:** `unknown`
  - **XenServer CI:** `partial` (updated in QUEENS release)
  - **VMware CI:** `missing`
  - **Hyper-V CI:** `missing`
  - **Ironic:** `missing`
  - **PowerVM CI:** `missing`

4.51.5 Notes on Concepts

This document uses the following terminology.

Users

These are the users we talk about in this document:

application deployer creates and deletes servers, directly or indirectly using an API

application developer creates images and apps that run on the cloud

cloud operator administers the cloud

self service administrator runs and uses the cloud

Note: This is not an exhaustive list of personas, but rather an indicative set of users.

Feature Group

To reduce the size of the matrix, we organize the features into groups. Each group maps to a set of user stories that can be validated by a set of scenarios and tests. Typically, this means a set of tempest tests.

This list focuses on API concepts like attach and detach volumes, rather than deployment specific concepts like attach an iSCSI volume to a KVM based VM.

Deployment

A deployment maps to a specific test environment. We provide a full description of the environment, so it is possible to reproduce the reported test results for each of the Feature Groups.

This description includes all aspects of the deployment, for example the hypervisor, number of nova-compute services, storage, network driver, and types of images being tested.

Feature Group Maturity

The Feature Group Maturity rating is specific to the API concepts, rather than specific to a particular deployment. That detail is covered in the deployment rating for each feature group.

Note: Although having some similarities, this list is not directly related to the DefCore effort.

Feature Group ratings:

Incomplete Incomplete features are those that do not have enough functionality to satisfy real world use cases.

Experimental Experimental features should be used with extreme caution. They are likely to have little or no upstream testing, and are therefore likely to contain bugs.

Complete For a feature to be considered complete, it must have:

- complete API docs (concept and REST call definition)
- complete Administrator docs
- tempest tests that define if the feature works correctly
- sufficient functionality and reliability to be useful in real world scenarios
- a reasonable expectation that the feature will be supported long-term

Complete and Required There are various reasons why a complete feature may be required, but generally it is when all drivers support that feature. New drivers need to prove they support all required features before they are allowed in upstream Nova.

Required features are those that any new technology must support before being allowed into tree. The larger the list, the more features are available on all Nova based clouds.

Deprecated Deprecated features are those that are scheduled to be removed in a future major release of Nova. If a feature is marked as complete, it should never be deprecated.

If a feature is incomplete or experimental for several releases, it runs the risk of being deprecated and later removed from the code base.

Deployment Rating for a Feature Group

The deployment rating refers to the state of the tests for each Feature Group on a particular deployment.

Deployment ratings:

Unknown No data is available.

Not Implemented No tests exist.

Implemented Self declared that the tempest tests pass.

Regularly Tested Tested by third party CI.

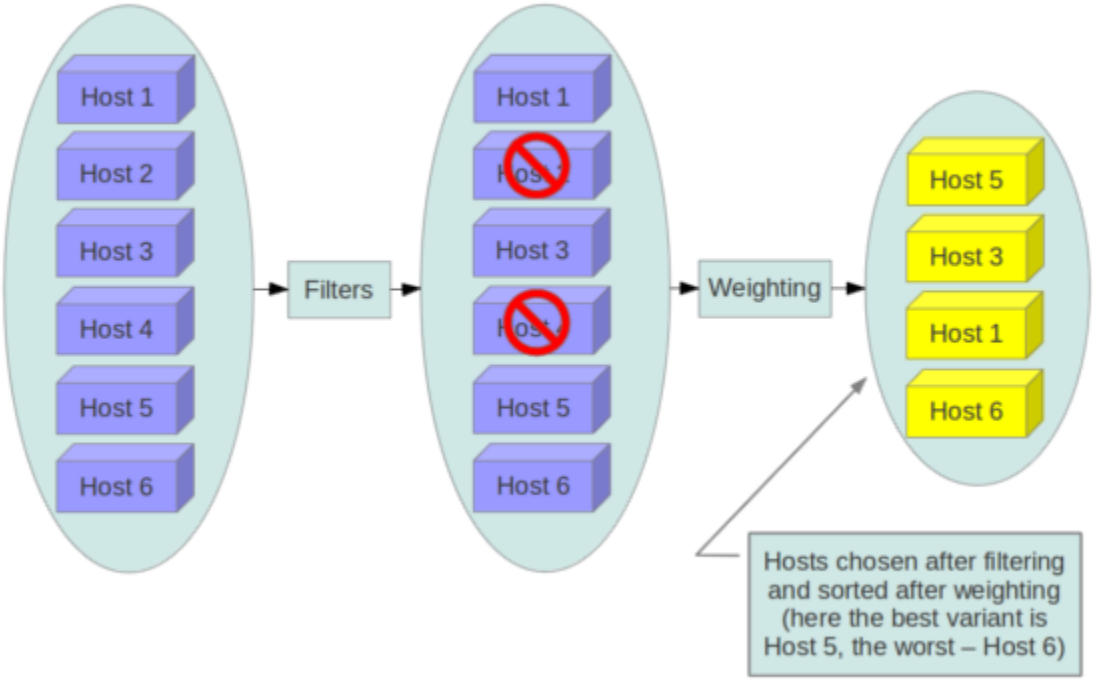
Checked Tested as part of the check or gate queue.

The eventual goal is to automate this list from a third party CI reporting system, but currently we document manual inspections in an ini file. Ideally, we will review the list at every milestone.

4.52 Filter Scheduler

The **Filter Scheduler** supports *filtering* and *weighting* to make informed decisions on where a new instance should be created. This Scheduler supports working with Compute Nodes only.

4.52.1 Filtering



During its work Filter Scheduler iterates over all found compute nodes, evaluating each against a set of filters. The list of resulting hosts is ordered by weighers. The Scheduler then chooses hosts for the requested number of instances, choosing the most weighted hosts. For a specific filter to succeed for a specific host, the filter matches the user request against the state of the host plus some extra magic as defined by each filter (described in more detail below).

If the Scheduler cannot find candidates for the next instance, it means that there are no appropriate hosts where that instance can be scheduled.

The Filter Scheduler has to be quite flexible to support the required variety of *filtering* and *weighting* strategies. If this flexibility is insufficient you can implement *your own filtering algorithm*.

There are many standard filter classes which may be used (`nova.scheduler.filters`):

- `AllHostsFilter` - does no filtering. It passes all the available hosts.

- `ImagePropertiesFilter` - filters hosts based on properties defined on the instances image. It passes hosts that can support the properties specified on the image used by the instance.
- `AvailabilityZoneFilter` - filters hosts by availability zone. It passes hosts matching the availability zone specified in the instance properties. Use a comma to specify multiple zones. The filter will then ensure it matches any zone specified.
- `ComputeCapabilitiesFilter` - checks that the capabilities provided by the host compute service satisfy any extra specifications associated with the instance type. It passes hosts that can create the specified instance type.

  If an extra specs key contains a colon (:), anything before the colon is treated as a namespace and anything after the colon is treated as the key to be matched. If a namespace is present and is not `capabilities`, the filter ignores the namespace. For example `capabilities:cpu_info:features` is a valid scope format. For backward compatibility, when a key doesnt contain a colon (:), the keys contents are important. If this key is an attribute of HostState object, like `free_disk_mb`, the filter also treats the extra specs key as the key to be matched. If not, the filter will ignore the key.

  The extra specifications can have an operator at the beginning of the value string of a key/value pair. If there is no operator specified, then a default operator of `s==` is used. Valid operators are:

```
* = (equal to or greater than as a number; same as vcpus case)
* == (equal to as a number)
* != (not equal to as a number)
* >= (greater than or equal to as a number)
* <= (less than or equal to as a number)
* s== (equal to as a string)
* s!= (not equal to as a string)
* s>= (greater than or equal to as a string)
* s> (greater than as a string)
* s<= (less than or equal to as a string)
* s< (less than as a string)
* <in> (substring)
* <all-in> (all elements contained in collection)
* <or> (find one of these)

Examples are: ">= 5", "s== 2.1.0", "<in> gcc", "<all-in> aes mmx", and "<or> fpu
↪<or> gpu"
```

  some of attributes that can be used as useful key and their values contains:

```
* free_ram_mb (compared with a number, values like ">= 4096")
* free_disk_mb (compared with a number, values like ">= 10240")
* host (compared with a string, values like: "<in> compute","s== compute_01")
* hypervisor_type (compared with a string, values like: "s== QEMU", "s== powervm")
* hypervisor_version (compared with a number, values like : ">= 1005003", "==␣
↪2000000")
* num_instances (compared with a number, values like: "<= 10")
* num_io_ops (compared with a number, values like: "<= 5")
* vcpus_total (compared with a number, values like: "= 48", ">=24")
* vcpus_used (compared with a number, values like: "= 0", "<= 10")
```

- `AggregateInstanceExtraSpecsFilter` - checks that the aggregate metadata satisfies any extra specifications associated with the instance type (that have no scope or are scoped with `aggregate_instance_extra_specs`). It passes hosts that can create the specified instance type. The extra specifications can have the same operators as `ComputeCapabilitiesFilter`. To specify multiple values for the same key use a comma. E.g., value1,value2. All hosts are passed if no extra_specs are specified.
- `ComputeFilter` - passes all hosts that are operational and enabled.

- `AggregateCoreFilter` - DEPRECATED; filters hosts by CPU core number with per-aggregate *`cpu_allocation_ratio`* setting. If no per-aggregate value is found, it will fall back to the global default *`cpu_allocation_ratio`*. If more than one value is found for a host (meaning the host is in two different aggregates with different ratio settings), the minimum value will be used.
- `IsolatedHostsFilter` - filter based on *`filter_scheduler.isolated_images`*, *`filter_scheduler.isolated_hosts`* and *`filter_scheduler.restrict_isolated_hosts_to_isolated_images`* flags.
- `JsonFilter` - allows simple JSON-based grammar for selecting hosts.
- `AggregateRamFilter` - DEPRECATED; filters hosts by RAM with per-aggregate *`ram_allocation_ratio`* setting. If no per-aggregate value is found, it will fall back to the global default *`ram_allocation_ratio`*. If more than one value is found for a host (meaning the host is in two different aggregates with different ratio settings), the minimum value will be used.
- `AggregateDiskFilter` - DEPRECATED; filters hosts by disk allocation with per-aggregate *`disk_allocation_ratio`* setting. If no per-aggregate value is found, it will fall back to the global default *`disk_allocation_ratio`*. If more than one value is found for a host (meaning the host is in two or more different aggregates with different ratio settings), the minimum value will be used.
- `NumInstancesFilter` - filters compute nodes by number of instances. Nodes with too many instances will be filtered. The host will be ignored by the scheduler if more than *`filter_scheduler.max_instances_per_host`* already exist on the host.
- `AggregateNumInstancesFilter` - filters hosts by number of instances with per-aggregate *`filter_scheduler.max_instances_per_host`* setting. If no per-aggregate value is found, it will fall back to the global default *`filter_scheduler.max_instances_per_host`*. If more than one value is found for a host (meaning the host is in two or more different aggregates with different max instances per host settings), the minimum value will be used.
- `IoOpsFilter` - filters hosts by concurrent I/O operations on it. hosts with too many concurrent I/O operations will be filtered. *`filter_scheduler.max_io_ops_per_host`* setting. Maximum number of I/O intensive instances allowed to run on this host, the host will be ignored by scheduler if more than *`filter_scheduler.max_io_ops_per_host`* instances such as build/resize/snapshot etc are running on it.
- `AggregateIoOpsFilter` - filters hosts by I/O operations with per-aggregate *`filter_scheduler.max_io_ops_per_host`* setting. If no per-aggregate value is found, it will fall back to the global default *:oslo.config:option:'filter_scheduler.max_io_ops_per_host*. If more than one value is found for a host (meaning the host is in two or more different aggregates with different max io operations settings), the minimum value will be used.
- `PciPassthroughFilter` - Filter that schedules instances on a host if the host has devices to meet the device requests in the extra_specs for the flavor.
- `SimpleCIDRAffinityFilter` - allows a new instance on a host within the same IP block.
- `DifferentHostFilter` - allows the instance on a different host from a set of instances.
- `SameHostFilter` - puts the instance on the same host as another instance in a set of instances.
- `RetryFilter` - DEPRECATED; filters hosts that have been attempted for scheduling. Only passes hosts that have not been previously attempted.
- **`AggregateTypeAffinityFilter` - limits instance_type by aggregate.** This filter passes hosts if no instance_type key is set or the instance_type aggregate metadata value contains the name of the instance_type requested. The value of the instance_type metadata entry is a string that may contain either a single instance_type name or a comma separated list of instance_type names. e.g. m1.nano or m1.nano,m1.small

- `ServerGroupAntiAffinityFilter` - This filter implements anti-affinity for a server group. First you must create a server group with a policy of anti-affinity via the server groups API. Then, when you boot a new server, provide a scheduler hint of group=<uuid> where <uuid> is the UUID of the server group you created. This will result in the server getting added to the group. When the server gets scheduled, anti-affinity will be enforced among all servers in that group.
- `ServerGroupAffinityFilter` - This filter works the same way as ServerGroupAntiAffinityFilter. The difference is that when you create the server group, you should specify a policy of affinity.
- `AggregateMultiTenancyIsolation` - isolate tenants in specific aggregates. To specify multiple tenants use a comma. Eg. tenant1,tenant2
- `AggregateImagePropertiesIsolation` - isolates hosts based on image properties and aggregate metadata. Use a comma to specify multiple values for the same property. The filter will then ensure at least one value matches.
- `MetricsFilter` - filters hosts based on metrics weight_setting. Only hosts with the available metrics are passed.
- `NUMATopologyFilter` - filters hosts based on the NUMA topology requested by the instance, if any.

Now we can focus on these standard filter classes in some detail. Some filters such as `AllHostsFilter` and `NumInstancesFilter` are relatively simple and can be understood from the code. For example, `NumInstancesFilter` has the following implementation:

```
class NumInstancesFilter(filters.BaseHostFilter):
    """Filter out hosts with too many instances."""

    def _get_max_instances_per_host(self, host_state, spec_obj):
        return CONF.filter_scheduler.max_instances_per_host

    def host_passes(self, host_state, spec_obj):
        num_instances = host_state.num_instances
        max_instances = self._get_max_instances_per_host(host_state, spec_obj)
        passes = num_instances < max_instances
        return passes
```

Here *`filter_scheduler.max_instances_per_host`* means the maximum number of instances that can be on a host.

