python-novaclient Documentation

Release 17.7.1.dev3

OpenStack Foundation

Jan 10, 2023

CONTENTS

This is a client for OpenStack Nova API. Theres a *Python API* (the `novaclient` module), and a *command-line script* (installed as **`nova`**). Each implements the entire OpenStack Nova API.

Youll need credentials for an OpenStack cloud that implements the Compute API, such as TryStack, HP, or Rackspace, in order to use the nova client.

See also:

You may want to read the OpenStack Compute API Guide to get an idea of the concepts. By understanding the concepts this library should make more sense.

CHAPTER

ONE

USER GUIDE

1.1 The `nova` Shell Utility

The **`nova`** shell utility interacts with OpenStack Nova API from the command line. It supports the entirety of the OpenStack Nova API.

Youll need to provide **`nova`** with your OpenStack Keystone user information. You can do this with the *os-username*, *os-password*, *os-project-name* (*os-project-id*), *os-project-domain-name* (*os-project-domain-id*) and *os-user-domain-name* (*os-user-domain-id*) options, but its easier to just set them as environment variables by setting some environment variables:

`OS_USERNAME`
    Your OpenStack Keystone user name.

`OS_PASSWORD`
    Your password.

`OS_PROJECT_NAME`
    The name of project for work.

`OS_PROJECT_ID`
    The ID of project for work.

`OS_PROJECT_DOMAIN_NAME`
    The name of domain containing the project.

`OS_PROJECT_DOMAIN_ID`
    The ID of domain containing the project.

`OS_USER_DOMAIN_NAME`
    The users domain name.

`OS_USER_DOMAIN_ID`
    The users domain ID.

`OS_AUTH_URL`
    The OpenStack Keystone endpoint URL.

`OS_COMPUTE_API_VERSION`
    The OpenStack Nova API version (microversion).

`OS_REGION_NAME`
    The Keystone region name. Defaults to the first region if multiple regions are available.

OS_TRUSTED_IMAGE_CERTIFICATE_IDS

A comma-delimited list of trusted image certificate IDs. Only used with the `nova boot` and `nova rebuild` commands starting with the 2.63 microversion.

For example:

```
export OS_TRUSTED_IMAGE_CERTIFICATE_IDS=trusted-cert-id1,trusted-cert-id2
```

For example, in Bash youd use:

```
export OS_USERNAME=yourname
export OS_PASSWORD=yadayadayada
export OS_PROJECT_NAME=myproject
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_AUTH_URL=http://<url-to-openstack-keystone>/identity
export OS_COMPUTE_API_VERSION=2.1
```

From there, all shell commands take the form:

```
nova <command> [arguments...]
```

Run **nova help** to get a full list of all possible commands, and run **nova help <command>** to get detailed help for that command.

For more information, see *the command reference*.

1.2 The novaclient Python API

1.2.1 Usage

First create a client instance with your credentials:

```
>>> from novaclient import client
>>> nova = client.Client(VERSION, USERNAME, PASSWORD, PROJECT_ID, AUTH_URL)
```

Here `VERSION` can be a string or `novaclient.api_versions.APIVersion` obj. If you prefer string value, you can use `1.1` (deprecated now), `2` or `2.X` (where X is a microversion).

Alternatively, you can create a client instance using the keystoneauth session API:

```
>>> from keystoneauth1 import loading
>>> from keystoneauth1 import session
>>> from novaclient import client
>>> loader = loading.get_plugin_loader('password')
>>> auth = loader.load_from_options(auth_url=AUTH_URL,
...                                 username=USERNAME,
...                                 password=PASSWORD,
...                                 project_id=PROJECT_ID)
>>> sess = session.Session(auth=auth)
>>> nova = client.Client(VERSION, session=sess)
```

If you have PROJECT_NAME instead of a PROJECT_ID, use the project_name parameter. Similarly, if your cloud uses keystone v3 and you have a DOMAIN_NAME or DOMAIN_ID, provide it as *user_domain_(name|id)* and if you are using a PROJECT_NAME also provide the domain information as *project_domain_(name|id)*.

novaclient adds python-novaclient and its version to the user-agent string that keystoneauth produces. If you are creating an application using novaclient and want to register a name and version in the user-agent string, pass those to the Session:

```
>>> sess = session.Session(
...     auth=auth, app_name'nodepool', app_version'1.2.3')
```

If you are making a library that consumes novaclient but is not an end-user application, you can append a (name, version) tuple to the sessions *additional_user_agent* property:

```
>>> sess = session.Session(auth=auth)
>>> sess.additional_user_agent.append(('shade', '1.2.3'))
```

For more information on this keystoneauth API, see Using Sessions.

It is also possible to use an instance as a context manager in which case there will be a session kept alive for the duration of the with statement:

```
>>> from novaclient import client
>>> with client.Client(VERSION, USERNAME, PASSWORD,
...                    PROJECT_ID, AUTH_URL) as nova:
...     nova.servers.list()
...     nova.flavors.list()
...
```

It is also possible to have a permanent (process-long) connection pool, by passing a connection_pool=True:

```
>>> from novaclient import client
>>> nova = client.Client(VERSION, USERNAME, PASSWORD, PROJECT_ID,
...                      AUTH_URL, connection_pool=True)
```

Then call methods on its managers:

```
>>> nova.servers.list()
[<Server: buildslave-ubuntu-9.10>]

>>> nova.flavors.list()
[<Flavor: 256 server>,
 <Flavor: 512 server>,
 <Flavor: 1GiB server>,
 <Flavor: 2GiB server>,
 <Flavor: 4GiB server>,
 <Flavor: 8GiB server>,
 <Flavor: 15.5GiB server>]

>>> fl = nova.flavors.find(ram=512)
```

(continues on next page)

(continued from previous page)

```
>>> nova.servers.create("my-server", flavor=fl)
<Server: my-server>
```

Warning: Direct initialization of `novaclient.v2.client.Client` object can cause you to shoot yourself in the foot. See launchpad bug-report 1493576 for more details.

1.2.2 Reference

See *the module reference*.

CHAPTER TWO

CLI REFERENCE

2.1 nova

The nova client is the command-line interface (CLI) for the Compute service (nova) API and its extensions.

For help on a specific **nova** command, enter:

```
$ nova help COMMAND
```

Note: Over time, command line functionality will be phased out of the nova CLI and into the openstack CLI. Using the openstack client where possible is preferred but there is not full parity yet for all of the nova commands. For information on using the openstack CLI, see OpenStackClient.

2.1.1 nova usage

```
usage: nova [--version] [--debug] [--os-cache] [--timings]
            [--os-region-name <region-name>] [--service-type <service-type>]
            [--service-name <service-name>]
            [--os-endpoint-type <endpoint-type>]
            [--os-compute-api-version <compute-api-ver>]
            [--os-endpoint-override <bypass-url>] [--profile HMAC_KEY]
            [--insecure] [--os-cacert <ca-certificate>]
            [--os-cert <certificate>] [--os-key <key>] [--timeout <seconds>]
            [--collect-timing] [--os-auth-type <name>]
            [--os-auth-url OS_AUTH_URL] [--os-system-scope OS_SYSTEM_SCOPE]
            [--os-domain-id OS_DOMAIN_ID] [--os-domain-name OS_DOMAIN_NAME]
            [--os-project-id OS_PROJECT_ID]
            [--os-project-name OS_PROJECT_NAME]
            [--os-project-domain-id OS_PROJECT_DOMAIN_ID]
            [--os-project-domain-name OS_PROJECT_DOMAIN_NAME]
            [--os-trust-id OS_TRUST_ID]
            [--os-default-domain-id OS_DEFAULT_DOMAIN_ID]
            [--os-default-domain-name OS_DEFAULT_DOMAIN_NAME]
            [--os-user-id OS_USER_ID] [--os-username OS_USERNAME]
            [--os-user-domain-id OS_USER_DOMAIN_ID]
            [--os-user-domain-name OS_USER_DOMAIN_NAME]
```

(continues on next page)

(continued from previous page)

```
            [--os-password OS_PASSWORD]
            <subcommand> ...
```

Subcommands:

`add-fixed-ip` **DEPRECATED** Add new IP address on a network to server.