The `AvailabilityZoneFilter` looks at the availability zone of compute node and availability zone from the properties of the request. Each compute service has its own availability zone. So deployment engineers have an option to run scheduler with availability zones support and can configure availability zones on each compute host. This classs method `host_passes` returns `True` if availability zone mentioned in request is the same on the current compute host.

The `ImagePropertiesFilter` filters hosts based on the architecture, hypervisor type and virtual machine mode specified in the instance. For example, an instance might require a host that supports the ARM architecture on a qemu compute host. The `ImagePropertiesFilter` will only pass hosts that can satisfy this request. These instance properties are populated from properties defined on the instances image. E.g. an image can be decorated with these properties using `glance image-update img-uuid --property architecture=arm --property hypervisor_type=qemu` Only hosts that satisfy these requirements will pass the `ImagePropertiesFilter`.

`ComputeCapabilitiesFilter` checks if the host satisfies any `extra_specs` specified on the instance type. The `extra_specs` can contain key/value pairs. The key for the filter is either non-scope format (i.e. no `:` contained), or scope format in capabilities scope (i.e. `capabilities:xxx:yyy`). One example of capabilities scope is `capabilities:cpu_info:features`, which will match hosts cpu features capabilities. The `ComputeCapabilitiesFilter` will only pass hosts whose capabilities satisfy the requested specifications. All hosts are passed if no `extra_specs` are specified.

`ComputeFilter` is quite simple and passes any host whose compute service is enabled and operational.

Now we are going to `IsolatedHostsFilter`. There can be some special hosts reserved for specific images. These hosts are called **isolated**. So the images to run on the isolated hosts are also called isolated. The filter checks if *`filter_scheduler.isolated_images`* flag named in instance specifications is the same as the host specified in *`filter_scheduler.isolated_hosts`*. Isolated hosts can run non-isolated images if the flag *`filter_scheduler.restrict_isolated_hosts_to_isolated_images`* is set to false.

`DifferentHostFilter` - method `host_passes` returns `True` if the host to place an instance on is different from all the hosts used by a set of instances.

`SameHostFilter` does the opposite to what `DifferentHostFilter` does. `host_passes` returns `True` if the host we want to place an instance on is one of the hosts used by a set of instances.

`SimpleCIDRAffinityFilter` looks at the subnet mask and investigates if the network address of the current host is in the same sub network as it was defined in the request.

`JsonFilter` - this filter provides the opportunity to write complicated queries for the hosts capabilities filtering, based on simple JSON-like syntax. There can be used the following operations for the host states properties: =, <, >, `in`, <=, >=, that can be combined with the following logical operations: `not`, `or`, `and`. For example, the following query can be found in tests:

```
['and',
    ['>=', '$free_ram_mb', 1024],
    ['>=', '$free_disk_mb', 200 * 1024]
]
```

This query will filter all hosts with free RAM greater or equal than 1024 MB and at the same time with free disk space greater or equal than 200 GB.

Many filters use data from `scheduler_hints`, that is defined in the moment of creation of the new server for the user. The only exception for this rule is `JsonFilter`, that takes data from the schedulers `HostState` data structure directly. Variable naming, such as the `$free_ram_mb` example above, should be based on those attributes.

The `RetryFilter` filters hosts that have already been attempted for scheduling. It only passes hosts that have not been previously attempted. If a compute node is raising an exception when spawning an instance, then the compute manager will reschedule it by adding the failing host to a retry dictionary so that the RetryFilter will not accept it as a possible destination. That means that if all of your compute nodes are failing, then the RetryFilter will return 0 hosts and the scheduler will raise a NoValidHost exception even if the problem is related to 1:N compute nodes. If you see that case in the scheduler logs, then your problem is most likely related to a compute problem and you should check the compute logs.

Note: The `RetryFilter` is deprecated since the 20.0.0 (Train) release and will be removed in an upcoming release. Since the 17.0.0 (Queens) release, the scheduler has provided alternate hosts for rescheduling so the scheduler does not need to be called during a reschedule which makes the `RetryFilter` useless. See the Return Alternate Hosts spec for details.

The `NUMATopologyFilter` considers the NUMA topology that was specified for the instance through the use of flavor extra_specs in combination with the image properties, as described in detail in the related nova-spec document:

- https://opendev.org/openstack/nova-specs/src/branch/master/specs/juno/implemented/virt-driver-numa-placement.rst

and try to match it with the topology exposed by the host, accounting for the *`ram_allocation_ratio`* and *`cpu_allocation_ratio`* for over-subscription. The filtering is done in the following manner:

- Filter will attempt to pack instance cells onto host cells.

- It will consider the standard over-subscription limits for each host NUMA cell, and provide limits to the compute host accordingly (as mentioned above).
- If instance has no topology defined, it will be considered for any host.
- If instance has a topology defined, it will be considered only for NUMA capable hosts.

4.52.2 Configuring Filters

To use filters you specify two settings:

- *filter_scheduler.available_filters* - Defines filter classes made available to the scheduler. This setting can be used multiple times.
- *filter_scheduler.enabled_filters* - Of the available filters, defines those that the scheduler uses by default.

The default values for these settings in nova.conf are:

```
--filter_scheduler.available_filters=nova.scheduler.filters.all_filters
--filter_scheduler.enabled_filters=ComputeFilter,AvailabilityZoneFilter,
↪ComputeCapabilitiesFilter,ImagePropertiesFilter,ServerGroupAntiAffinityFilter,
↪ServerGroupAffinityFilter
```

With this configuration, all filters in `nova.scheduler.filters` would be available, and by default the `ComputeFilter`, `AvailabilityZoneFilter`, `ComputeCapabilitiesFilter`, `ImagePropertiesFilter`, `ServerGroupAntiAffinityFilter`, and `ServerGroupAffinityFilter` would be used.

Each filter selects hosts in a different way and has different costs. The order of *filter_scheduler.enabled_filters* affects scheduling performance. The general suggestion is to filter out invalid hosts as soon as possible to avoid unnecessary costs. We can sort *filter_scheduler.enabled_filters* items by their costs in reverse order. For example, `ComputeFilter` is better before any resource calculating filters like `NUMATopologyFilter`.

In medium/large environments having AvailabilityZoneFilter before any capability or resource calculating filters can be useful.

4.52.3 Writing Your Own Filter

To create **your own filter** you must inherit from `BaseHostFilter` and implement one method: `host_passes`. This method should return `True` if a host passes the filter and return `False` elsewhere. It takes two parameters (named arbitrarily as `host_state` and `spec_obj`):

- the `HostState` object allows to get attributes of the host.
- the `RequestSpec` object describes the user request, including the flavor, the image and the scheduler hints.

For further details about each of those objects and their corresponding attributes, please refer to the codebase (at least by looking at the other filters code) or ask for help in the #openstack-nova IRC channel.

As an example, nova.conf could contain the following scheduler-related settings:

```
--scheduler.driver=nova.scheduler.FilterScheduler
--filter_scheduler.available_filters=nova.scheduler.filters.all_filters
--filter_scheduler.available_filters=myfilter.MyFilter
--filter_scheduler.enabled_filters=ComputeFilter,MyFilter
```

Note: When writing your own filter, be sure to add it to the list of available filters and enable it in the default filters. The all_filters setting only includes the filters shipped with nova.

With these settings, nova will use the `FilterScheduler` for the scheduler driver. All of the standard nova filters and MyFilter are available to the FilterScheduler, but just the `ComputeFilter` and `MyFilter` will be used on each request.

4.52.4 Weights

Filter Scheduler uses the so-called **weights** during its work. A weigher is a way to select the best suitable host from a group of valid hosts by giving weights to all the hosts in the list.

In order to prioritize one weigher against another, all the weighers have to define a multiplier that will be applied before computing the weight for a node. All the weights are normalized beforehand so that the multiplier can be applied easily. Therefore the final weight for the object will be:

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

A weigher should be a subclass of `weights.BaseHostWeigher` and they can implement both the `weight_multiplier` and `_weight_object` methods or just implement the `weight_objects` method. `weight_objects` method is overridden only if you need access to all objects in order to calculate weights, and it just return a list of weights, and not modify the weight of the object directly, since final weights are normalized and computed by `weight.BaseWeightHandler`.

The Filter Scheduler weighs hosts based on the config option *filter_scheduler.weight_classes*, this defaults to *nova.scheduler.weights.all_weighers*, which selects the following weighers:

- `RAMWeigher` Compute weight based on available RAM on the compute node. Sort with the largest weight winning. If the multiplier, *filter_scheduler.ram_weight_multiplier*, is negative, the host with least RAM available will win (useful for stacking hosts, instead of spreading). Starting with the Stein release, if per-aggregate value with the key *filter_scheduler.ram_weight_multiplier* is found, this value would be chosen as the ram weight multiplier. Otherwise, it will fall back to the *filter_scheduler.ram_weight_multiplier*. If more than one value is found for a host in aggregate metadata, the minimum value will be used.
- `CPUWeigher` Compute weight based on available vCPUs on the compute node. Sort with the largest weight winning. If the multiplier, *filter_scheduler.cpu_weight_multiplier*, is negative, the host with least CPUs available will win (useful for stacking hosts, instead of spreading). Starting with the Stein release, if per-aggregate value with the key *filter_scheduler.cpu_weight_multiplier* is found, this value would be chosen as the cpu weight multiplier. Otherwise, it will fall back to the *filter_scheduler.cpu_weight_multiplier*. If more than one value is found for a host in aggregate metadata, the minimum value will be used.
- `DiskWeigher` Hosts are weighted and sorted by free disk space with the largest weight winning. If the multiplier is negative, the host with less disk space available will win (useful for stacking hosts, instead of spreading). Starting with the Stein release, if per-aggregate value with the key *filter_scheduler.disk_weight_multiplier* is found, this value would be chosen as the disk weight multiplier. Otherwise, it will fall back to the *filter_scheduler.disk_weight_multiplier*. If more than one value is found for a host in aggregate metadata, the minimum value will be used.
- `MetricsWeigher` This weigher can compute the weight based on the compute node hosts various metrics. The to-be weighed metrics and their weighing ratio are specified in the configuration file as the followings:

```
metrics_weight_setting = name1=1.0, name2=-1.0
```

Starting with the Stein release, if per-aggregate value with the key *metrics_weight_multiplier* is found, this value would be chosen as the metrics weight multiplier. Otherwise, it will fall back to the *metrics.weight_multiplier*. If more than one value is found for a host in aggregate metadata, the minimum value will be used.

- `IoOpsWeigher` The weigher can compute the weight based on the compute node hosts workload. The default is to preferably choose light workload compute hosts. If the multiplier is positive, the weigher prefer choosing heavy workload compute hosts, the weighing has the opposite effect of the default. Starting with the Stein release, if per-aggregate value with the key *filter_scheduler.io_ops_weight_multiplier* is found, this value would be chosen as the IO ops weight multiplier. Otherwise, it will fall back to the *filter_scheduler.io_ops_weight_multiplier*. If more than one value is found for a host in aggregate metadata, the minimum value will be used.

- `PCIWeigher` Compute a weighting based on the number of PCI devices on the host and the number of PCI devices requested by the instance. For example, given three hosts - one with a single PCI device, one with many PCI devices, and one with no PCI devices - nova should prioritise these differently based on the demands of the instance. If the instance requests a single PCI device, then the first of the hosts should be preferred. Similarly, if the instance requests multiple PCI devices, then the second of these hosts would be preferred. Finally, if the instance does not request a PCI device, then the last of these hosts should be preferred.

  For this to be of any value, at least one of the `PciPassthroughFilter` or `NUMATopologyFilter` filters must be enabled.

  Configuration Option `[filter_scheduler] pci_weight_multiplier`. Only positive values are allowed for the multiplier as a negative value would force non-PCI instances away from non-PCI hosts, thus, causing future scheduling issues.

  Starting with the Stein release, if per-aggregate value with the key *filter_scheduler.pci_weight_multiplier* is found, this value would be chosen as the pci weight multiplier. Otherwise, it will fall back to the *filter_scheduler.pci_weight_multiplier*. If more than one value is found for a host in aggregate metadata, the minimum value will be used.

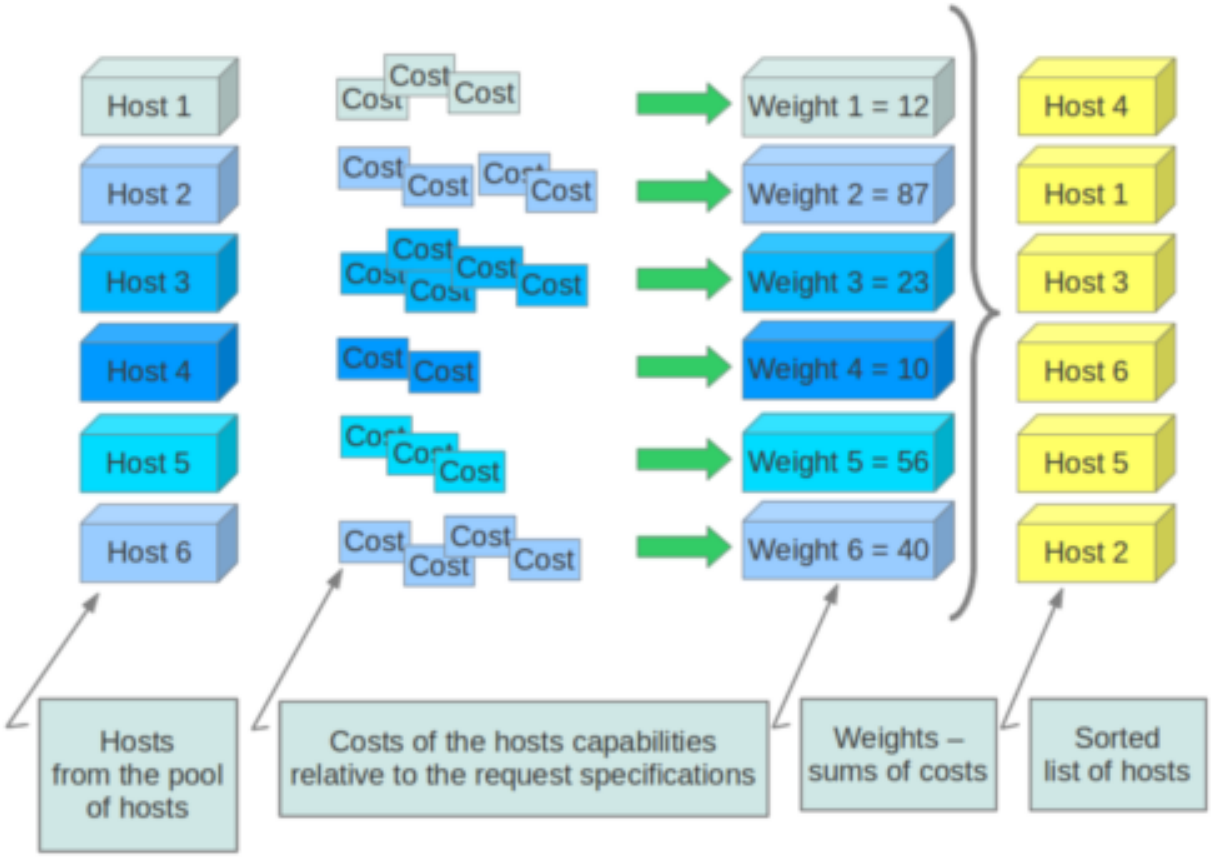
- `ServerGroupSoftAffinityWeigher` The weigher can compute the weight based on the number of instances that run on the same server group. The largest weight defines the preferred host for the new instance. For the multiplier only a positive value is allowed for the calculation. Starting with the Stein release, if per-aggregate value with the key *filter_scheduler.soft_affinity_weight_multiplier* is found, this value would be chosen as the soft affinity weight multiplier. Otherwise, it will fall back to the *filter_scheduler.soft_affinity_weight_multiplier*. If more than one value is found for a host in aggregate metadata, the minimum value will be used.