`add-secgroup` Add a Security Group to a server.

`agent-create` Create new agent build.

`agent-delete` Delete existing agent build.

`agent-list` List all builds.

`agent-modify` Modify existing agent build.

`aggregate-add-host` Add the host to the specified aggregate.

`aggregate-cache-images` Request images be pre-cached on hosts within an aggregate. (Supported by API versions 2.81 - 2.latest)

`aggregate-create` Create a new aggregate with the specified details.

`aggregate-delete` Delete the aggregate.

`aggregate-list` Print a list of all aggregates.

`aggregate-remove-host` Remove the specified host from the specified aggregate.

`aggregate-set-metadata` Update the metadata associated with the aggregate.

`aggregate-show` Show details of the specified aggregate.

`aggregate-update` Update the aggregates name and optionally availability zone.

`availability-zone-list` List all the availability zones.

`backup` Backup a server by creating a backup type snapshot.

`boot` Boot a new server.

`clear-password` Clear the admin password for a server from the metadata server. This action does not actually change the instance server password.

`cloudpipe-configure` **DEPRECATED** Update the VPN IP/port of a cloudpipe instance.

`cloudpipe-create` **DEPRECATED** Create a cloudpipe instance for the given project.

`cloudpipe-list` **DEPRECATED** Print a list of all cloudpipe instances.

`console-log` Get console log output of a server.

`delete` Immediately shut down and delete specified server(s).

`diagnostics` Retrieve server diagnostics.

`evacuate` Evacuate server from failed host.

`flavor-access-add` Add flavor access for the given tenant.

`flavor-access-list` Print access information about the given flavor.

`flavor-access-remove` Remove flavor access for the given tenant.

`flavor-create` Create a new flavor.

`flavor-delete` Delete a specific flavor

`flavor-key` Set or unset extra_spec for a flavor.

`flavor-list` Print a list of available flavors (sizes of servers).

`flavor-show` Show details about the given flavor.

`flavor-update` Update the description of an existing flavor. (Supported by API versions 2.55 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`floating-ip-associate` **DEPRECATED** Associate a floating IP address to a server.

`floating-ip-disassociate` **DEPRECATED** Disassociate a floating IP address from a server.

`force-delete` Force delete a server.

`get-mks-console` Get an MKS console to a server. (Supported by API versions 2.8 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`get-password` Get the admin password for a server. This operation calls the metadata service to query metadata information and does not read password information from the server itself.

`get-rdp-console` Get a rdp console to a server.

`get-serial-console` Get a serial console to a server.

`get-spice-console` Get a spice console to a server.

`get-vnc-console` Get a vnc console to a server.

`host-action` **DEPRECATED** Perform a power action on a host.

`host-describe` **DEPRECATED** Describe a specific host.

`host-evacuate` Evacuate all instances from failed host.

`host-evacuate-live` Live migrate all instances off the specified host to other available hosts.

`host-list` **DEPRECATED** List all hosts by service.

`host-meta` Set or Delete metadata on all instances of a host.

`host-servers-migrate` Cold migrate all instances off the specified host to other available hosts.

`host-update` **DEPRECATED** Update host settings.

`hypervisor-list` List hypervisors. (Supported by API versions 2.0 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`hypervisor-servers` List servers belonging to specific hypervisors.

`hypervisor-show` Display the details of the specified hypervisor.

`hypervisor-stats` Get hypervisor statistics over all compute nodes.

`hypervisor-uptime` Display the uptime of the specified hypervisor.

`image-create` Create a new image by taking a snapshot of a running server.

`instance-action` Show an action.

`instance-action-list` List actions on a server.

`instance-usage-audit-log` List/Get server usage audits.

`interface-attach` Attach a network interface to a server.

`interface-detach` Detach a network interface from a server.

`interface-list` List interfaces attached to a server.

`keypair-add` Create a new key pair for use with servers.

`keypair-delete` Delete keypair given by its name. (Supported by API versions 2.0 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`keypair-list` Print a list of keypairs for a user (Supported by API versions 2.0 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`keypair-show` Show details about the given keypair. (Supported by API versions 2.0 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`limits` Print rate and absolute limits.

`list` List servers.

`list-secgroup` List Security Group(s) of a server.

`live-migration` Migrate running server to a new machine.

`live-migration-abort` Abort an on-going live migration. (Supported by API versions 2.24 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`live-migration-force-complete` Force on-going live migration to complete. (Supported by API versions 2.22 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`lock` Lock a server. A normal (non-admin) user will not be able to execute actions on a locked server.

`meta` Set or delete metadata on a server.

`migrate` Migrate a server. The new host will be selected by the scheduler.

`migration-list` Print a list of migrations.

`pause` Pause a server.

`quota-class-show` List the quotas for a quota class.

`quota-class-update` Update the quotas for a quota class. (Supported by API versions 2.0 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`quota-defaults` List the default quotas for a tenant.

`quota-delete` Delete quota for a tenant/user so their quota will Revert back to default.

`quota-show` List the quotas for a tenant/user.

`quota-update` Update the quotas for a tenant/user. (Supported by API versions 2.0 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`reboot` Reboot a server.

`rebuild` Shutdown, re-image, and re-boot a server.

`refresh-network` Refresh server network information.

`remove-fixed-ip` **DEPRECATED** Remove an IP address from a server.

`remove-secgroup` Remove a Security Group from a server.

`rescue` Reboots a server into rescue mode, which starts the machine from either the initial image or a specified image, attaching the current boot disk as secondary.

`reset-network` Reset network of a server.

`reset-state` Reset the state of a server.

`resize` Resize a server.

`resize-confirm` Confirm a previous resize.

`resize-revert` Revert a previous resize (and return to the previous VM).

`restore` Restore a soft-deleted server.

`resume` Resume a server.

`server-group-create` Create a new server group with the specified details.

`server-group-delete` Delete specific server group(s).

`server-group-get` Get a specific server group.

`server-group-list` Print a list of all server groups.

`server-migration-list` Get the migrations list of specified server. (Supported by API versions 2.23 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`server-migration-show` Get the migration of specified server. (Supported by API versions 2.23 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`server-tag-add` Add one or more tags to a server. (Supported by API versions 2.26 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`server-tag-delete` Delete one or more tags from a server. (Supported by API versions 2.26 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`server-tag-delete-all` Delete all tags from a server. (Supported by API versions 2.26 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`server-tag-list` Get list of tags from a server. (Supported by API versions 2.26 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`server-tag-set` Set list of tags to a server. (Supported by API versions 2.26 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`server-topology` Retrieve NUMA topology of the given server. (Supported by API versions 2.78 - 2.latest)

`service-delete` Delete the service.

`service-disable` Disable the service.

`service-enable` Enable the service.

`service-force-down` Force service to down. (Supported by API versions 2.11 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`service-list` Show a list of all running services. Filter by host & binary.

`set-password` Change the admin password for a server.

`shelve` Shelve a server.

`shelve-offload` Remove a shelved server from the compute node.

`show` Show details about the given server.

`ssh` SSH into a server.