- `ServerGroupSoftAntiAffinityWeigher` The weigher can compute the weight based on the number of instances that run on the same server group as a negative value. The largest weight defines the preferred host for the new instance. For the multiplier only a positive value is allowed for the calculation. Starting with the Stein release, if per-aggregate value with the key *filter_scheduler.soft_anti_affinity_weight_multiplier* is found, this value would be chosen as the soft anti-affinity weight multiplier. Otherwise, it will fall back to the *filter_scheduler.soft_anti_affinity_weight_multiplier*. If more than one value is found for a host in aggregate metadata, the minimum value will be used.

- `BuildFailureWeigher` Weigh hosts by the number of recent failed boot attempts. It considers the build failure counter and can negatively weigh hosts with recent failures. This avoids taking computes fully out of rotation. Starting with the Stein release, if per-aggregate value with the key *filter_scheduler.build_failure_weight_multiplier* is found, this value would be chosen as the build failure weight multiplier. Otherwise, it will fall back to the *filter_scheduler.build_failure_weight_multiplier*. If more than one value is found for a host in aggregate metadata, the minimum value will be used.

Filter Scheduler makes a local list of acceptable hosts by repeated filtering and weighing. Each time it chooses a host,

it virtually consumes resources on it, so subsequent selections can adjust accordingly. It is useful if the customer asks for a large block of instances, because weight is computed for each instance requested.



At the end Filter Scheduler sorts selected hosts by their weight and attempts to provision instances on the chosen hosts.

P.S.: you can find more examples of using Filter Scheduler and standard filters in `nova.tests.scheduler`.

4.53 Flavors

In OpenStack, flavors define the compute, memory, and storage capacity of nova computing instances. To put it simply, a flavor is an available hardware configuration for a server. It defines the *size* of a virtual server that can be launched.

Note: Flavors can also determine on which compute host a flavor can be used to launch an instance. For information about customizing flavors, refer to *Manage Flavors*.

4.53.1 Overview

A flavor consists of the following parameters:

Flavor ID Unique ID (integer or UUID) for the new flavor. This property is required. If specifying auto, a UUID will be automatically generated.

Name Name for the new flavor. This property is required.

Historically, names were given a format *XX.SIZE_NAME*. These are typically not required, though some third party tools may rely on it.

VCPUs Number of virtual CPUs to use. This property is required.

Memory MB Amount of RAM to use (in megabytes). This property is required.

Root Disk GB Amount of disk space (in gigabytes) to use for the root (`/`) partition. This property is required.

The root disk is an ephemeral disk that the base image is copied into. When booting from a persistent volume it is not used. The `0` size is a special case which uses the native base image size as the size of the ephemeral root volume. However, in this case the filter scheduler cannot select the compute host based on the virtual image size. As a result, `0` should only be used for volume booted instances or for testing purposes. Volume-backed instances can be enforced for flavors with zero root disk via the `os_compute_api:servers:create:zero_disk_flavor` policy rule.

Ephemeral Disk GB Amount of disk space (in gigabytes) to use for the ephemeral partition. This property is optional. If unspecified, the value is `0` by default.

Ephemeral disks offer machine local disk storage linked to the lifecycle of a VM instance. When a VM is terminated, all data on the ephemeral disk is lost. Ephemeral disks are not included in any snapshots.

Swap Amount of swap space (in megabytes) to use. This property is optional. If unspecified, the value is `0` by default.

RXTX Factor The receive/transmit factor of any network ports on the instance. This property is optional. If unspecified, the value is `1.0` by default.

Note: This property only applies if using the `xen` compute driver with the `nova-network` network driver. It will likely be deprecated in a future release. `neutron` users should refer to the neutron QoS documentation

---

Is Public Boolean value that defines whether the flavor is available to all users or private to the project it was created in. This property is optional. In unspecified, the value is `True` by default.

By default, a flavor is public and available to all projects. Private flavors are only accessible to those on the access list for a given project and are invisible to other projects.

Extra Specs Key and value pairs that define on which compute nodes a flavor can run. These are optional.

Extra specs are generally used as scheduler hints for more advanced instance configuration. The key-value pairs used must correspond to well-known options. For more information on the standardized extra specs available, *see below*

Description A free form description of the flavor. Limited to 65535 characters in length. Only printable characters are allowed. Available starting in microversion 2.55.

Extra Specs

Todo: A lot of these need investigation - for example, I can find no reference to the `cpu_shares_level` option outside of documentation and (possibly) useless tests. We should assess which drivers each option actually apply to.

CPU limits You can configure the CPU limits with control parameters. For example, to configure the I/O limit, use:

```
$ openstack flavor set FLAVOR-NAME \
    --property quota:read_bytes_sec=10240000 \
    --property quota:write_bytes_sec=10240000
```

Use these optional parameters to control weight shares, enforcement intervals for runtime quotas, and a quota for maximum allowed bandwidth:

- `cpu_shares`: Specifies the proportional weighted share for the domain. If this element is omitted, the service defaults to the OS provided defaults. There is no unit for the value; it is a relative measure based on the setting of other VMs. For example, a VM configured with value 2048 gets twice as much CPU time as a VM configured with value 1024.
- `cpu_shares_level`: On VMware, specifies the allocation level. Can be `custom`, `high`, `normal`, or `low`. If you choose `custom`, set the number of shares using `cpu_shares_share`.
- `cpu_period`: Specifies the enforcement interval (unit: microseconds) for QEMU and LXC hypervisors. Within a period, each VCPU of the domain is not allowed to consume more than the quota worth of runtime. The value should be in range `[1000, 1000000]`. A period with value 0 means no value.
- `cpu_limit`: Specifies the upper limit for VMware machine CPU allocation in MHz. This parameter ensures that a machine never uses more than the defined amount of CPU time. It can be used to enforce a limit on the machines CPU performance.
- `cpu_reservation`: Specifies the guaranteed minimum CPU reservation in MHz for VMware. This means that if needed, the machine will definitely get allocated the reserved amount of CPU cycles.
- `cpu_quota`: Specifies the maximum allowed bandwidth (unit: microseconds). A domain with a negative-value quota indicates that the domain has infinite bandwidth, which means that it is not bandwidth controlled. The value should be in range `[1000, 18446744073709551]` or less than 0. A quota with value 0 means no value. You can use this feature to ensure that all vCPUs run at the same speed. For example:

```
$ openstack flavor set FLAVOR-NAME \
    --property quota:cpu_quota=10000 \
    --property quota:cpu_period=20000
```

In this example, an instance of `FLAVOR-NAME` can only consume a maximum of 50% CPU of a physical CPU computing capability.

Memory limits For VMware, you can configure the memory limits with control parameters.

Use these optional parameters to limit the memory allocation, guarantee minimum memory reservation, and to specify shares used in case of resource contention:

- `memory_limit`: Specifies the upper limit for VMware machine memory allocation in MB. The utilization of a virtual machine will not exceed this limit, even if there are available resources. This is typically used to ensure a consistent performance of virtual machines independent of available resources.
- `memory_reservation`: Specifies the guaranteed minimum memory reservation in MB for VMware. This means the specified amount of memory will definitely be allocated to the machine.
- `memory_shares_level`: On VMware, specifies the allocation level. This can be `custom`, `high`, `normal` or `low`. If you choose `custom`, set the number of shares using `memory_shares_share`.
- `memory_shares_share`: Specifies the number of shares allocated in the event that `custom` is used. There is no unit for this value. It is a relative measure based on the settings for other VMs. For example:

```
$ openstack flavor set FLAVOR-NAME \
    --property quota:memory_shares_level=custom \
    --property quota:memory_shares_share=15
```

Disk I/O limits For VMware, you can configure the resource limits for disk with control parameters.

Use these optional parameters to limit the disk utilization, guarantee disk allocation, and to specify shares used in case of resource contention. This allows the VMware driver to enable disk allocations for the running instance.

- `disk_io_limit`: Specifies the upper limit for disk utilization in I/O per second. The utilization of a virtual machine will not exceed this limit, even if there are available resources. The default value is -1 which indicates unlimited usage.
- `disk_io_reservation`: Specifies the guaranteed minimum disk allocation in terms of Input/output Operations Per Second (IOPS).
- `disk_io_shares_level`: Specifies the allocation level. This can be `custom`, `high`, `normal` or `low`. If you choose custom, set the number of shares using `disk_io_shares_share`.
- `disk_io_shares_share`: Specifies the number of shares allocated in the event that `custom` is used. When there is resource contention, this value is used to determine the resource allocation.

  The example below sets the `disk_io_reservation` to 2000 IOPS.

```
$ openstack flavor set FLAVOR-NAME \
    --property quota:disk_io_reservation=2000
```

Disk tuning Using disk I/O quotas, you can set maximum disk write to 10 MB per second for a VM user. For example:

```
$ openstack flavor set FLAVOR-NAME \
    --property quota:disk_write_bytes_sec=10485760
```

The disk I/O options are:

- `disk_read_bytes_sec`
- `disk_read_iops_sec`
- `disk_write_bytes_sec`
- `disk_write_iops_sec`
- `disk_total_bytes_sec`
- `disk_total_iops_sec`

Bandwidth I/O The vif I/O options are:

- `vif_inbound_average`
- `vif_inbound_burst`
- `vif_inbound_peak`
- `vif_outbound_average`
- `vif_outbound_burst`
- `vif_outbound_peak`

Incoming and outgoing traffic can be shaped independently. The bandwidth element can have at most, one inbound and at most, one outbound child element. If you leave any of these child elements out, no quality of service (QoS) is applied on that traffic direction. So, if you want to shape only the networks incoming traffic, use inbound only (and vice versa). Each element has one mandatory attribute average, which specifies the average bit rate on the interface being shaped.

There are also two optional attributes (integer): `peak`, which specifies the maximum rate at which a bridge can send data (kilobytes/second), and `burst`, the amount of bytes that can be burst at peak speed (kilobytes). The rate is shared equally within domains connected to the network.

The example below sets network traffic bandwidth limits for existing flavor as follows:

- Outbound traffic:

  - average: 262 Mbps (32768 kilobytes/second)
  - peak: 524 Mbps (65536 kilobytes/second)
  - burst: 65536 kilobytes
- Inbound traffic:
  - average: 262 Mbps (32768 kilobytes/second)
  - peak: 524 Mbps (65536 kilobytes/second)
  - burst: 65536 kilobytes

```
$ openstack flavor set FLAVOR-NAME \
    --property quota:vif_outbound_average=32768 \
    --property quota:vif_outbound_peak=65536 \
    --property quota:vif_outbound_burst=65536 \
    --property quota:vif_inbound_average=32768 \
    --property quota:vif_inbound_peak=65536 \
    --property quota:vif_inbound_burst=65536
```

Note: All the speed limit values in above example are specified in kilobytes/second. And burst values are in kilobytes. Values were converted using Data rate units on Wikipedia.

Hardware video RAM Specify `hw_video:ram_max_mb` to control the maximum RAM for the video image. Used in conjunction with the `hw_video_ram` image property. `hw_video_ram` must be less than or equal to `hw_video:ram_max_mb`.

This is currently supported by the libvirt and the vmware drivers.

See https://libvirt.org/formatdomain.html#elementsVideo for more information on how this is used to set the `vram` attribute with the libvirt driver.

See https://pubs.vmware.com/vi-sdk/visdk250/ReferenceGuide/vim.vm.device.VirtualVideoCard.html for more information on how this is used to set the `videoRamSizeInKB` attribute with the vmware driver.

Watchdog behavior For the libvirt driver, you can enable and set the behavior of a virtual hardware watchdog device for each flavor. Watchdog devices keep an eye on the guest server, and carry out the configured 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.

To set the behavior, use:

```
$ openstack flavor set FLAVOR-NAME --property hw:watchdog_action=ACTION
```

Valid ACTION values are:

- `disabled`: (default) The device is not attached.
- `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.

Note: Watchdog behavior set using a specific images properties will override behavior set using flavors.

Random-number generator If a random-number generator device has been added to the instance through its image properties, the device can be enabled and configured using:

```
$ openstack flavor set FLAVOR-NAME \
    --property hw_rng:allowed=True \
    --property hw_rng:rate_bytes=RATE-BYTES \
    --property hw_rng:rate_period=RATE-PERIOD
```

Where:

- RATE-BYTES: (integer) Allowed amount of bytes that the guest can read from the hosts entropy per period.
- RATE-PERIOD: (integer) Duration of the read period in milliseconds.