`start` Start the server(s).

`stop` Stop the server(s).

`suspend` Suspend a server.

`trigger-crash-dump` Trigger crash dump in an instance. (Supported by API versions 2.17 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

`unlock` Unlock a server.

`unpause` Unpause a server.

`unrescue` Restart the server from normal boot disk again.

`unshelve` Unshelve a server.

`update` Update the name or the description for a server.

`usage` Show usage data for a single tenant.

`usage-list` List usage data for all tenants.

`version-list` List all API versions.

`virtual-interface-list` **DEPRECATED** Show virtual interface info about the given server.

`volume-attach` Attach a volume to a server.

`volume-attachments` List all the volumes attached to a server.

`volume-detach` Detach a volume from a server.

`volume-update` Update the attachment on the server. Migrates the data from an attached volume to the specified available volume and swaps out the active attachment to the new volume. Since microversion 2.85, support for updating the `delete_on_termination` delete flag, which allows changing the behavior of volume deletion on instance deletion.

`x509-create-cert` **DEPRECATED** Create x509 cert for a user in tenant.

`x509-get-root-cert` **DEPRECATED** Fetch the x509 root cert.

`bash-completion` Prints all of the commands and options to stdout so that the nova.bash_completion script doesnt have to hard code them.

`help` Display help about this program or one of its subcommands.

2.1.2 nova optional arguments

`--version` show programs version number and exit

`--debug` Print debugging output.

`--os-cache` Use the auth token cache. Defaults to False if `env[OS_CACHE]` is not set.

`--timings` Print call timing info.

`--os-region-name <region-name>` Defaults to `env[OS_REGION_NAME]`.

`--service-type <service-type>` Defaults to compute for most actions.

--service-name <service-name> Defaults to `env[NOVA_SERVICE_NAME]`.

--os-endpoint-type <endpoint-type> Defaults to `env[NOVA_ENDPOINT_TYPE]`, `env[OS_ENDPOINT_TYPE]` or publicURL.

--os-compute-api-version <compute-api-ver> Accepts X, X.Y (where X is major and Y is minor part) or X.latest, defaults to `env[OS_COMPUTE_API_VERSION]`.

--os-endpoint-override <bypass-url> Use this API endpoint instead of the Service Catalog. Defaults to `env[OS_ENDPOINT_OVERRIDE]`.

--profile HMAC_KEY HMAC key to use for encrypting context data for performance profiling of operation. This key should be the value of the HMAC key configured for the OSprofiler middleware in nova; it is specified in the Nova configuration file at /etc/nova/nova.conf. Without the key, profiling will not be triggered even if OSprofiler is enabled on the server side.

--os-auth-type <name>, --os-auth-plugin <name> Authentication type to use

nova add-secgroup

```
usage: nova add-secgroup <server> <secgroup>
```

Add a Security Group to a server.

Positional arguments:

<server> Name or ID of server.

<secgroup> Name or ID of Security Group.

nova agent-create

```
usage: nova agent-create <os> <architecture> <version> <url> <md5hash>
                         <hypervisor>
```

Create new agent build.

Positional arguments:

<os> Type of OS.

<architecture> Type of architecture.

<version> Version.

<url> URL.

<md5hash> MD5 hash.

<hypervisor> Type of hypervisor.

nova agent-delete

```
usage: nova agent-delete <id>
```

Delete existing agent build.

Positional arguments:

<id> ID of the agent-build.

nova agent-list

```
usage: nova agent-list [--hypervisor <hypervisor>]
```

List all builds.

Optional arguments:

--hypervisor <hypervisor> Type of hypervisor.

nova agent-modify

```
usage: nova agent-modify <id> <version> <url> <md5hash>
```

Modify existing agent build.

Positional arguments:

<id> ID of the agent-build.

<version> Version.

<url> URL

<md5hash> MD5 hash.

nova aggregate-add-host

```
usage: nova aggregate-add-host <aggregate> <host>
```

Add the host to the specified aggregate.

Positional arguments:

<aggregate> Name or ID of aggregate.

<host> The host to add to the aggregate.

nova aggregate-cache-images

```
usage: nova aggregate-cache-images <aggregate> <image> [<image> ..]
```

Request image(s) be pre-cached on hosts within the aggregate. (Supported by API versions 2.81 - 2.latest)

New in version 16.0.0.

Positional arguments:

`<aggregate>` Name or ID of aggregate.

`<image>` Name or ID of image(s) to cache.

nova aggregate-create

```
usage: nova aggregate-create <name> [<availability-zone>]
```

Create a new aggregate with the specified details.

Positional arguments:

`<name>` Name of aggregate.

`<availability-zone>` The availability zone of the aggregate (optional).

nova aggregate-delete

```
usage: nova aggregate-delete <aggregate>
```

Delete the aggregate.

Positional arguments:

`<aggregate>` Name or ID of aggregate to delete.

nova aggregate-list

```
usage: nova aggregate-list
```

Print a list of all aggregates.

nova aggregate-remove-host

```
usage: nova aggregate-remove-host <aggregate> <host>
```

Remove the specified host from the specified aggregate.

Positional arguments:

`<aggregate>` Name or ID of aggregate.

`<host>` The host to remove from the aggregate.

nova aggregate-set-metadata

```
usage: nova aggregate-set-metadata <aggregate> <key=value> [<key=value> ...]
```

Update the metadata associated with the aggregate.

Positional arguments:

`<aggregate>` Name or ID of aggregate to update.

`<key=value>` Metadata to add/update to aggregate. Specify only the key to delete a metadata item.

nova aggregate-show

```
usage: nova aggregate-show <aggregate>
```

Show details of the specified aggregate.

Positional arguments:

`<aggregate>` Name or ID of aggregate.

nova aggregate-update

```
usage: nova aggregate-update [--name NAME]
                             [--availability-zone <availability-zone>]
                             <aggregate>
```

Update the aggregates name and optionally availability zone.

Positional arguments:

`<aggregate>` Name or ID of aggregate to update.

Optional arguments:

`--name NAME` New name for aggregate.

`--availability-zone <availability-zone>` New availability zone for aggregate.

nova availability-zone-list

```
usage: nova availability-zone-list
```

List all the availability zones.

nova backup

```
usage: nova backup <server> <name> <backup-type> <rotation>
```

Backup a server by creating a backup type snapshot.

Positional arguments:

<server> Name or ID of server.

<name> Name of the backup image.

<backup-type> The backup type, like daily or weekly.

<rotation> Int parameter representing how many backups to keep around.

nova boot

```
usage: nova boot [--flavor <flavor>] [--image <image>]
                 [--image-with <key=value>] [--boot-volume <volume_id>]
                 [--snapshot <snapshot_id>] [--min-count <number>]
                 [--max-count <number>] [--meta <key=value>]
                 [--key-name <key-name>] [--user-data <user-data>]
                 [--availability-zone <availability-zone>]
                 [--security-groups <security-groups>]
                 [--block-device-mapping <dev-name=mapping>]
                 [--block-device key1=value1[,key2=value2...]]
                 [--swap <swap_size>]
                 [--ephemeral size=<size>[,format=<format>]]
                 [--hint <key=value>]
                 [--nic <auto,none,net-id=net-uuid,net-name=network-name,port-
↪id=port-uuid,v4-fixed-ip=ip-addr,v6-fixed-ip=ip-addr,tag=tag>]
                 [--config-drive <value>] [--poll] [--admin-pass <value>]
                 [--access-ip-v4 <value>] [--access-ip-v6 <value>]
                 [--description <description>] [--tags <tags>]
                 [--return-reservation-id]
                 [--trusted-image-certificate-id <trusted-image-certificate-
↪id>]
                 [--host <host>]
                 [--hypervisor-hostname <hypervisor-hostname>]
                 [--hostname <hostname>]
                 <name>
```

Boot a new server.