Performance Monitoring Unit (vPMU) If nova is deployed with the libvirt virt driver and *libvirt.virt_type* is set to `qemu` or `kvm`, a vPMU can be enabled or disabled for an instance using the `hw:pmu` extra_spec or the `hw_pmu` image property. The supported values are `True` or `False`. If the vPMU is not explicitly enabled or disabled via the flavor or image, its presence is left to QEMU to decide.

```
$ openstack flavor set FLAVOR-NAME --property hw:pmu=True|False
```

The vPMU is used by tools like `perf` in the guest to provide more accurate information for profiling application and monitoring guest performance. For realtime workloads, the emulation of a vPMU can introduce additional latency which may be undesirable. If the telemetry it provides is not required, such workloads should set `hw:pmu=False`. For most workloads the default of unset or enabling the vPMU `hw:pmu=True` will be correct.

CPU topology For the libvirt driver, you can define the topology of the processors in the virtual machine using properties. The properties with `max` limit the number that can be selected by the user with image properties.

```
$ openstack flavor set FLAVOR-NAME \
    --property hw:cpu_sockets=FLAVOR-SOCKETS \
    --property hw:cpu_cores=FLAVOR-CORES \
    --property hw:cpu_threads=FLAVOR-THREADS \
    --property hw:cpu_max_sockets=FLAVOR-SOCKETS \
    --property hw:cpu_max_cores=FLAVOR-CORES \
    --property hw:cpu_max_threads=FLAVOR-THREADS
```

Where:

- FLAVOR-SOCKETS: (integer) The number of sockets for the guest VM. By default, this is set to the number of vCPUs requested.
- FLAVOR-CORES: (integer) The number of cores per socket for the guest VM. By default, this is set to `1`.
- FLAVOR-THREADS: (integer) The number of threads per core for the guest VM. By default, this is set to `1`.

CPU pinning policy For the libvirt driver, you can pin the virtual CPUs (vCPUs) of instances to the hosts physical CPU cores (pCPUs) using properties. You can further refine this by stating how hardware CPU threads in a simultaneous multithreading-based (SMT) architecture be used. These configurations will result in improved per-instance determinism and performance.

Note: SMT-based architectures include Intel processors with Hyper-Threading technology. In these architectures, processor cores share a number of components with one or more other cores. Cores in such architectures are commonly referred to as hardware threads, while the cores that a given core share components with are known as thread siblings.

Note: Host aggregates should be used to separate these pinned instances from unpinned instances as the latter will not respect the resourcing requirements of the former.

```
$ openstack flavor set FLAVOR-NAME \
    --property hw:cpu_policy=CPU-POLICY \
    --property hw:cpu_thread_policy=CPU-THREAD-POLICY
```

Valid CPU-POLICY values are:

- `shared`: (default) The guest vCPUs will be allowed to freely float across host pCPUs, albeit potentially constrained by NUMA policy.
- `dedicated`: The guest vCPUs will be strictly pinned to a set of host pCPUs. In the absence of an explicit vCPU topology request, the drivers typically expose all vCPUs as sockets with one core and one thread. When strict CPU pinning is in effect the guest CPU topology will be setup to match the topology of the CPUs to which it is pinned. This option implies an overcommit ratio of 1.0. For example, if a two vCPU guest is pinned to a single host core with two threads, then the guest will get a topology of one socket, one core, two threads.

Valid CPU-THREAD-POLICY values are:

- `prefer`: (default) The host may or may not have an SMT architecture. Where an SMT architecture is present, thread siblings are preferred.
- `isolate`: The host must not have an SMT architecture or must emulate a non-SMT architecture. If the host does not have an SMT architecture, each vCPU is placed on a different core as expected. If the host does have an SMT architecture - that is, one or more cores have thread siblings - then each vCPU is placed on a different physical core. No vCPUs from other guests are placed on the same core. All but one thread sibling on each utilized core is therefore guaranteed to be unusable.
- `require`: The host must have an SMT architecture. Each vCPU is allocated on thread siblings. If the host does not have an SMT architecture, then it is not used. If the host has an SMT architecture, but not enough cores with free thread siblings are available, then scheduling fails.

Note: The `hw:cpu_thread_policy` option is only valid if `hw:cpu_policy` is set to `dedicated`.

NUMA topology For the libvirt driver, you can define the host NUMA placement for the instance vCPU threads as well as the allocation of instance vCPUs and memory from the host NUMA nodes. For flavors whose memory and vCPU allocations are larger than the size of NUMA nodes in the compute hosts, the definition of a NUMA topology allows hosts to better utilize NUMA and improve performance of the instance OS.

```
$ openstack flavor set FLAVOR-NAME \
    --property hw:numa_nodes=FLAVOR-NODES \
    --property hw:numa_cpus.N=FLAVOR-CORES \
    --property hw:numa_mem.N=FLAVOR-MEMORY
```

Where:

- FLAVOR-NODES: (integer) The number of host NUMA nodes to restrict execution of instance vCPU threads to. If not specified, the vCPU threads can run on any number of the host NUMA nodes available.
- N: (integer) The instance NUMA node to apply a given CPU or memory configuration to, where N is in the range `0` to `FLAVOR-NODES - 1`.
- FLAVOR-CORES: (comma-separated list of integers) A list of instance vCPUs to map to instance NUMA node N. If not specified, vCPUs are evenly divided among available NUMA nodes.

- FLAVOR-MEMORY: (integer) The number of MB of instance memory to map to instance NUMA node N. If not specified, memory is evenly divided among available NUMA nodes.

Note: `hw:numa_cpus.N` and `hw:numa_mem.N` are only valid if `hw:numa_nodes` is set. Additionally, they are only required if the instances NUMA nodes have an asymmetrical allocation of CPUs and RAM (important for some NFV workloads).

Note: The `N` parameter is an index of *guest* NUMA nodes and may not correspond to *host* NUMA nodes. For example, on a platform with two NUMA nodes, the scheduler may opt to place guest NUMA node 0, as referenced in `hw:numa_mem.0` on host NUMA node 1 and vice versa. Similarly, the integers used for `FLAVOR-CORES` are indexes of *guest* vCPUs and may not correspond to *host* CPUs. As such, this feature cannot be used to constrain instances to specific host CPUs or NUMA nodes.

Warning: If the combined values of `hw:numa_cpus.N` or `hw:numa_mem.N` are greater than the available number of CPUs or memory respectively, an exception is raised.

Hardware encryption of guest memory If there are compute hosts which support encryption of guest memory at the hardware level, this functionality can be requested via the `hw:mem_encryption` extra spec parameter:

```
$ openstack flavor set FLAVOR-NAME \
    --property hw:mem_encryption=True
```

CPU real-time policy For the libvirt driver, you can state that one or more of your instance virtual CPUs (vCPUs), though not all of them, run with a real-time policy. When used on a correctly configured host, this provides stronger guarantees for worst case scheduler latency for vCPUs and is a requirement for certain applications.

Todo: Document the required steps to configure hosts and guests. There are a lot of things necessary, from isolating hosts and configuring the `[compute] cpu_dedicated_set` nova configuration option on the host, to choosing a correctly configured guest image.

Important: While most of your instance vCPUs can run with a real-time policy, you must mark at least one vCPU as non-real-time, to be used for both non-real-time guest processes and emulator overhead (housekeeping) processes.

Important: To use this extra spec, you must enable pinned CPUs. Refer to *CPU policy* for more information.

```
$ openstack flavor set FLAVOR-NAME \
    --property hw:cpu_realtime=CPU-REALTIME-POLICY \
    --property hw:cpu_realtime_mask=CPU-REALTIME-MASK
```

Where:

CPU-REALTIME-POLICY (enum): One of:

- `no`: (default) The guest vCPUs will not have a real-time policy
- `yes`: The guest vCPUs will have a real-time policy

CPU-REALTIME-MASK (coremask): A coremask indicating which vCPUs **will not** have a real-time policy. This should start with a `^`. For example, a value of `^0-1` indicates that all vCPUs *except* vCPUs `0` and `1` will have a real-time policy.

Note: The `hw:cpu_realtime_mask` option is only valid if `hw:cpu_realtime` is set to `yes`.

Emulator threads policy For the libvirt driver, you can assign a separate pCPU to an instance that will be used for emulator threads, which are emulator processes not directly related to the guest OS. This pCPU will used in addition to the pCPUs used for the guest. This is generally required for use with a *real-time workload*.

Important: To use this extra spec, you must enable pinned CPUs. Refer to *CPU policy* for more information.

```
$ openstack flavor set FLAVOR-NAME \
    --property hw:emulator_threads_policy=THREAD-POLICY
```

The expected behavior of emulator threads depends on the value of the `hw:emulator_threads_policy` flavor extra spec and the value of *`compute.cpu_shared_set`*. It is presented in the following table:

	`compute.cpu_shared_set` set	*`compute.cpu_shared_set`* unset
`hw:emulator_treads_policy` unset (default)	Pinned to all of the instances pCPUs	Pinned to all of the instances pCPUs
`hw:emulator_threads_policy` = `share`	Pinned to *`compute.cpu_shared_set`*	Pinned to all of the instances pCPUs
`hw:emulator_threads_policy` = `isolate`	Pinned to a single pCPU distinct from the instances pCPUs	Pinned to a single pCPU distinct from the instances pCPUs

Large pages allocation You can configure the size of large pages used to back the VMs.

```
$ openstack flavor set FLAVOR-NAME \
    --property hw:mem_page_size=PAGE_SIZE
```

Valid `PAGE_SIZE` values are:

- `small`: (default) The smallest page size is used. Example: 4 KB on x86.
- `large`: Only use larger page sizes for guest RAM. Example: either 2 MB or 1 GB on x86.
- `any`: It is left up to the compute driver to decide. In this case, the libvirt driver might try to find large pages, but fall back to small pages. Other drivers may choose alternate policies for `any`.
- pagesize: (string) An explicit page size can be set if the workload has specific requirements. This value can be an integer value for the page size in KB, or can use any standard suffix. Example: `4KB`, `2MB`, `2048`, `1GB`.

Note: Large pages can be enabled for guest RAM without any regard to whether the guest OS will use them or not. If the guest OS chooses not to use huge pages, it will merely see small pages as before. Conversely, if a guest OS does intend to use huge pages, it is very important that the guest RAM be backed by huge pages. Otherwise, the guest OS will not be getting the performance benefit it is expecting.

PCI passthrough You can assign PCI devices to a guest by specifying them in the flavor.

```
$ openstack flavor set FLAVOR-NAME \
    --property pci_passthrough:alias=ALIAS:COUNT
```

Where:

- ALIAS: (string) The alias which correspond to a particular PCI device class as configured in the nova configuration file (see *pci.alias*).
- COUNT: (integer) The amount of PCI devices of type ALIAS to be assigned to a guest.

Hiding hypervisor signature Some hypervisors add a signature to their guests. While the presence of the signature can enable some paravirtualization features on the guest, it can also have the effect of preventing some drivers from loading. Hiding the signature by setting this property to true may allow such drivers to load and work.

Note: As of the 18.0.0 Rocky release, this is only supported by the libvirt driver.

Prior to the 21.0.0 Ussuri release, this was called `hide_hypervisor_id`. An alias is provided to provide backwards compatibility.

```
$ openstack flavor set FLAVOR-NAME \
    --property hw:hide_hypervisor_id=VALUE
```

Where:

- VALUE: (string) true or false. false is equivalent to the property not existing.

Secure Boot When your Compute services use the Hyper-V hypervisor, you can enable secure boot for Windows and Linux instances.

```
$ openstack flavor set FLAVOR-NAME \
    --property os:secure_boot=SECURE_BOOT_OPTION
```

Valid `SECURE_BOOT_OPTION` values are:

- `required`: Enable Secure Boot for instances running with this flavor.
- `disabled` or `optional`: (default) Disable Secure Boot for instances running with this flavor.

Custom resource classes and standard resource classes to override Added in the 16.0.0 Pike release.

Specify custom resource classes to require or override quantity values of standard resource classes.

The syntax of the extra spec is `resources:<resource_class_name>=VALUE` (`VALUE` is integer). The name of custom resource classes must start with `CUSTOM_`. Standard resource classes to override are `VCPU`, `MEMORY_MB` or `DISK_GB`. In this case, you can disable scheduling based on standard resource classes by setting the value to `0`.

For example:

- resources:CUSTOM_BAREMETAL_SMALL=1
- resources:VCPU=0

See Create flavors for use with the Bare Metal service for more examples.

Required traits Added in the 17.0.0 Queens release.

Required traits allow specifying a server to build on a compute node with the set of traits specified in the flavor. The traits are associated with the resource provider that represents the compute node in the Placement API. See the resource provider traits API reference for more details: https://docs.openstack.org/api-ref/placement/#resource-provider-traits

The syntax of the extra spec is `trait:<trait_name>=required`, for example:

- trait:HW_CPU_X86_AVX2=required
- trait:STORAGE_DISK_SSD=required

The scheduler will pass required traits to the `GET /allocation_candidates` endpoint in the Placement API to include only resource providers that can satisfy the required traits. In 17.0.0 the only valid value is `required`. In 18.0.0 `forbidden` is added (see below). Any other value will be considered invalid.

The FilterScheduler is currently the only scheduler driver that supports this feature.

Traits can be managed using the osc-placement plugin.

Forbidden traits Added in the 18.0.0 Rocky release.

Forbidden traits are similar to required traits, described above, but instead of specifying the set of traits that must be satisfied by a compute node, forbidden traits must **not** be present.

The syntax of the extra spec is `trait:<trait_name>=forbidden`, for example:

- trait:HW_CPU_X86_AVX2=forbidden
- trait:STORAGE_DISK_SSD=forbidden

The FilterScheduler is currently the only scheduler driver that supports this feature.

Traits can be managed using the osc-placement plugin.

Numbered groupings of resource classes and traits Added in the 18.0.0 Rocky release.

Specify numbered groupings of resource classes and traits.

The syntax is as follows (`N` and `VALUE` are integers):

`resources`*`N`*`:<`*`resource_class_name`*`>=`*`VALUE`
`trait`*`N`*`:<`*`trait_name`*`>=required`

A given numbered `resources` or `trait` key may be repeated to specify multiple resources/traits in the same grouping, just as with the un-numbered syntax.

Specify inter-group affinity policy via the `group_policy` key, which may have the following values:

- `isolate`: Different numbered request groups will be satisfied by *different* providers.
- `none`: Different numbered request groups may be satisfied by different providers *or* common providers.

Note: If more than one group is specified then the `group_policy` is mandatory in the request. However such groups might come from other sources than flavor extra_spec (e.g. from Neutron ports with QoS minimum bandwidth policy). If the flavor does not specify any groups and `group_policy` but more than one group is coming from other sources then nova will default the `group_policy` to `none` to avoid scheduler failure.