In order to create a server with pre-existing ports that contain a `resource_request` value, such as for guaranteed minimum bandwidth quality of service support, microversion `2.72` is required.

Positional arguments:

<name> Name for the new server.

Optional arguments:

--flavor <flavor> Name or ID of flavor (see nova flavor-list).

--image <image> Name or ID of image (see glance image-list).

--image-with <key=value> Image metadata property (see glance image-show).

--boot-volume <volume_id> Volume ID to boot from.

--snapshot <snapshot_id> Snapshot ID to boot from (will create a volume).

--min-count <number> Boot at least <number> servers (limited by quota).

--max-count <number> Boot up to <number> servers (limited by quota).

--meta <key=value> Record arbitrary key/value metadata to /meta_data.json on the metadata server. Can be specified multiple times.

--key-name <key-name> Key name of keypair that should be created earlier with the command keypair-add.

--user-data <user-data> user data file to pass to be exposed by the metadata server.

--availability-zone <availability-zone> The availability zone for server placement.

--security-groups <security-groups> Comma separated list of security group names.

--block-device-mapping <dev-name=mapping> Block device mapping in the format <dev-name>=<id>:<type>:<size(GiB)>:<delete-on-terminate>.

--block-device key1=value1[,key2=value2] Block device mapping with the keys: id=UUID (image_id, snapshot_id or volume_id only if using source image, snapshot or volume) source=source type (image, snapshot, volume or blank), dest=destination type of the block device (volume or local), bus=devices bus (e.g. uml, lxc, virtio, ; if omitted, hypervisor driver chooses a suitable default, honoured only if device type is supplied) type=device type (e.g. disk, cdrom, ; defaults to disk) device=name of the device (e.g. vda, xda, ; if omitted, hypervisor driver chooses suitable device depending on selected bus; note the libvirt driver always uses default device names), size=size of the block device in MiB(for swap) and in GiB(for other formats) (if omitted, hypervisor driver calculates size), format=device will be formatted (e.g. swap, ntfs, ; optional), bootindex=integer used for ordering the boot disks (for image backed instances it is equal to 0, for others need to be specified), shutdown=shutdown behaviour (either preserve or remove, for local destination set to remove), tag=device metadata tag (optional; supported by API versions 2.42 - 2.latest), and volume_type=type of volume to create (either ID or name) when source is *blank*, *image* or *snapshot* and dest is *volume* (optional; supported by API versions 2.67 - 2.latest).

--swap <swap_size> Create and attach a local swap block device of <swap_size> MiB.

--ephemeral size=<size>[,format=<format>] Create and attach a local ephemeral block device of <size> GiB and format it to <format>.

--hint <key=value> Send arbitrary key/value pairs to the scheduler for custom use.

--nic <auto,none,net-id=net-uuid,net-name=network-name,port-id=port-uuid,v4-fixed-ip=ip-addr
Create a NIC on the server. Specify option multiple times to create multiple nics unless using the special auto or none values. auto: automatically allocate network resources if none are available. This cannot be specified with any other nic value and cannot be specified multiple times. none: do not attach a NIC at all. This cannot be specified with any other nic value and cannot be specified multiple times. net-id: attach NIC to network with a specific UUID. net-name: attach NIC to network with this name (either port-id or net-id or net-name must be provided), v4-fixed-ip: IPv4 fixed address for NIC (optional), v6-fixed-ip: IPv6 fixed address for NIC (optional), port-id: attach NIC to port with this UUID tag: interface metadata tag (optional) (either port-id or net-id must be provided). (Supported by API versions 2.42 - 2.latest)

--config-drive <value> Enable config drive. The value must be a boolean value.

--poll Report the new server boot progress until it completes.

--admin-pass <value> Admin password for the instance.

--access-ip-v4 <value> Alternative access IPv4 of the instance.

--access-ip-v6 <value> Alternative access IPv6 of the instance.

--description <description> Description for the server. (Supported by API versions 2.19 - 2.latest)

--tags <tags> Tags for the server.Tags must be separated by commas: tags <tag1,tag2> (Supported by API versions 2.52 - 2.latest)

--return-reservation-id Return a reservation id bound to created servers.

--trusted-image-certificate-id <trusted-image-certificate-id> Trusted image certificate IDs used to validate certificates during the image signature verification process. Defaults to env[OS_TRUSTED_IMAGE_CERTIFICATE_IDS]. May be specified multiple times to pass multiple trusted image certificate IDs. (Supported by API versions 2.63 - 2.latest)

--host <host> Requested host to create servers. Admin only by default. (Supported by API versions 2.74 - 2.latest)

--hypervisor-hostname <hypervisor-hostname> Requested hypervisor hostname to create servers. Admin only by default. (Supported by API versions 2.74 - 2.latest)

--hostname <hostname> Hostname for the instance. This sets the hostname stored in the metadata server: a utility such as cloud-init running on the guest is required to propagate these changes to the guest. (Supported by API versions 2.90 - 2.latest)

nova clear-password

```
usage: nova clear-password <server>
```

Clear the admin password for a server from the metadata server. This action does not actually change the instance server password.

Positional arguments:

<server> Name or ID of server.

nova console-log

```
usage: nova console-log [--length <length>] <server>
```

Get console log output of a server.

Locale encoding issues

If you encounter an error such as:

```
UnicodeEncodeError: 'ascii' codec can't encode characters in position
```

The solution to these problems is different depending on which locale your computer is running in.

For instance, if you have a German Linux machine, you can fix the problem by exporting the locale to de_DE.utf-8:

```
export LC_ALL=de_DE.utf-8
export LANG=de_DE.utf-8
```

If you are on a US machine, en_US.utf-8 is the encoding of choice. On some newer Linux systems, you could also try C.UTF-8 as the locale:

```
export LC_ALL=C.UTF-8
export LANG=C.UTF-8
```

Positional arguments:

`<server>` Name or ID of server.

Optional arguments:

`--length <length>` Length in lines to tail.

nova delete

```
usage: nova delete [--all-tenants] <server> [<server> ...]
```

Immediately shut down and delete specified server(s).

Positional arguments:

`<server>` Name or ID of server(s).

Optional arguments:

`--all-tenants` Delete server(s) in another tenant by name (Admin only).

nova diagnostics

```
usage: nova diagnostics <server>
```

Retrieve server diagnostics.

Positional arguments:

`<server>` Name or ID of server.

nova evacuate

```
usage: nova evacuate [--password <password>] [--on-shared-storage] [--force]
↪<server> [<host>]
```

Evacuate server from failed host.

Positional arguments:

<server> Name or ID of server.

<host> Name or ID of the target host. If no host is specified, the scheduler will choose one.

Optional arguments:

--password <password> Set the provided admin password on the evacuated server. Not applicable if the server is on shared storage.

--on-shared-storage Specifies whether server files are located on shared storage. (Supported by API versions 2.0 - 2.13)

--force Force an evacuation by not verifying the provided destination host by the scheduler. (Supported by API versions 2.29 - 2.67)

> **Warning:** This could result in failures to actually evacuate the server to the specified host. It is recommended to either not specify a host so that the scheduler will pick one, or specify a host without `--force`.

nova flavor-access-add

```
usage: nova flavor-access-add <flavor> <tenant_id>
```

Add flavor access for the given tenant.

Positional arguments:

<flavor> Flavor name or ID to add access for the given tenant.

<tenant_id> Tenant ID to add flavor access for.

nova flavor-access-list

```
usage: nova flavor-access-list [--flavor <flavor>]
```

Print access information about the given flavor.

Optional arguments:

--flavor <flavor> Filter results by flavor name or ID.

nova flavor-access-remove

```
usage: nova flavor-access-remove <flavor> <tenant_id>
```

Remove flavor access for the given tenant.

Positional arguments:

<flavor> Flavor name or ID to remove access for the given tenant.

<tenant_id> Tenant ID to remove flavor access for.

nova flavor-create

```
usage: nova flavor-create [--ephemeral <ephemeral>] [--swap <swap>]
                          [--rxtx-factor <factor>] [--is-public <is-public>]
                          [--description <description>]
                          <name> <id> <ram> <disk> <vcpus>
```

Create a new flavor.

Positional arguments:

<name> Unique name of the new flavor.

<id> Unique ID of the new flavor. Specifying auto will generated a UUID for the ID.

<ram> Memory size in MiB.

<disk> Disk size in GiB.

<vcpus> Number of vcpus

Optional arguments:

--ephemeral <ephemeral> Ephemeral space size in GiB (default 0).

--swap <swap> Swap space size in MiB (default 0).

--rxtx-factor <factor> RX/TX factor (default 1).

--is-public <is-public> Make flavor accessible to the public (default true).

--description <description> A free form description of the flavor. Limited to 65535 characters in length. Only printable characters are allowed. (Supported by API versions 2.55 - 2.latest)

nova flavor-delete

```
usage: nova flavor-delete <flavor>
```

Delete a specific flavor

Positional arguments:

<flavor> Name or ID of the flavor to delete.

nova flavor-key

```
usage: nova flavor-key <flavor> <action> <key=value> [<key=value> ...]
```

Set or unset extra_spec for a flavor.

Positional arguments:

`<flavor>` Name or ID of flavor.

`<action>` Actions: set or unset.

`<key=value>` Extra_specs to set/unset (only key is necessary on unset).

nova flavor-list

```
usage: nova flavor-list [--extra-specs] [--all] [--marker <marker>]
                        [--min-disk <min-disk>] [--min-ram <min-ram>]
                        [--limit <limit>] [--sort-key <sort-key>]
                        [--sort-dir <sort-dir>]
```

Print a list of available flavors (sizes of servers).

Optional arguments:

`--extra-specs` Get extra-specs of each flavor.

`--all` Display all flavors (Admin only).

`--marker <marker>` The last flavor ID of the previous page; displays list of flavors after marker.

`--min-disk <min-disk>` Filters the flavors by a minimum disk space, in GiB.

`--min-ram <min-ram>` Filters the flavors by a minimum RAM, in MiB.

`--limit <limit>` Maximum number of flavors to display. If limit is bigger than CONF.api.max_limit option of Nova API, limit CONF.api.max_limit will be used instead.

`--sort-key <sort-key>` Flavors list sort key.

`--sort-dir <sort-dir>` Flavors list sort direction.

nova flavor-show

```
usage: nova flavor-show <flavor>
```

Show details about the given flavor.

Positional arguments:

`<flavor>` Name or ID of flavor.

nova flavor-update

```
usage: nova flavor-update <flavor> <description>
```

Update the description of an existing flavor. (Supported by API versions 2.55 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

New in version 10.0.0.

Positional arguments

`<flavor>` Name or ID of the flavor to update.

`<description>` A free form description of the flavor. Limited to 65535 characters in length. Only printable characters are allowed.

nova force-delete

```
usage: nova force-delete <server>
```

Force delete a server.

Positional arguments:

`<server>` Name or ID of server.

nova get-mks-console

```
usage: nova get-mks-console <server>
```

Get an MKS console to a server. (Supported by API versions 2.8 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

New in version 3.0.0.

Positional arguments:

`<server>` Name or ID of server.

nova get-password

```
usage: nova get-password <server> [<private-key>]
```

Get the admin password for a server. This operation calls the metadata service to query metadata information and does not read password information from the server itself.

Positional arguments:

`<server>` Name or ID of server.

`<private-key>` Private key (used locally to decrypt password) (Optional). When specified, the command displays the clear (decrypted) VM password. When not specified, the ciphered VM password is displayed.

nova get-rdp-console

```
usage: nova get-rdp-console <server> <console-type>
```

Get a rdp console to a server.

Positional arguments:

`<server>` Name or ID of server.

`<console-type>` Type of rdp console (rdp-html5).

nova get-serial-console

```
usage: nova get-serial-console [--console-type CONSOLE_TYPE] <server>
```

Get a serial console to a server.

Positional arguments:

`<server>` Name or ID of server.

Optional arguments:

`--console-type CONSOLE_TYPE` Type of serial console, default=serial.

nova get-spice-console

```
usage: nova get-spice-console <server> <console-type>
```

Get a spice console to a server.

Positional arguments:

`<server>` Name or ID of server.

`<console-type>` Type of spice console (spice-html5).

nova get-vnc-console

```
usage: nova get-vnc-console <server> <console-type>
```

Get a vnc console to a server.

Positional arguments:

`<server>` Name or ID of server.

`<console-type>` Type of vnc console (novnc or xvpvnc).

nova host-evacuate

```
usage: nova host-evacuate [--target_host <target_host>] [--force] [--strict]
                          <host>
```

Evacuate all instances from failed host.

Positional arguments:

`<host>` The hypervisor hostname (or pattern) to search for.

> **Warning:** Use a fully qualified domain name if you only want to evacuate from a specific host.

Optional arguments:

`--target_host <target_host>` Name of target host. If no host is specified the scheduler will select a target.

`--force` Force an evacuation by not verifying the provided destination host by the scheduler. (Supported by API versions 2.29 - 2.67)

> **Warning:** This could result in failures to actually evacuate the server to the specified host. It is recommended to either not specify a host so that the scheduler will pick one, or specify a host without `--force`.

`--strict` Evacuate host with exact hypervisor hostname match

nova host-evacuate-live

```
usage: nova host-evacuate-live [--target-host <target_host>] [--block-migrate]
                               [--max-servers <max_servers>] [--force]
                               [--strict]
                               <host>
```

Live migrate all instances off the specified host to other available hosts.

Positional arguments:

`<host>` Name of host. The hypervisor hostname (or pattern) to search for.

> **Warning:** Use a fully qualified domain name if you only want to live migrate from a specific host.

Optional arguments:

`--target-host <target_host>` Name of target host. If no host is specified, the scheduler will choose one.

`--block-migrate` Enable block migration. (Default=auto) (Supported by API versions 2.25 - 2.latest)

`--max-servers <max_servers>` Maximum number of servers to live migrate simultaneously

`--force` Force a live-migration by not verifying the provided destination host by the scheduler. (Supported by API versions 2.30 - 2.67)

> **Warning:** This could result in failures to actually live migrate the servers to the specified host. It is recommended to either not specify a host so that the scheduler will pick one, or specify a host without `--force`.

`--strict` live Evacuate host with exact hypervisor hostname match

nova host-meta

```
usage: nova host-meta [--strict] <host> <action> <key=value> [<key=value> ...]
```

Set or Delete metadata on all instances of a host.

Positional arguments:

`<host>` The hypervisor hostname (or pattern) to search for.

> **Warning:** Use a fully qualified domain name if you only want to update metadata for servers on a specific host.

`<action>` Actions: set or delete

`<key=value>` Metadata to set or delete (only key is necessary on delete)

Optional arguments:

`--strict` Set host-meta to the hypervisor with exact hostname match

nova host-servers-migrate

```
usage: nova host-servers-migrate [--strict] <host>
```

Cold migrate all instances off the specified host to other available hosts.