For example, to create a server with the following VFs:

- One SR-IOV virtual function (VF) on NET1 with bandwidth 10000 bytes/sec
- One SR-IOV virtual function (VF) on NET2 with bandwidth 20000 bytes/sec on a *different* NIC with SSL acceleration

It is specified in the extra specs as follows:

```
resources1:SRIOV_NET_VF=1
resources1:NET_EGRESS_BYTES_SEC=10000
trait1:CUSTOM_PHYSNET_NET1=required
```

(continues on next page)

(continued from previous page)

```
resources2:SRIOV_NET_VF=1
resources2:NET_EGRESS_BYTES_SEC:20000
trait2:CUSTOM_PHYSNET_NET2=required
trait2:HW_NIC_ACCEL_SSL=required
group_policy=isolate
```

See Granular Resource Request Syntax for more details.

4.54 Manage IP addresses

Each instance has a private, fixed IP address and can also have a public, or floating IP address. Private IP addresses are used for communication between instances, and public addresses are used for communication with networks outside the cloud, including the Internet.

When you launch an instance, it is automatically assigned a private IP address that stays the same until you explicitly terminate the instance. Rebooting an instance has no effect on the private IP address.

A pool of floating IP addresses, configured by the cloud administrator, is available in OpenStack Compute. The project quota defines the maximum number of floating IP addresses that you can allocate to the project. After you allocate a floating IP address to a project, you can:

- Associate the floating IP address with an instance of the project.
- Disassociate a floating IP address from an instance in the project.
- Delete a floating IP from the project which automatically deletes that IPs associations.

Use the **openstack** commands to manage floating IP addresses.

4.54.1 List floating IP address information

To list all pools that provide floating IP addresses, run:

```
$ openstack floating ip pool list
+--------+
| name   |
+--------+
| public |
| test   |
+--------+
```

Note: If this list is empty, the cloud administrator must configure a pool of floating IP addresses.

To list all floating IP addresses that are allocated to the current project, run:

```
$ openstack floating ip list
+--------------------------------------+---------------------+------------------+-----
↪-+
| ID                                   | Floating IP Address | Fixed IP Address |␣
↪Port |
+--------------------------------------+---------------------+------------------+-----
↪-+
| 760963b2-779c-4a49-a50d-f073c1ca5b9e | 172.24.4.228        | None             |␣
↪None |
```

(continues on next page)

(continued from previous page)

```
| 89532684-13e1-4af3-bd79-f434c9920cc3 | 172.24.4.235        | None                |␣
↪None |
| ea3ebc6d-a146-47cd-aaa8-35f06e1e8c3d | 172.24.4.229        | None                |␣
↪None |
+--------------------------------------+---------------------+-----------------+-----
↪-+
```

For each floating IP address that is allocated to the current project, the command outputs the floating IP address, the ID for the instance to which the floating IP address is assigned, the associated fixed IP address, and the pool from which the floating IP address was allocated.

4.54.2 Associate floating IP addresses

You can assign a floating IP address to a project and to an instance.

1. Run the following command to allocate a floating IP address to the current project. By default, the floating IP address is allocated from the public pool. The command outputs the allocated IP address:

```
$ openstack floating ip create public
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| created_at          | 2016-11-30T15:02:05Z                 |
| description         |                                      |
| fixed_ip_address    | None                                 |
| floating_ip_address | 172.24.4.236                         |
| floating_network_id | 0bf90de6-fc0f-4dba-b80d-96670dfb331a |
| headers             |                                      |
| id                  | c70ad74b-2f64-4e60-965e-f24fc12b3194 |
| port_id             | None                                 |
| project_id          | 5669caad86a04256994cdf755df4d3c1     |
| project_id          | 5669caad86a04256994cdf755df4d3c1     |
| revision_number     | 1                                    |
| router_id           | None                                 |
| status              | DOWN                                 |
| updated_at          | 2016-11-30T15:02:05Z                 |
+---------------------+--------------------------------------+
```

2. List all project instances with which a floating IP address could be associated.

```
$ openstack server list
+---------------------+------+---------+------------+-------------+---------------
↪---+------------+
| ID                  | Name | Status  | Task State | Power State | Networks      ␣
↪    | Image Name |
+---------------------+------+---------+------------+-------------+---------------
↪---+------------+
| d5c854f9-d3e5-4f... | VM1  | ACTIVE  | -          | Running     | private=10.0.
↪0.3 | cirros     |
| 42290b01-0968-43... | VM2  | SHUTOFF | -          | Shutdown    | private=10.0.
↪0.4 | centos     |
+---------------------+------+---------+------------+-------------+---------------
↪---+------------+
```

   Note the server ID to use.

3. List ports associated with the selected server.

```
$ openstack port list --device-id SERVER_ID
+--------------------------------------+------+-------------------+---------------
↪-----------------------------------------------+--------+
| ID                                   | Name | MAC Address       | Fixed IP␣
↪Addresses                                             | Status |
+--------------------------------------+------+-------------------+---------------
↪-----------------------------------------------+--------+
| 40e9dea9-f457-458f-bc46-6f4ebea3c268 |      | fa:16:3e:00:57:3e | ip_address=
↪'10.0.0.4', subnet_id='23ee9de7-362e-             | ACTIVE |
|                                      |      |                   | 49e2-a3b0-
↪0de1c14930cb'                                       |        |
|                                      |      |                   | ip_address=
↪'fd22:4c4c:81c2:0:f816:3eff:fe00:573e', subnet_id |        |
|                                      |      |                   | ='a2b3acbe-
↪fbeb-40d3-b21f-121268c21b55'                        |        |
+--------------------------------------+------+-------------------+---------------
↪-----------------------------------------------+--------+
```

Note the port ID to use.

4. Associate an IP address with an instance in the project, as follows:

```
$ openstack floating ip set --port PORT_ID FLOATING_IP_ADDRESS
```

For example:

```
$ openstack floating ip set --port 40e9dea9-f457-458f-bc46-6f4ebea3c268 172.24.4.
↪225
```

The instance is now associated with two IP addresses:

```
$ openstack server list
+------------------+------+--------+------------+-------------+------------------
↪------------+------------+
| ID               | Name | Status | Task State | Power State | Networks         ␣
↪            | Image Name |
+------------------+------+--------+------------+-------------+------------------
↪------------+------------+
| d5c854f9-d3e5... | VM1  | ACTIVE | -          | Running     | private=10.0.0.3,␣
↪172.24.4.225| cirros     |
| 42290b01-0968... | VM2  | SHUTOFF| -          | Shutdown    | private=10.0.0.4 ␣
↪            | centos     |
+------------------+------+--------+------------+-------------+------------------
↪------------+------------+
```

After you associate the IP address and configure security group rules for the instance, the instance is publicly available at the floating IP address.

4.54.3 Disassociate floating IP addresses

To disassociate a floating IP address from an instance:

```
$ openstack floating ip unset --port FLOATING_IP_ADDRESS
```

To remove the floating IP address from a project:

```
$ openstack floating ip delete FLOATING_IP_ADDRESS
```

The IP address is returned to the pool of IP addresses that is available for all projects. If the IP address is still associated with a running instance, it is automatically disassociated from that instance.

4.55 Quotas

Nova uses a quota system for setting limits on resources such as number of instances or amount of CPU that a specific project or user can use.

Starting in the 16.0.0 Pike release the quota calculation system in nova was overhauled and the old reserve/commit/rollback flow was changed to count resource usage at the point of whatever operation is being performed, such as creating or resizing a server. A check will be performed by counting current usage for the relevant resource and then if `quota.recheck_quota` is True (it is by default) another check will be performed to ensure the initial check is still valid.

By default resource usage is counted using the API and cell databases but nova can be configured to count some resource usage without using the cell databases, see *Quota usage from placement* for details.

Quota limits and usage can be retrieved via the limits REST API. Quota limits can be set per-tenant using the quota sets REST API or per class (all tenants) using the quota class sets API.

4.55.1 Checking quota

When calculating limits for a given resource and tenant, the following checks are made in order:

- Depending on the resource, is there a tenant-specific limit on the resource in either the *quotas* or *project_user_quotas* tables in the database? If so, use that as the limit. You can create these resources by doing:

  ```
  openstack quota set --instances 5 <project>
  ```

- Check to see if there is a hard limit for the given resource in the *quota_classes* table in the database for the *default* quota class. If so, use that as the limit. You can modify the default quota limit for a resource by doing:

  ```
  openstack quota set --class --instances 5 default
  ```

- If the above does not provide a resource limit, then rely on the `quota` configuration options for the default limit.

Note: The API sets the limit in the *quota_classes* table. Once a default limit is set via the *default* quota class, that takes precedence over any changes to that resource limit in the configuration options. In other words, once youve changed things via the API, you either have to keep those synchronized with the configuration values or remove the default limit from the database manually as there is no REST API for removing quota class values from the database.

4.55.2 Quota usage from placement

Starting in the Train (20.0.0) release, it is possible to configure quota usage counting of cores and ram from the placement service and instances from instance mappings in the API database instead of counting resources from cell databases. This makes quota usage counting resilient in the presence of down or poor-performing cells.

Quota usage counting from placement is opt-in via configuration option:

```
[quota]
count_usage_from_placement = True
```

There are some things to note when opting in to counting quota usage from placement:

- Counted usage will not be accurate in an environment where multiple Nova deployments are sharing a placement deployment because currently placement has no way of partitioning resource providers between different Nova deployments. Operators who are running multiple Nova deployments that share a placement deployment should not set the *quota.count_usage_from_placement* configuration option to `True`.
- Behavior will be different for resizes. During a resize, resource allocations are held on both the source and destination (even on the same host, see https://bugs.launchpad.net/nova/+bug/1790204) until the resize is confirmed or reverted. Quota usage will be inflated for servers in this state and operators should weigh the advantages and disadvantages before enabling *quota.count_usage_from_placement*.
- The `populate_queued_for_delete` and `populate_user_id` online data migrations must be completed before usage can be counted from placement. Until the data migration is complete, the system will fall back to legacy quota usage counting from cell databases depending on the result of an EXISTS database query during each quota check, if *quota.count_usage_from_placement* is set to `True`. Operators who want to avoid the performance hit from the EXISTS queries should wait to set the *quota.count_usage_from_placement* configuration option to `True` until after they have completed their online data migrations via `nova-manage db online_data_migrations`.
- Behavior will be different for unscheduled servers in `ERROR` state. A server in `ERROR` state that has never been scheduled to a compute host will not have placement allocations, so it will not consume quota usage for cores and ram.
- Behavior will be different for servers in `SHELVED_OFFLOADED` state. A server in `SHELVED_OFFLOADED` state will not have placement allocations, so it will not consume quota usage for cores and ram. Note that because of this, it will be possible for a request to unshelve a server to be rejected if the user does not have enough quota available to support the cores and ram needed by the server to be unshelved.

4.55.3 Known issues

If not *counting quota usage from placement* it is possible for down or poor performing cells to impact quota calculations. See the *cells documentation* for details.

4.55.4 Future plans

Hierarchical quotas

There has long been a desire to support hierarchical or nested quotas leveraging support in the identity service for hierarchical projects. See the unified limits spec for details.

4.56 Feature Support Matrix

When considering which capabilities should be marked as mandatory the following general guiding principles were applied

- **Inclusivity** - people have shown ability to make effective use of a wide range of virtualization technologies with broadly varying feature sets. Aiming to keep the requirements as inclusive as possible, avoids second-guessing what a user may wish to use the cloud compute service for.

- **Bootstrapping** - a practical use case test is to consider that starting point for the compute deploy is an empty data center with new machines and network connectivity. The look at what are the minimum features required of a compute service, in order to get user instances running and processing work over the network.
- **Competition** - an early leader in the cloud compute service space was Amazon EC2. A sanity check for whether a feature should be mandatory is to consider whether it was available in the first public release of EC2. This had quite a narrow feature set, but none the less found very high usage in many use cases. So it serves to illustrate that many features need not be considered mandatory in order to get useful work done.
- **Reality** - there are many virt drivers currently shipped with Nova, each with their own supported feature set. Any feature which is missing in at least one virt driver that is already in-tree, must by inference be considered optional until all in-tree drivers support it. This does not rule out the possibility of a currently optional feature becoming mandatory at a later date, based on other principles above.

Summary

Feature	*Status*	**Hyper-V**	**Ironic**	**Libvirt KVM (aarch64)**	**Libvirt KVM**
Attach block volume to instance	optional	✓		*?*	✓
Attach tagged block device to instance	optional			*?*	✓
Detach block volume from instance	optional	✓		*?*	✓
Extend block volume attached to instance	optional			*?*	*?*
Attach virtual network interface to instance	optional	✓	✓	✓	✓
Attach tagged virtual network interface to instance	optional			*?*	✓
Detach virtual network interface from instance	optional	✓	✓	✓	✓
Set the host in a maintenance mode	optional				
Evacuate instances from a host	optional	*?*	*?*	✓	*?*
Rebuild instance	optional	✓	✓	✓	✓
Guest instance status	mandatory	✓	✓	✓	✓
Guest host uptime	optional	✓		✓	✓
Guest host ip	optional	✓		✓	✓
Live migrate instance across hosts	optional	✓			✓
Force live migration to complete	optional				✓
Launch instance	mandatory	✓	✓	✓	✓
Stop instance CPUs (pause)	optional	✓		✓	✓
Reboot instance	optional	✓	✓	✓	✓
Rescue instance	optional	✓	✓	*?*	✓
Resize instance	optional	✓		✓	✓
Restore instance	optional	✓		✓	✓
Set instance admin password	optional			*?*	
Save snapshot of instance disk	optional	✓		✓	✓
Suspend instance	optional	✓		✓	✓
Swap block volumes	optional			*?*	✓
Shutdown instance	mandatory	✓	✓	✓	✓
Trigger crash dump	optional		✓	*?*	✓
Resume instance CPUs (unpause)	optional	✓		✓	✓
uefi boot	optional		✓	✓	
Device tags	optional	✓		*?*	✓
quiesce	optional			*?*	✓
unquiesce	optional			*?*	✓
Attach block volume to multiple instances	optional			*?*	✓
Attach encrypted block volume to server	optional			*?*	*?*
Validate image with trusted certificates	optional			✓	✓
File backed memory	optional			*?*	*?*

Feature	*Status*	**Hyper-V**	**Ironic**	**Libvirt KVM (aarch64)**	**Libvirt KVM**
Report CPU traits	optional			*?*	✓
SR-IOV ports with resource request	optional				
Boot instance with secure encrypted memory	optional				

Details

- **Attach block volume to instance Status: optional.**

  CLI commands:

  - `nova volume-attach <server> <volume>`

  Notes: The attach volume operation provides a means to hotplug additional block storage to a running instance. This allows storage capabilities to be expanded without interruption of service. In a cloud model it would be more typical to just spin up a new instance with large storage, so the ability to hotplug extra storage is for those cases where the instance is considered to be more of a pet than cattle. Therefore this operation is not considered to be mandatory to support.