Positional arguments:

`<host>` Name of host. The hypervisor hostname (or pattern) to search for.

> **Warning:** Use a fully qualified domain name if you only want to cold migrate from a specific host.

Optional arguments:

`--strict` Migrate host with exact hypervisor hostname match

nova hypervisor-list

```
usage: nova hypervisor-list [--matching <hostname>] [--marker <marker>]
                            [--limit <limit>]
```

List hypervisors. (Supported by API versions 2.0 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

Optional arguments:

`--matching <hostname>` List hypervisors matching the given <hostname>. If matching is used limit and marker options will be ignored.

`--marker <marker>` The last hypervisor of the previous page; displays list of hypervisors after marker. (Supported by API versions 2.33 - 2.latest)

`--limit <limit>` Maximum number of hypervisors to display. If limit is bigger than CONF.api.max_limit option of Nova API, limit CONF.api.max_limit will be used instead. (Supported by API versions 2.33 - 2.latest)

nova hypervisor-servers

```
usage: nova hypervisor-servers <hostname>
```

List servers belonging to specific hypervisors.

Positional arguments:

`<hostname>` The hypervisor hostname (or pattern) to search for.

nova hypervisor-show

```
usage: nova hypervisor-show [--wrap <integer>] <hypervisor>
```

Display the details of the specified hypervisor.

Positional arguments:

`<hypervisor>` Name or ID of the hypervisor. Starting with microversion 2.53 the ID must be a UUID.

Optional arguments:

`--wrap <integer>` Wrap the output to a specified length. Default is 40 or 0 to disable

nova hypervisor-stats

```
usage: nova hypervisor-stats
```

Get hypervisor statistics over all compute nodes.

nova hypervisor-uptime

```
usage: nova hypervisor-uptime <hypervisor>
```

Display the uptime of the specified hypervisor.

Positional arguments:

`<hypervisor>` Name or ID of the hypervisor. Starting with microversion 2.53 the ID must be a UUID.

nova image-create

```
usage: nova image-create [--metadata <key=value>] [--show] [--poll]
                         <server> <name>
```

Create a new image by taking a snapshot of a running server.

Positional arguments:

`<server>` Name or ID of server.

`<name>` Name of snapshot.

Optional arguments:

`--metadata <key=value>` Record arbitrary key/value metadata to /meta_data.json on the metadata server. Can be specified multiple times.

`--show` Print image info.

`--poll` Report the snapshot progress and poll until image creation is complete.

nova instance-action

```
usage: nova instance-action <server> <request_id>
```

Show an action.

Positional arguments:

`<server>` Name or UUID of the server to show actions for. Only UUID can be used to show actions for a deleted server. (Supported by API versions 2.21 - 2.latest)

`<request_id>` Request ID of the action to get.

nova instance-action-list

```
usage: nova instance-action-list [--marker <marker>] [--limit <limit>]
                                 [--changes-since <changes_since>]
                                 [--changes-before <changes_before>]
                                 <server>
```

List actions on a server.

Positional arguments:

<server> Name or UUID of the server to list actions for. Only UUID can be used to list actions on a deleted server. (Supported by API versions 2.21 - 2.latest)

Optional arguments:

--marker <marker> The last instance action of the previous page; displays list of actions after marker. (Supported by API versions 2.58 - 2.latest)

--limit <limit> Maximum number of instance actions to display. Note that there is a configurable max limit on the server, and the limit that is used will be the minimum of what is requested here and what is configured in the server. (Supported by API versions 2.58 - 2.latest)

--changes-since <changes_since> List only instance actions changed later or equal to a certain point of time. The provided time should be an ISO 8061 formatted time. e.g. 2016-03-04T06:27:59Z. (Supported by API versions 2.58 - 2.latest)

--changes-before <changes_before> List only instance actions changed earlier or equal to a certain point of time. The provided time should be an ISO 8061 formatted time. e.g. 2016-03-04T06:27:59Z. (Supported by API versions 2.66 - 2.latest)

nova instance-usage-audit-log

```
usage: nova instance-usage-audit-log [--before <before>]
```

List/Get server usage audits.

Optional arguments:

--before <before> Filters the response by the date and time before which to list usage audits. The date and time stamp format is as follows: CCYY-MM-DD hh:mm:ss.NNNNNN ex 2015-08-27 09:49:58 or 2015-08-27 09:49:58.123456.

nova interface-attach

```
usage: nova interface-attach [--port-id <port_id>] [--net-id <net_id>]
                             [--fixed-ip <fixed_ip>] [--tag <tag>]
                             <server>
```

Attach a network interface to a server.

Positional arguments:

<server> Name or ID of server.

Optional arguments:

--port-id <port_id> Port ID.

--net-id <net_id> Network ID

--fixed-ip <fixed_ip> Requested fixed IP.

--tag <tag> Tag for the attached interface. (Supported by API versions 2.49 - 2.latest)

nova interface-detach

```
usage: nova interface-detach <server> <port_id>
```

Detach a network interface from a server.

Positional arguments:

<server> Name or ID of server.

<port_id> Port ID.

nova interface-list

```
usage: nova interface-list <server>
```

List interfaces attached to a server.

Positional arguments:

<server> Name or ID of server.

nova keypair-add

```
usage: nova keypair-add [--pub-key <pub-key>] [--key-type <key-type>]
                        [--user <user-id>]
                        <name>
```

Create a new key pair for use with servers.

Positional arguments:

<name> Name of key.

Optional arguments:

--pub-key <pub-key> Path to a public ssh key.

--key-type <key-type> Keypair type. Can be ssh or x509. (Supported by API versions 2.2 - 2.latest)

--user <user-id> ID of user to whom to add key-pair (Admin only). (Supported by API versions 2.10 - 2.latest)

nova keypair-delete

```
usage: nova keypair-delete [--user <user-id>] <name>
```

Delete keypair given by its name. (Supported by API versions 2.0 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

Positional arguments:

<name> Keypair name to delete.

Optional arguments:

--user <user-id> ID of key-pair owner (Admin only).

nova keypair-list

```
usage: nova keypair-list [--user <user-id>] [--marker <marker>]
                         [--limit <limit>]
```

Print a list of keypairs for a user (Supported by API versions 2.0 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

Optional arguments:

--user <user-id> List key-pairs of specified user ID (Admin only).

--marker <marker> The last keypair of the previous page; displays list of keypairs after marker.

--limit <limit> Maximum number of keypairs to display. If limit is bigger than CONF.api.max_limit option of Nova API, limit CONF.api.max_limit will be used instead.

nova keypair-show

```
usage: nova keypair-show [--user <user-id>] <keypair>
```

Show details about the given keypair. (Supported by API versions 2.0 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

Positional arguments:

<keypair> Name of keypair.

Optional arguments:

--user <user-id> ID of key-pair owner (Admin only).

nova limits

```
usage: nova limits [--tenant [<tenant>]] [--reserved]
```

Print rate and absolute limits.

Optional arguments:

--tenant [<tenant>] Display information from single tenant (Admin only).

--reserved Include reservations count.

nova list

```
usage: nova list [--reservation-id <reservation-id>] [--ip <ip-regexp>]
                 [--ip6 <ip6-regexp>] [--name <name-regexp>]
                 [--status <status>] [--flavor <flavor>] [--image <image>]
                 [--host <hostname>] [--all-tenants [<0|1>]]
                 [--tenant [<tenant>]] [--user [<user>]] [--deleted]
                 [--fields <fields>] [--minimal]
                 [--sort <key>[:<direction>]] [--marker <marker>]
                 [--limit <limit>] [--availability-zone <availability_zone>]
                 [--key-name <key_name>] [--[no-]config-drive]
                 [--progress <progress>] [--vm-state <vm_state>]
                 [--task-state <task_state>] [--power-state <power_state>]
                 [--changes-since <changes_since>]
                 [--changes-before <changes_before>]
                 [--tags <tags>] [--tags-any <tags-any>]
                 [--not-tags <not-tags>] [--not-tags-any <not-tags-any>]
                 [--locked]
```

List servers.

Note that from microversion 2.69, during partial infrastructure failures in the deployment, the output of this command may return partial results for the servers present in the failure domain.

Optional arguments:

`--reservation-id <reservation-id>` Only return servers that match reservation-id.

`--ip <ip-regexp>` Search with regular expression match by IP address.

`--ip6 <ip6-regexp>` Search with regular expression match by IPv6 address.

`--name <name-regexp>` Search with regular expression match by name.

`--status <status>` Search by server status.

`--flavor <flavor>` Search by flavor name or ID.

`--image <image>` Search by image name or ID.

`--host <hostname>` Search servers by hostname to which they are assigned (Admin only).

`--all-tenants [<0|1>]` Display information from all tenants (Admin only).

`--tenant [<tenant>]` Display information from single tenant (Admin only).

`--user [<user>]` Display information from single user (Admin only until microversion 2.82).

`--deleted` Only display deleted servers (Admin only).

`--fields <fields>` Comma-separated list of fields to display. Use the show command to see which fields are available.

`--minimal` Get only UUID and name.

`--sort <key>[:<direction>]` Comma-separated list of sort keys and directions in the form of <key>[:<asc|desc>]. The direction defaults to descending if not specified.

`--marker <marker>` The last server UUID of the previous page; displays list of servers after marker.

--limit <limit> Maximum number of servers to display. If limit == -1, all servers will be displayed. If limit is bigger than CONF.api.max_limit option of Nova API, limit CONF.api.max_limit will be used instead.

--availability-zone <availability_zone> Display servers based on their availability zone (Admin only until microversion 2.82).

--key-name <key_name> Display servers based on their keypair name (Admin only until microversion 2.82).

--config-drive Display servers that have a config drive attached. It is mutually exclusive with no-config-drive. (Admin only until microversion 2.82).

--no-config-drive Display servers that do not have a config drive attached. It is mutually exclusive with config-drive. (Admin only until microversion 2.82).

--progress <progress> Display servers based on their progress value (Admin only until microversion 2.82).

--vm-state <vm_state> Display servers based on their vm_state value (Admin only until microversion 2.82).

--task-state <task_state> Display servers based on their task_state value (Admin only until microversion 2.82).

--power-state <power_state> Display servers based on their power_state value (Admin only until microversion 2.82).

--changes-since <changes_since> List only servers changed later or equal to a certain point of time. The provided time should be an ISO 8061 formatted time. e.g. 2016-03-04T06:27:59Z .

--changes-before <changes_before> List only servers changed earlier or equal to a certain point of time. The provided time should be an ISO 8061 formatted time. e.g. 2016-03-05T06:27:59Z . (Supported by API versions 2.66 - 2.latest)

--tags <tags> The given tags must all be present for a server to be included in the list result. Boolean expression in this case is t1 AND t2. Tags must be separated by commas: tags <tag1,tag2> (Supported by API versions 2.26 - 2.latest)

--tags-any <tags-any> If one of the given tags is present the server will be included in the list result. Boolean expression in this case is t1 OR t2. Tags must be separated by commas: tags-any <tag1,tag2> (Supported by API versions 2.26 - 2.latest)

--not-tags <not-tags> Only the servers that do not have any of the given tags will be included in the list results. Boolean expression in this case is NOT(t1 AND t2). Tags must be separated by commas: not-tags <tag1,tag2> (Supported by API versions 2.26 - 2.latest)

--not-tags-any <not-tags-any> Only the servers that do not have at least one of the given tags will be included in the list result. Boolean expression in this case is NOT(t1 OR t2). Tags must be separated by commas: not-tags-any <tag1,tag2> (Supported by API versions 2.26 - 2.latest)

--locked <locked> Display servers based on their locked value. A value must be specified; eg. true will list only locked servers and false will list only unlocked servers. (Supported by API versions 2.73 - 2.latest)

nova list-secgroup

```
usage: nova list-secgroup <server>
```

List Security Group(s) of a server.

Positional arguments:

`<server>` Name or ID of server.

nova live-migration

```
usage: nova live-migration [--block-migrate] [--force] <server> [<host>]
```

Migrate running server to a new machine.

Positional arguments:

`<server>` Name or ID of server.

`<host>` Destination host name. If no host is specified, the scheduler will choose one.

Optional arguments:

`--block-migrate` True in case of block_migration. (Default=auto:live_migration) (Supported by API versions 2.25 - 2.latest)

`--force` Force a live-migration by not verifying the provided destination host by the scheduler. (Supported by API versions 2.30 - 2.67)

> **Warning:** This could result in failures to actually live migrate the server to the specified host. It is recommended to either not specify a host so that the scheduler will pick one, or specify a host without `--force`.

nova live-migration-abort

```
usage: nova live-migration-abort <server> <migration>
```

Abort an on-going live migration. (Supported by API versions 2.24 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

For microversions from 2.24 to 2.64 the migration status must be `running`; for microversion 2.65 and greater, the migration status can also be `queued` and `preparing`.

New in version 3.3.0.

Positional arguments:

`<server>` Name or ID of server.

`<migration>` ID of migration.

nova live-migration-force-complete

```
usage: nova live-migration-force-complete <server> <migration>
```

Force on-going live migration to complete. (Supported by API versions 2.22 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

New in version 3.3.0.

Positional arguments:

`<server>` Name or ID of server.

`<migration>` ID of migration.

nova lock

```
usage: nova lock [--reason <reason>] <server>
```

Lock a server. A normal (non-admin) user will not be able to execute actions on a locked server.

Positional arguments:

`<server>` Name or ID of server.

Optional arguments:

`--reason <reason>` Reason for locking the server. (Supported by API versions 2.73 - 2.latest)

nova meta

```
usage: nova meta <server> <action> <key=value> [<key=value> ...]
```

Set or delete metadata on a server.

Positional arguments:

`<server>` Name or ID of server.

`<action>` Actions: set or delete.

`<key=value>` Metadata to set or delete (only key is necessary on delete).

nova migrate

```
usage: nova migrate [--host <host>] [--poll] <server>
```

Migrate a server. The new host will be selected by the scheduler.

Positional arguments:

`<server>` Name or ID of server.

Optional arguments:

`--host <host>` Destination host name. (Supported by API versions 2.56 - 2.latest)

--poll Report the server migration progress until it completes.

nova migration-list

```
usage: nova migration-list [--instance-uuid <instance_uuid>]
                           [--host <host>]
                           [--status <status>]
                           [--migration-type <migration_type>]
                           [--source-compute <source_compute>]
                           [--marker <marker>]
                           [--limit <limit>]
                           [--changes-since <changes_since>]
                           [--changes-before <changes_before>]
                           [--project-id <project_id>]
                           [--user-id <user_id>]
```

Print a list of migrations.

Examples

To see the list of evacuation operations *from* a compute service host:

```
nova migration-list --migration-type evacuation --source-compute host.foo.bar
```

Optional arguments:

--instance-uuid <instance_uuid> Fetch migrations for the given instance.

--host <host> Fetch migrations for the given source or destination host.

--status <status> Fetch migrations for the given status.