  Driver Support:

  - **Hyper-V:** `complete`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `unknown`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `missing`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `complete` **Notes:** This is not tested for every CI run. Add a powervm:volume-check comment to trigger a CI job running volume tests.
  - **VMware vCenter:** `complete`
  - **XenServer:** `complete`
  - **zVM:** `missing`

- **Attach tagged block device to instance Status: optional.**

  CLI commands:

  - `nova volume-attach <server> <volume> [--tag <tag>]`

  Notes: Attach a block device with a tag to an existing server instance. See Device tags for more information.

  Driver Support:

  - **Hyper-V:** `missing`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `unknown`

  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `missing`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `missing`
  - **XenServer:** `missing`
  - **zVM:** `missing`

- **Detach block volume from instance Status: optional.**

  CLI commands:

  - `nova volume-detach <server> <volume>`

  Notes: See notes for attach volume operation.

  Driver Support:

  - **Hyper-V:** `complete`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `unknown`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `missing`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `complete` **Notes:** This is not tested for every CI run. Add a powervm:volume-check comment to trigger a CI job running volume tests.
  - **VMware vCenter:** `complete`
  - **XenServer:** `complete`
  - **zVM:** `missing`

- **Extend block volume attached to instance Status: optional.**

  CLI commands:

  - `cinder extend <volume> <new_size>`

Notes: The extend volume operation provides a means to extend the size of an attached volume. This allows volume size to be expanded without interruption of service. In a cloud model it would be more typical to just spin up a new instance with large storage, so the ability to extend the size of an attached volume is for those cases where the instance is considered to be more of a pet than cattle. Therefore this operation is not considered to be mandatory to support.

Driver Support:

- **Hyper-V:** `missing`
- **Ironic:** `missing`
- **Libvirt KVM (aarch64):** `unknown`
- **Libvirt KVM (ppc64):** `unknown`
- **Libvirt KVM (s390x):** `unknown`
- **Libvirt KVM (x86):** `complete`
- **Libvirt LXC:** `missing`
- **Libvirt QEMU (x86):** `complete`
- **Libvirt Virtuozzo CT:** `missing`
- **Libvirt Virtuozzo VM:** `unknown`
- **Libvirt Xen:** `unknown`
- **PowerVM:** `complete` **Notes:** This is not tested for every CI run. Add a powervm:volume-check comment to trigger a CI job running volume tests.
- **VMware vCenter:** `missing`
- **XenServer:** `missing`
- **zVM:** `missing`

- **Attach virtual network interface to instance Status: optional.**

  CLI commands:

  - `nova interface-attach <server>`

  Notes: The attach interface operation provides a means to hotplug additional interfaces to a running instance. Hotplug support varies between guest OSes and some guests require a reboot for new interfaces to be detected. This operation allows interface capabilities to be expanded without interruption of service. In a cloud model it would be more typical to just spin up a new instance with more interfaces.

  Driver Support:

  - **Hyper-V:** `partial` **Notes:** Works without issue if instance is off. When hotplugging, only works if using Windows/Hyper-V Server 2016 and the instance is a Generation 2 VM.
  - **Ironic:** `complete`
  - **Libvirt KVM (aarch64):** `complete`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `complete`

  - **Libvirt Virtuozzo CT:** `complete`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `complete`
  - **VMware vCenter:** `complete`
  - **XenServer:** `complete`
  - **zVM:** `missing`

- **Attach tagged virtual network interface to instance Status: optional.**

  CLI commands:

  - `nova interface-attach <server> [--tag <tag>]`

  Notes: Attach a virtual network interface with a tag to an existing server instance. See Device tags for more information.

  Driver Support:

  - **Hyper-V:** `missing`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `unknown`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `missing`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `missing`
  - **XenServer:** `missing`
  - **zVM:** `missing`

- **Detach virtual network interface from instance Status: optional.**

  CLI commands:

  - `nova interface-detach <server> <port_id>`

  Notes: See notes for attach-interface operation.

  Driver Support:

  - **Hyper-V:** `complete` **Notes:** Works without issue if instance is off. When hotplugging, only works if using Windows/Hyper-V Server 2016 and the instance is a Generation 2 VM.
  - **Ironic:** `complete`
  - **Libvirt KVM (aarch64):** `complete`

  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `complete`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `complete`
  - **VMware vCenter:** `complete`
  - **XenServer:** `complete`
  - **zVM:** `missing`

- **Set the host in a maintenance mode Status: optional.**

  CLI commands:

  - `nova host-update <host>`

  Notes: This operation allows a host to be placed into maintenance mode, automatically triggering migration of any running instances to an alternative host and preventing new instances from being launched. This is not considered to be a mandatory operation to support. The driver methods to implement are host_maintenance_mode and set_host_enabled.

  Driver Support:

  - **Hyper-V:** `missing`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `missing`
  - **Libvirt KVM (ppc64):** `missing`
  - **Libvirt KVM (s390x):** `missing`
  - **Libvirt KVM (x86):** `missing`
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `missing`
  - **Libvirt Virtuozzo CT:** `missing`
  - **Libvirt Virtuozzo VM:** `missing`
  - **Libvirt Xen:** `missing`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `missing`
  - **XenServer:** `complete`
  - **zVM:** `missing`

- **Evacuate instances from a host Status: optional.**

  CLI commands:

  - `nova evacuate <server>`
  - `nova host-evacuate <host>`

  Notes: A possible failure scenario in a cloud environment is the outage of one of the compute nodes. In such a case the instances of the down host can be evacuated to another host. It is assumed that the old host is unlikely ever to be powered back on, otherwise the evacuation attempt will be rejected. When the instances get moved to the new host, their volumes get re-attached and the locally stored data is dropped. That happens in the same way as a rebuild. This is not considered to be a mandatory operation to support.

  Driver Support:

  - **Hyper-V:** `unknown`
  - **Ironic:** `unknown`
  - **Libvirt KVM (aarch64):** `complete`
  - **Libvirt KVM (ppc64):** `unknown`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `unknown`
  - **Libvirt QEMU (x86):** `unknown`
  - **Libvirt Virtuozzo CT:** `missing`
  - **Libvirt Virtuozzo VM:** `missing`
  - **Libvirt Xen:** `unknown`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `unknown`
  - **XenServer:** `unknown`
  - **zVM:** `unknown`

- **Rebuild instance Status: optional.**

  CLI commands:

  - `nova rebuild <server> <image>`

  Notes: A possible use case is additional attributes need to be set to the instance, nova will purge all existing data from the system and remakes the VM with given information such as metadata and personalities. Though this is not considered to be a mandatory operation to support.

  Driver Support:

  - **Hyper-V:** `complete`
  - **Ironic:** `complete`
  - **Libvirt KVM (aarch64):** `complete`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `complete`
  - **Libvirt QEMU (x86):** `complete`

  - **Libvirt Virtuozzo CT:** `complete`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `complete`
  - **XenServer:** `complete`
  - **zVM:** `unknown`

- **Guest instance status Status: mandatory.**

  Notes: Provides realtime information about the power state of the guest instance. Since the power state is used by the compute manager for tracking changes in guests, this operation is considered mandatory to support.

  Driver Support:

  - **Hyper-V:** `complete`
  - **Ironic:** `complete`
  - **Libvirt KVM (aarch64):** `complete`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `complete`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `complete`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `complete`
  - **VMware vCenter:** `complete`
  - **XenServer:** `complete`
  - **zVM:** `complete`

- **Guest host uptime Status: optional.**

  Notes: Returns the result of host uptime since power on, its used to report hypervisor status.

  Driver Support:

  - **Hyper-V:** `complete`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `complete`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `complete`

  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `complete`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `complete`
  - **VMware vCenter:** `missing`
  - **XenServer:** `complete`
  - **zVM:** `complete`

- **Guest host ip Status: optional.**

  Notes: Returns the ip of this host, its used when doing resize and migration.

  Driver Support:

  - **Hyper-V:** `complete`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `complete`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `complete`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `complete`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `complete`
  - **VMware vCenter:** `complete`
  - **XenServer:** `complete`
  - **zVM:** `complete`

- **Live migrate instance across hosts Status: optional.**

  CLI commands:

  - `nova live-migration <server>`
  - `nova host-evacuate-live <host>`

  Notes: Live migration provides a way to move an instance off one compute host, to another compute host. Administrators may use this to evacuate instances from a host that needs to undergo maintenance tasks, though of course this may not help if the host is already suffering a failure. In general instances are considered cattle rather than pets, so it is expected that an instance is liable to be killed if host maintenance is required. It is technically challenging for some hypervisors to provide support for the live migration operation, particularly those built on the container based virtualization. Therefore this operation is not considered mandatory to support.

  Driver Support:

  - **Hyper-V:** `complete`

  - **Ironic:** missing
  - **Libvirt KVM (aarch64):** missing
  - **Libvirt KVM (ppc64):** complete
  - **Libvirt KVM (s390x):** complete
  - **Libvirt KVM (x86):** complete
  - **Libvirt LXC:** missing
  - **Libvirt QEMU (x86):** complete
  - **Libvirt Virtuozzo CT:** complete
  - **Libvirt Virtuozzo VM:** complete
  - **Libvirt Xen:** complete
  - **PowerVM:** missing
  - **VMware vCenter:** complete
  - **XenServer:** complete
  - **zVM:** missing

- **Force live migration to complete Status: optional.**

  CLI commands:

  - nova live-migration-force-complete <server> <migration>

  Notes: Live migration provides a way to move a running instance to another compute host. But it can sometimes fail to complete if an instance has a high rate of memory or disk page access. This operation provides the user with an option to assist the progress of the live migration. The mechanism used to complete the live migration depends on the underlying virtualization subsystem capabilities. If libvirt/qemu is used and the post-copy feature is available and enabled then the force complete operation will cause a switch to post-copy mode. Otherwise the instance will be suspended until the migration is completed or aborted.

  Driver Support:

  - **Hyper-V:** missing
  - **Ironic:** missing
  - **Libvirt KVM (aarch64):** missing
  - **Libvirt KVM (ppc64):** complete **Notes:** Requires libvirt>=1.3.3, qemu>=2.5.0
  - **Libvirt KVM (s390x):** complete **Notes:** Requires libvirt>=1.3.3, qemu>=2.5.0
  - **Libvirt KVM (x86):** complete **Notes:** Requires libvirt>=1.3.3, qemu>=2.5.0
  - **Libvirt LXC:** missing
  - **Libvirt QEMU (x86):** complete **Notes:** Requires libvirt>=1.3.3, qemu>=2.5.0
  - **Libvirt Virtuozzo CT:** missing
  - **Libvirt Virtuozzo VM:** missing
  - **Libvirt Xen:** missing
  - **PowerVM:** missing
  - **VMware vCenter:** missing
  - **XenServer:** missing

  - **zVM:** missing

- **Launch instance Status: mandatory.**

  Notes: Importing pre-existing running virtual machines on a host is considered out of scope of the cloud paradigm. Therefore this operation is mandatory to support in drivers.

  Driver Support:

  - **Hyper-V:** complete
  - **Ironic:** complete
  - **Libvirt KVM (aarch64):** complete
  - **Libvirt KVM (ppc64):** complete
  - **Libvirt KVM (s390x):** complete
  - **Libvirt KVM (x86):** complete
  - **Libvirt LXC:** complete
  - **Libvirt QEMU (x86):** complete
  - **Libvirt Virtuozzo CT:** complete
  - **Libvirt Virtuozzo VM:** complete
  - **Libvirt Xen:** complete
  - **PowerVM:** complete
  - **VMware vCenter:** complete
  - **XenServer:** complete
  - **zVM:** complete

- **Stop instance CPUs (pause) Status: optional.**

  CLI commands:

  - nova pause <server>

  Notes: Stopping an instances CPUs can be thought of as roughly equivalent to suspend-to-RAM. The instance is still present in memory, but execution has stopped. The problem, however, is that there is no mechanism to inform the guest OS that this takes place, so upon unpausing, its clocks will no longer report correct time. For this reason hypervisor vendors generally discourage use of this feature and some do not even implement it. Therefore this operation is considered optional to support in drivers.

  Driver Support:

  - **Hyper-V:** complete
  - **Ironic:** missing
  - **Libvirt KVM (aarch64):** complete
  - **Libvirt KVM (ppc64):** complete
  - **Libvirt KVM (s390x):** complete
  - **Libvirt KVM (x86):** complete
  - **Libvirt LXC:** complete
  - **Libvirt QEMU (x86):** complete
  - **Libvirt Virtuozzo CT:** missing

  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `missing`
  - **XenServer:** `complete`
  - **zVM:** `complete`

- **Reboot instance Status: optional.**

  CLI commands:

  - `nova reboot <server>`

  Notes: It is reasonable for a guest OS administrator to trigger a graceful reboot from inside the instance. A host initiated graceful reboot requires guest co-operation and a non-graceful reboot can be achieved by a combination of stop+start. Therefore this operation is considered optional.

  Driver Support:

  - **Hyper-V:** `complete`
  - **Ironic:** `complete`
  - **Libvirt KVM (aarch64):** `complete`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `complete`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `complete`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `complete`
  - **VMware vCenter:** `complete`
  - **XenServer:** `complete`
  - **zVM:** `complete`

- **Rescue instance Status: optional.**

  CLI commands:

  - `nova rescue <server>`

  Notes: The rescue operation starts an instance in a special configuration whereby it is booted from an special root disk image. The goal is to allow an administrator to recover the state of a broken virtual machine. In general the cloud model considers instances to be cattle, so if an instance breaks the general expectation is that it be thrown away and a new instance created. Therefore this operation is considered optional to support in drivers.

  Driver Support:

  - **Hyper-V:** `complete`
  - **Ironic:** `complete`

  - **Libvirt KVM (aarch64):** `unknown`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `complete`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `complete`
  - **XenServer:** `complete`
  - **zVM:** `missing`

- **Resize instance Status: optional.**

  CLI commands:

  - `nova resize <server> <flavor>`

  Notes: The resize operation allows the user to change a running instance to match the size of a different flavor from the one it was initially launched with. There are many different flavor attributes that potentially need to be updated. In general it is technically challenging for a hypervisor to support the alteration of all relevant config settings for a running instance. Therefore this operation is considered optional to support in drivers.

  Driver Support:

  - **Hyper-V:** `complete`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `complete`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `complete`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `complete`
  - **XenServer:** `complete`
  - **zVM:** `missing`

- **Restore instance Status: optional.**

  CLI commands:

  - `nova resume <server>`

  Notes: See notes for the suspend operation

  Driver Support:

  - **Hyper-V:** `complete`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `complete`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `complete`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `complete`
  - **XenServer:** `complete`
  - **zVM:** `missing`

- **Set instance admin password Status: optional.**

  CLI commands:

  - `nova set-password <server>`

  Notes: Provides a mechanism to (re)set the password of the administrator account inside the instance operating system. This requires that the hypervisor has a way to communicate with the running guest operating system. Given the wide range of operating systems in existence it is unreasonable to expect this to be practical in the general case. The configdrive and metadata service both provide a mechanism for setting the administrator password at initial boot time. In the case where this operation were not available, the administrator would simply have to login to the guest and change the password in the normal manner, so this is just a convenient optimization. Therefore this operation is not considered mandatory for drivers to support.

  Driver Support:

  - **Hyper-V:** `missing`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `unknown`
  - **Libvirt KVM (ppc64):** `missing`
  - **Libvirt KVM (s390x):** `missing`
  - **Libvirt KVM (x86):** `complete` **Notes:** Requires libvirt>=1.2.16 and hw_qemu_guest_agent.
  - **Libvirt LXC:** `missing`

  - **Libvirt QEMU (x86):** `complete` **Notes:** Requires libvirt>=1.2.16 and hw_qemu_guest_agent.
  - **Libvirt Virtuozzo CT:** `complete` **Notes:** Requires libvirt>=2.0.0
  - **Libvirt Virtuozzo VM:** `complete` **Notes:** Requires libvirt>=2.0.0
  - **Libvirt Xen:** `missing`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `missing`
  - **XenServer:** `complete` **Notes:** Requires XenAPI agent on the guest.
  - **zVM:** `missing`

- **Save snapshot of instance disk Status: optional.**

  CLI commands:

  - `nova image-create <server> <name>`

  Notes: The snapshot operation allows the current state of the instance root disk to be saved and uploaded back into the glance image repository. The instance can later be booted again using this saved image. This is in effect making the ephemeral instance root disk into a semi-persistent storage, in so much as it is preserved even though the guest is no longer running. In general though, the expectation is that the root disks are ephemeral so the ability to take a snapshot cannot be assumed. Therefore this operation is not considered mandatory to support.

  Driver Support:

  - **Hyper-V:** `complete`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `complete`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `complete`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `partial` **Notes:** Only cold snapshots (pause + snapshot) supported
  - **PowerVM:** `complete` **Notes:** When using the localdisk disk driver, snapshot is only supported if I/O is being hosted by the management partition. If hosting I/O on traditional VIOS, we are limited by the fact that a VSCSI device cant be mapped to two partitions (the VIOS and the management) at once.
  - **VMware vCenter:** `complete`
  - **XenServer:** `complete`
  - **zVM:** `complete`

- **Suspend instance Status: optional.**

  CLI commands:

  - `nova suspend <server>`

Notes: Suspending an instance can be thought of as roughly equivalent to suspend-to-disk. The instance no longer consumes any RAM or CPUs, with its live running state having been preserved in a file on disk. It can later be restored, at which point it should continue execution where it left off. As with stopping instance CPUs, it suffers from the fact that the guest OS will typically be left with a clock that is no longer telling correct time. For container based virtualization solutions, this operation is particularly technically challenging to implement and is an area of active research. This operation tends to make more sense when thinking of instances as pets, rather than cattle, since with cattle it would be simpler to just terminate the instance instead of suspending. Therefore this operation is considered optional to support.

Driver Support:

- **Hyper-V:** `complete`
- **Ironic:** `missing`
- **Libvirt KVM (aarch64):** `complete`
- **Libvirt KVM (ppc64):** `complete`
- **Libvirt KVM (s390x):** `complete`
- **Libvirt KVM (x86):** `complete`
- **Libvirt LXC:** `missing`
- **Libvirt QEMU (x86):** `complete`
- **Libvirt Virtuozzo CT:** `complete`
- **Libvirt Virtuozzo VM:** `complete`
- **Libvirt Xen:** `complete`
- **PowerVM:** `missing`
- **VMware vCenter:** `complete`
- **XenServer:** `complete`
- **zVM:** `missing`

- **Swap block volumes Status: optional.**

  CLI commands:

  - `nova volume-update <server> <attachment> <volume>`

  Notes: The swap volume operation is a mechanism for changing a running instance so that its attached volume(s) are backed by different storage in the host. An alternative to this would be to simply terminate the existing instance and spawn a new instance with the new storage. In other words this operation is primarily targeted towards the pet use case rather than cattle, however, it is required for volume migration to work in the volume service. This is considered optional to support.

  Driver Support:

  - **Hyper-V:** `missing`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `unknown`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`

  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `missing`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `missing`
  - **XenServer:** `missing`
  - **zVM:** `missing`

- **Shutdown instance Status: mandatory.**

  CLI commands:

  - `nova delete <server>`

  Notes: The ability to terminate a virtual machine is required in order for a cloud user to stop utilizing resources and thus avoid indefinitely ongoing billing. Therefore this operation is mandatory to support in drivers.

  Driver Support:

  - **Hyper-V:** `complete`
  - **Ironic:** `complete`
  - **Libvirt KVM (aarch64):** `complete`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `complete` **Notes:** Fails in latest Ubuntu Trusty kernel from security repository (3.13.0-76-generic), but works in upstream 3.13.x kernels as well as default Ubuntu Trusty latest kernel (3.13.0-58-generic).
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `complete`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `complete`
  - **VMware vCenter:** `complete`
  - **XenServer:** `complete`
  - **zVM:** `complete`

- **Trigger crash dump Status: optional.**

  CLI commands:

  - `nova trigger-crash-dump <server>`

  Notes: The trigger crash dump operation is a mechanism for triggering a crash dump in an instance. The feature is typically implemented by injecting an NMI (Non-maskable Interrupt) into the instance. It provides a means to dump the production memory image as a dump file which is useful for users. Therefore this operation is considered optional to support.

Driver Support:

- **Hyper-V:** missing
- **Ironic:** complete
- **Libvirt KVM (aarch64):** unknown
- **Libvirt KVM (ppc64):** complete
- **Libvirt KVM (s390x):** complete
- **Libvirt KVM (x86):** complete
- **Libvirt LXC:** missing
- **Libvirt QEMU (x86):** complete
- **Libvirt Virtuozzo CT:** missing
- **Libvirt Virtuozzo VM:** missing
- **Libvirt Xen:** missing
- **PowerVM:** missing
- **VMware vCenter:** missing
- **XenServer:** missing
- **zVM:** missing

- **Resume instance CPUs (unpause) Status: optional.**

  CLI commands:

  - nova unpause <server>

  Notes: See notes for the Stop instance CPUs operation

  Driver Support:

  - **Hyper-V:** complete
  - **Ironic:** missing
  - **Libvirt KVM (aarch64):** complete
  - **Libvirt KVM (ppc64):** complete
  - **Libvirt KVM (s390x):** complete
  - **Libvirt KVM (x86):** complete
  - **Libvirt LXC:** complete
  - **Libvirt QEMU (x86):** complete
  - **Libvirt Virtuozzo CT:** complete
  - **Libvirt Virtuozzo VM:** complete
  - **Libvirt Xen:** complete
  - **PowerVM:** missing
  - **VMware vCenter:** missing
  - **XenServer:** complete
  - **zVM:** complete

- **uefi boot Status: optional.**

  Notes: This allows users to boot a guest with uefi firmware.

  Driver Support:

  - **Hyper-V:** `missing`
  - **Ironic:** `partial` **Notes:** depends on hardware support
  - **Libvirt KVM (aarch64):** `complete`
  - **Libvirt KVM (ppc64):** `missing`
  - **Libvirt KVM (s390x):** `missing`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `missing`
  - **Libvirt Virtuozzo VM:** `missing`
  - **Libvirt Xen:** `missing`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `complete`
  - **XenServer:** `missing`
  - **zVM:** `missing`

- **Device tags Status: optional.**

  CLI commands:

  - `nova boot`

  Notes: This allows users to set tags on virtual devices when creating a server instance. Device tags are used to identify virtual device metadata, as exposed in the metadata API and on the config drive. For example, a network interface tagged with nic1 will appear in the metadata along with its bus (ex: PCI), bus address (ex: 0000:00:02.0), MAC address, and tag (nic1). If multiple networks are defined, the order in which they appear in the guest operating system will not necessarily reflect the order in which they are given in the server boot request. Guests should therefore not depend on device order to deduce any information about their network devices. Instead, device role tags should be used. Device tags can be applied to virtual network interfaces and block devices.

  Driver Support:

  - **Hyper-V:** `complete`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `unknown`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `unknown`
  - **Libvirt QEMU (x86):** `complete`

  - **Libvirt Virtuozzo CT:** `unknown`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `missing`
  - **XenServer:** `complete`
  - **zVM:** `missing`

- **quiesce Status: optional.**

  Notes: Quiesce the specified instance to prepare for snapshots. For libvirt, guest filesystems will be frozen through qemu agent.

  Driver Support:

  - **Hyper-V:** `missing`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `unknown`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `missing`
  - **Libvirt Virtuozzo VM:** `missing`
  - **Libvirt Xen:** `missing`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `missing`
  - **XenServer:** `missing`
  - **zVM:** `missing`

- **unquiesce Status: optional.**

  Notes: See notes for the quiesce operation

  Driver Support:

  - **Hyper-V:** `missing`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `unknown`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`

  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `missing`
  - **Libvirt Virtuozzo VM:** `missing`
  - **Libvirt Xen:** `missing`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `missing`
  - **XenServer:** `missing`
  - **zVM:** `missing`

- **Attach block volume to multiple instances Status: optional.**

  CLI commands:

  - `nova volume-attach <server> <volume>`

  Notes: The multiattach volume operation is an extension to the attach volume operation. It allows to attach a single volume to multiple instances. This operation is not considered to be mandatory to support. Note that for the libvirt driver, this is only supported if qemu<2.10 or libvirt>=3.10.

  Driver Support:

  - **Hyper-V:** `missing`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `unknown`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `missing`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `missing`
  - **XenServer:** `missing`
  - **zVM:** `missing`

- **Attach encrypted block volume to server Status: optional.**

  CLI commands:

  - `nova volume-attach <server> <volume>`

  Notes: This is the same as the attach volume operation except with an encrypted block device. Encrypted volumes are controlled via admin-configured volume types in the block storage service. Since attach volume is optional this feature is also optional for compute drivers to support.

  Driver Support:

  - **Hyper-V:** `missing`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `unknown`
  - **Libvirt KVM (ppc64):** `unknown`
  - **Libvirt KVM (s390x):** `unknown`
  - **Libvirt KVM (x86):** `complete` **Notes:** For native QEMU decryption of the encrypted volume (and rbd support), QEMU>=2.6.0 and libvirt>=2.2.0 are required and only the luks type provider is supported. Otherwise both luks and cryptsetup types are supported but not natively, i.e. not all volume types are supported.
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `complete` **Notes:** The same restrictions apply as KVM x86.
  - **Libvirt Virtuozzo CT:** `missing`
  - **Libvirt Virtuozzo VM:** `unknown`
  - **Libvirt Xen:** `unknown`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `missing`
  - **XenServer:** `missing`
  - **zVM:** `missing`

- **Validate image with trusted certificates Status: optional.**

  CLI commands:

  - `nova boot --trusted-image-certificate-id ...`

  Notes: Since trusted image certification validation is configurable by the cloud deployer it is considered optional. However, it is a virt-agnostic feature so there is no good reason that all virt drivers cannot support the feature since it is mostly just plumbing user requests through the virt driver when downloading images.

  Driver Support:

  - **Hyper-V:** `missing`
  - **Ironic:** `missing`
  - **Libvirt KVM (aarch64):** `complete`
  - **Libvirt KVM (ppc64):** `complete`
  - **Libvirt KVM (s390x):** `complete`
  - **Libvirt KVM (x86):** `complete`
  - **Libvirt LXC:** `complete`
  - **Libvirt QEMU (x86):** `complete`
  - **Libvirt Virtuozzo CT:** `complete`
  - **Libvirt Virtuozzo VM:** `complete`
  - **Libvirt Xen:** `complete`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `missing`

  - **XenServer:** missing
  - **zVM:** missing

- **File backed memory Status: optional.**

  Notes: The file backed memory feature in Openstack allows a Nova node to serve guest memory from a file backing store. This mechanism uses the libvirt file memory source, causing guest instance memory to be allocated as files within the libvirt memory backing directory. This is only supported if qemu>2.6 and libvirt>4.0.0

  Driver Support:

  - **Hyper-V:** missing
  - **Ironic:** missing
  - **Libvirt KVM (aarch64):** unknown
  - **Libvirt KVM (ppc64):** unknown
  - **Libvirt KVM (s390x):** unknown
  - **Libvirt KVM (x86):** complete
  - **Libvirt LXC:** missing
  - **Libvirt QEMU (x86):** complete
  - **Libvirt Virtuozzo CT:** missing
  - **Libvirt Virtuozzo VM:** missing
  - **Libvirt Xen:** missing
  - **PowerVM:** missing
  - **VMware vCenter:** missing
  - **XenServer:** missing
  - **zVM:** missing

- **Report CPU traits Status: optional.**

  Notes: The report CPU traits feature in OpenStack allows a Nova node to report its CPU traits according to CPU mode configuration. This gives users the ability to boot instances based on desired CPU traits.

  Driver Support:

  - **Hyper-V:** missing
  - **Ironic:** missing
  - **Libvirt KVM (aarch64):** unknown
  - **Libvirt KVM (ppc64):** complete
  - **Libvirt KVM (s390x):** missing
  - **Libvirt KVM (x86):** complete
  - **Libvirt LXC:** missing
  - **Libvirt QEMU (x86):** complete
  - **Libvirt Virtuozzo CT:** missing
  - **Libvirt Virtuozzo VM:** missing
  - **Libvirt Xen:** missing

  - **PowerVM:** missing
  - **VMware vCenter:** missing
  - **XenServer:** missing
  - **zVM:** missing

- **SR-IOV ports with resource request Status: optional.**

  CLI commands:

  - nova boot --nic port-id <neutron port with resource request> ...

  Notes: To support neutron SR-IOV ports (vnic_type=direct or vnic_type=macvtap) with resource request the virt driver needs to include the parent_ifname key in each subdict which represents a VF under the pci_passthrough_devices key in the dict returned from the ComputeDriver.get_available_resource() call.

  Driver Support:

  - **Hyper-V:** missing
  - **Ironic:** missing
  - **Libvirt KVM (aarch64):** missing
  - **Libvirt KVM (ppc64):** missing
  - **Libvirt KVM (s390x):** missing
  - **Libvirt KVM (x86):** complete
  - **Libvirt LXC:** missing
  - **Libvirt QEMU (x86):** complete
  - **Libvirt Virtuozzo CT:** missing
  - **Libvirt Virtuozzo VM:** missing
  - **Libvirt Xen:** missing
  - **PowerVM:** missing
  - **VMware vCenter:** missing
  - **XenServer:** missing
  - **zVM:** missing

- **Boot instance with secure encrypted memory Status: optional.**

  CLI commands:

  - openstack server create <usual server create parameters>

  Notes: The feature allows VMs to be booted with their memory hardware-encrypted with a key specific to the VM, to help protect the data residing in the VM against access from anyone other than the user of the VM. The Configuration and Security Guides specify usage of this feature.

  Driver Support:

  - **Hyper-V:** missing
  - **Ironic:** missing
  - **Libvirt KVM (aarch64):** missing
  - **Libvirt KVM (ppc64):** missing

  - **Libvirt KVM (s390x):** `missing`
  - **Libvirt KVM (x86):** `partial` **Notes:** This feature is currently only available with hosts which support the SEV (Secure Encrypted Virtualization) technology from AMD.
  - **Libvirt LXC:** `missing`
  - **Libvirt QEMU (x86):** `missing`
  - **Libvirt Virtuozzo CT:** `missing`
  - **Libvirt Virtuozzo VM:** `missing`
  - **Libvirt Xen:** `missing`
  - **PowerVM:** `missing`
  - **VMware vCenter:** `missing`
  - **XenServer:** `missing`
  - **zVM:** `missing`

Notes:

- **This document is a continuous work in progress**

4.57 Upgrades

Nova aims to provide upgrades with minimal downtime.

Firstly, the data plane. There should be no VM downtime when you upgrade Nova. Nova has had this since the early days, with the exception of some nova-network related services.

Secondly, we want no downtime during upgrades of the Nova control plane. This document is trying to describe how we can achieve that.

Once we have introduced the key concepts relating to upgrade, we will introduce the process needed for a no downtime upgrade of nova.

4.57.1 Minimal Downtime Upgrade Process

Plan your upgrade

- Read and ensure you understand the release notes for the next release.
- You should ensure all required steps from the previous upgrade have been completed, such as data migrations.
- Make a backup of your database. Nova does not support downgrading of the database. Hence, in case of upgrade failure, restoring database from backup is the only choice.
- During upgrade be aware that there will be additional load on nova-conductor. You may find you need to add extra nova-conductor workers to deal with the additional upgrade related load.

Rolling upgrade process

To reduce downtime, the compute services can be upgraded in a rolling fashion. It means upgrading a few services at a time. This results in a condition where both old (N) and new (N+1) nova-compute services co-exist for a certain time period. Note that, there is no upgrade of the hypervisor here, this is just upgrading the nova services. If reduced

downtime is not a concern (or lower complexity is desired), all services may be taken down and restarted at the same time.

1. Before maintenance window:
    - Start the process with the controller node. Install the code for the next version of Nova, either in a venv or a separate control plane node, including all the python dependencies.
    - Using the newly installed nova code, run the DB sync. First run `nova-manage api_db sync`, then `nova-manage db sync`. In a multi-cell environment, `nova-manage db sync` must currently be run in each cell. These schema change operations should have minimal or no effect on performance, and should not cause any operations to fail.
    - At this point, new columns and tables may exist in the database. These DB schema changes are done in a way that both the N and N+1 release can perform operations against the same schema.
2. During maintenance window:
    - Several nova services rely on the external placement service being at the latest level. Therefore, you must upgrade placement before any nova services. See the placement upgrade notes for more details on upgrading the placement service.
    - For maximum safety (no failed API operations), gracefully shutdown all the services (i.e. SIG_TERM) except nova-compute.
    - Before restarting services with new code, perform the release-specific readiness check with `nova-status upgrade check`. See the *nova-status upgrade check* for more details on status check.
    - Start all services on the new code, with `[upgrade_levels]compute=auto` in nova.conf. It is safest to start nova-conductor first and nova-api last. Note that you may use a static alias name instead of `auto`, such as `[upgrade_levels]compute=<release_name>`. Also note that this step is only required if compute services are not upgraded in lock-step with the control services.
    - If desired, gracefully shutdown nova-compute (i.e. SIG_TERM) services in small batches, then start the new version of the code with: `[upgrade_levels]compute=auto`. If this batch-based approach is used, only a few compute nodes will have any delayed API actions, and to ensure there is enough capacity online to service any boot requests that happen during this time.
3. After maintenance window:
    - Once all services are running the new code, double check in the DB that there are no old orphaned service records using *nova service-list*.
    - Now that all services are upgraded, we need to send the SIG_HUP signal, so all the services clear any cached service version data. When a new service starts, it automatically detects which version of the compute RPC protocol to use, and it can decide if it is safe to do any online data migrations. Note, if you used a static value for the upgrade_level, such as `[upgrade_levels]compute=<release_name>`, you must update nova.conf to remove that configuration value and do a full service restart.
    - Now all the services are upgraded and signaled, the system is able to use the latest version of the RPC protocol and can access all of the features in the new release.
    - Once all the services are running the latest version of the code, and all the services are aware they all have been upgraded, it is safe to transform the data in the database into its new format. While some of this work happens on demand when the system reads a database row that needs updating, we must get all the data transformed into the current version before the next upgrade. Additionally, some data may not be transformed automatically so performing the data migration is necessary to avoid performance degradation due to compatibility routines.
    - This process can put significant extra write load on the database. Complete all online data migrations using: `nova-manage db online_data_migrations --max-count <number>`. Note that you can use the `--max-count` argument to reduce the load this operation will place on the database,

which allows you to run a small chunk of the migrations until all of the work is done. The chunk size you should use depends on your infrastructure and how much additional load you can impose on the database. To reduce load, perform smaller batches with delays between chunks. To reduce time to completion, run larger batches. Each time it is run, the command will show a summary of completed and remaining records. If using the `--max-count` option, the command should be rerun while it returns exit status 1 (which indicates that some migrations took effect, and more work may remain to be done), even if some migrations produce errors. If all possible migrations have completed and some are still producing errors, exit status 2 will be returned. In this case, the cause of the errors should be investigated and resolved. Migrations should be considered successfully completed only when the command returns exit status 0.

- At this point, you must also ensure you update the configuration, to stop using any deprecated features or options, and perform any required work to transition to alternative features. All the deprecated options should be supported for one cycle, but should be removed before your next upgrade is performed.

4.57.2 Current Database Upgrade Types

Currently Nova has 2 types of database upgrades that are in use.

1. Schema Migrations
2. Data Migrations

Schema Migrations

Schema migrations are defined in `nova/db/sqlalchemy/migrate_repo/versions` and in `nova/db/sqlalchemy/api_migrations/migrate_repo/versions`. They are the routines that transform our database structure, which should be additive and able to be applied to a running system before service code has been upgraded.

Note: The API database migrations should be assumed to run before the migrations for the main/cell databases. This is because the former contains information about how to find and connect to the latter. Some management commands that operate on multiple cells will attempt to list and iterate over cell mapping records, which require a functioning API database schema.

Data Migrations

Online data migrations occur in two places:

1. Inline migrations that occur as part of normal run-time activity as data is read in the old format and written in the new format
2. Background online migrations that are performed using `nova-manage` to complete transformations that will not occur incidentally due to normal runtime activity.

An example of online data migrations are the flavor migrations done as part of Nova object version 1.18. This included a transient migration of flavor storage from one database location to another.

Note: Database downgrades are not supported.

Migration policy:

The following guidelines for schema and data migrations are followed in order to ease upgrades:

- Additive schema migrations - In general, almost all schema migrations should be additive. Put simply, they should only create elements like columns, indices, and tables.
- Subtractive schema migrations - To remove an element like a column or table during the N release cycle:
  1. The element must be deprecated and retained for backward compatibility. (This allows for graceful upgrade from N to N+1.)
  2. Data migration, by the objects layer, must completely migrate data from the old version of the schema to the new version.
     - Data migration example
     - Data migration enforcement example (for sqlalchemy migrate/deprecated scripts):
  3. The column can then be removed with a migration at the start of N+2.
- All schema migrations should be idempotent. (For example, a migration should check if an element exists in the schema before attempting to add it.) This logic comes for free in the autogenerated workflow of the online migrations.
- Constraints - When adding a foreign or unique key constraint, the schema migration code needs to handle possible problems with data before applying the constraint. (Example: A unique constraint must clean up duplicate records before applying said constraint.)
- Data migrations - As mentioned above, data migrations will be done in an online fashion by custom code in the object layer that handles moving data between the old and new portions of the schema. In addition, for each type of data migration performed, there should exist a nova-manage option for an operator to manually request that rows be migrated.
  - See flavor migration spec for an example of data migrations in the object layer.

***Future* work -**

1. Adding plumbing to enforce that relevant data migrations are completed before running *contract* in the expand/migrate/contract schema migration workflow. A potential solution would be for *contract* to run a gating test for each specific subtract operation to determine if the operation can be completed.

4.57.3 Concepts

Here are the key concepts you need to know before reading the section on the upgrade process:

RPC version pinning Through careful RPC versioning, newer nodes are able to talk to older nova-compute nodes. When upgrading control plane nodes, we can pin them at an older version of the compute RPC API, until all the compute nodes are able to be upgraded. https://wiki.openstack.org/wiki/RpcMajorVersionUpdates

> **Note:** The procedure for rolling upgrades with multiple cells v2 cells is not yet determined.

Online Configuration Reload During the upgrade, we pin new serves at the older RPC version. When all services are updated to use newer code, we need to unpin them so we are able to use any new functionality. To avoid having to restart the service, using the current SIGHUP signal handling, or otherwise, ideally we need a way to update the currently running process to use the latest configuration.

Graceful service shutdown Many nova services are python processes listening for messages on a AMQP queue, including nova-compute. When sending the process the SIGTERM the process stops getting new work from its

queue, completes any outstanding work, then terminates. During this process, messages can be left on the queue for when the python process starts back up. This gives us a way to shutdown a service using older code, and start up a service using newer code with minimal impact. If its a service that can have multiple workers, like nova-conductor, you can usually add the new workers before the graceful shutdown of the old workers. In the case of singleton services, like nova-compute, some actions could be delayed during the restart, but ideally no actions should fail due to the restart.

Note: While this is true for the RabbitMQ RPC backend, we need to confirm what happens for other RPC backends.

API load balancer draining When upgrading API nodes, you can make your load balancer only send new connections to the newer API nodes, allowing for a seamless update of your API nodes.

Expand/Contract DB Migrations Modern databases are able to make many schema changes while you are still writing to the database. Taking this a step further, we can make all DB changes by first adding the new structures, expanding. Then you can slowly move all the data into a new location and format. Once that is complete, you can drop bits of the scheme that are no long needed, i.e. contract. This happens multiple cycles after we have stopped using a particular piece of schema, and can happen in a schema migration without affecting runtime code.

Online Data Migrations using objects In Kilo we are moving all data migration into the DB objects code. When trying to migrate data in the database from the old format to the new format, this is done in the object code when reading or saving things that are in the old format. For records that are not updated, you need to run a background process to convert those records into the newer format. This process must be completed before you contract the database schema.

DB prune deleted rows Currently resources are soft deleted in the main database, so users are able to track instances in the DB that are created and destroyed in production. However, most people have a data retention policy, of say 30 days or 90 days after which they will want to delete those entries. Not deleting those entries affects DB performance as indices grow very large and data migrations take longer as there is more data to migrate.

nova-conductor object backports RPC pinning ensures new services can talk to the older services method signatures. But many of the parameters are objects that may well be too new for the old service to understand, so you are able to send the object back to the nova-conductor to be downgraded to a version the older service can understand.

4.57.4 Testing

Once we have all the pieces in place, we hope to move the Grenade testing to follow this new pattern.

The current tests only cover the existing upgrade process where:

- old computes can run with new control plane
- but control plane is turned off for DB migrations

4.58 Using WSGI with Nova

Though the compute and metadata APIs can be run using independent scripts that provide eventlet-based HTTP servers, it is generally considered more performant and flexible to run them using a generic HTTP server that supports WSGI (such as Apache or nginx).

The nova project provides two automatically generated entry points that support this: `nova-api-wsgi` and `nova-metadata-wsgi`. These read `nova.conf` and `api-paste.ini` and generate the required module-level

`application` that most WSGI servers require. If nova is installed using pip, these two scripts will be installed into whatever the expected `bin` directory is for the environment.

The new scripts replace older experimental scripts that could be found in the `nova/wsgi` directory of the code repository. The new scripts are *not* experimental.

When running the compute and metadata services with WSGI, sharing the compute and metadata service in the same process is not supported (as it is in the eventlet-based scripts).

In devstack as of May 2017, the compute and metadata APIs are hosted by a Apache communicating with uwsgi via mod_proxy_uwsgi. Inspecting the configuration created there can provide some guidance on one option for managing the WSGI scripts. It is important to remember, however, that one of the major features of using WSGI is that there are many different ways to host a WSGI application. Different servers make different choices about performance and configurability.

CHAPTER

FIVE

SEARCH

- Nova document search: Search the contents of this document.
- OpenStack wide search: Search the wider set of OpenStack documentation, including forums.