--migration-type <migration_type> Filter migrations by type. Valid values are:

- evacuation
- live-migration
- migration

  Note: This is a cold migration.

- resize

--source-compute <source_compute> Filter migrations by source compute host name.

--marker <marker> The last migration of the previous page; displays list of migrations after marker. Note that the marker is the migration UUID. (Supported by API versions 2.59 - 2.latest)

--limit <limit> Maximum number of migrations to display. Note that there is a configurable max limit on the server, and the limit that is used will be the minimum of what is requested here and what is configured in the server. (Supported by API versions 2.59 - 2.latest)

--changes-since <changes_since> List only migrations changed later or equal to a certain point of time. The provided time should be an ISO 8061 formatted time. e.g. 2016-03-04T06:27:59Z . (Supported by API versions 2.59 - 2.latest)

--changes-before <changes_before> List only migrations changed earlier or equal to a certain point of time. The provided time should be an ISO 8061 formatted time. e.g. 2016-03-04T06:27:59Z . (Supported by API versions 2.66 - 2.latest)

--project-id <project_id> Filter the migrations by the given project ID. (Supported by API versions 2.80 - 2.latest)

--user-id <user_id> Filter the migrations by the given user ID. (Supported by API versions 2.80 - 2.latest)

nova pause

```
usage: nova pause <server>
```

Pause a server.

Positional arguments:

<server> Name or ID of server.

nova quota-class-show

```
usage: nova quota-class-show <class>
```

List the quotas for a quota class.

Positional arguments:

<class> Name of quota class to list the quotas for.

nova quota-class-update

```
usage: nova quota-class-update [--instances <instances>] [--cores <cores>]
                               [--ram <ram>]
                               [--metadata-items <metadata-items>]
                               [--key-pairs <key-pairs>]
                               [--server-groups <server-groups>]
                               [--server-group-members <server-group-members>]
                               <class>
```

Update the quotas for a quota class. (Supported by API versions 2.0 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

Positional arguments:

<class> Name of quota class to set the quotas for.

Optional arguments:

--instances <instances> New value for the instances quota.

--cores <cores> New value for the cores quota.

--ram <ram> New value for the ram quota.

--metadata-items <metadata-items> New value for the metadata-items quota.

--key-pairs <key-pairs> New value for the key-pairs quota.

--server-groups <server-groups> New value for the server-groups quota.

--server-group-members <server-group-members> New value for the server-group-members quota.

nova quota-defaults

```
usage: nova quota-defaults [--tenant <tenant-id>]
```

List the default quotas for a tenant.

Optional arguments:

--tenant <tenant-id> ID of tenant to list the default quotas for.

nova quota-delete

```
usage: nova quota-delete --tenant <tenant-id> [--user <user-id>]
```

Delete quota for a tenant/user so their quota will Revert back to default.

Optional arguments:

--tenant <tenant-id> ID of tenant to delete quota for.

--user <user-id> ID of user to delete quota for.

nova quota-show

```
usage: nova quota-show [--tenant <tenant-id>] [--user <user-id>] [--detail]
```

List the quotas for a tenant/user.

Optional arguments:

--tenant <tenant-id> ID of tenant to list the quotas for.

--user <user-id> ID of user to list the quotas for.

--detail Show detailed info (limit, reserved, in-use).

nova quota-update

```
usage: nova quota-update [--user <user-id>] [--instances <instances>]
                         [--cores <cores>] [--ram <ram>]
                         [--metadata-items <metadata-items>]
                         [--key-pairs <key-pairs>]
                         [--server-groups <server-groups>]
                         [--server-group-members <server-group-members>]
                         [--force]
                         <tenant-id>
```

Update the quotas for a tenant/user. (Supported by API versions 2.0 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

Positional arguments:

`<tenant-id>` ID of tenant to set the quotas for.

Optional arguments:

`--user <user-id>` ID of user to set the quotas for.

`--instances <instances>` New value for the instances quota.

`--cores <cores>` New value for the cores quota.

`--ram <ram>` New value for the ram quota.

`--metadata-items <metadata-items>` New value for the metadata-items quota.

`--key-pairs <key-pairs>` New value for the key-pairs quota.

`--server-groups <server-groups>` New value for the server-groups quota.

`--server-group-members <server-group-members>` New value for the server-group-members quota.

`--force` Whether force update the quota even if the already used and reserved exceeds the new quota.

nova reboot

```
usage: nova reboot [--hard] [--poll] <server> [<server> ...]
```

Reboot a server.

Positional arguments:

`<server>` Name or ID of server(s).

Optional arguments:

`--hard` Perform a hard reboot (instead of a soft one). Note: Ironic does not currently support soft reboot; consequently, bare metal nodes will always do a hard reboot, regardless of the use of this option.

`--poll` Poll until reboot is complete.

nova rebuild

```
usage: nova rebuild [--rebuild-password <rebuild-password>] [--poll]
                    [--minimal] [--preserve-ephemeral] [--name <name>]
                    [--description <description>] [--meta <key=value>]
                    [--key-name <key-name>] [--key-unset]
                    [--user-data <user-data>] [--user-data-unset]
                    [--trusted-image-certificate-id <trusted-image-
↪certificate-id>]
                    [--trusted-image-certificates-unset]
                    [--hostname <hostname>]
                    <server> <image>
```

Shutdown, re-image, and re-boot a server.

Positional arguments:

`<server>` Name or ID of server.

`<image>` Name or ID of new image.

Optional arguments:

`--rebuild-password <rebuild-password>` Set the provided admin password on the rebuilt server.

`--poll` Report the server rebuild progress until it completes.

`--minimal` Skips flavor/image lookups when showing servers.

`--preserve-ephemeral` Preserve the default ephemeral storage partition on rebuild.

`--name <name>` Name for the new server.

`--description <description>` New description for the server. (Supported by API versions 2.19 - 2.latest)

`--meta <key=value>` Record arbitrary key/value metadata to /meta_data.json on the metadata server. Can be specified multiple times.

`--key-name <key-name>` Keypair name to set in the server. Cannot be specified with the key-unset option. (Supported by API versions 2.54 - 2.latest)

`--key-unset` Unset keypair in the server. Cannot be specified with the key-name option. (Supported by API versions 2.54 - 2.latest)

`--user-data <user-data>` User data file to pass to be exposed by the metadata server. (Supported by API versions 2.57 - 2.latest)

`--user-data-unset` Unset user_data in the server. Cannot be specified with the user-data option. (Supported by API versions 2.57 - 2.latest)

`--trusted-image-certificate-id <trusted-image-certificate-id>` Trusted image certificate IDs used to validate certificates during the image signature verification process. Defaults to env[OS_TRUSTED_IMAGE_CERTIFICATE_IDS]. May be specified multiple times to pass multiple trusted image certificate IDs. (Supported by API versions 2.63 - 2.latest)

`--trusted-image-certificates-unset` Unset trusted_image_certificates in the server. Cannot be specified with the `--trusted-image-certificate-id` option. (Supported by API versions 2.63 - 2.latest)

--hostname <hostname> New hostname for the instance. This only updates the hostname stored in the metadata server: a utility running on the guest is required to propagate these changes to the guest. (Supported by API versions 2.90 - 2.latest)

nova refresh-network

```
usage: nova refresh-network <server>
```

Refresh server network information.

Positional arguments:

<server> Name or ID of a server for which the network cache should be refreshed from neutron (Admin only).

nova remove-secgroup

```
usage: nova remove-secgroup <server> <secgroup>
```

Remove a Security Group from a server.

Positional arguments:

<server> Name or ID of server.

<secgroup> Name of Security Group.

nova rescue

```
usage: nova rescue [--password <password>] [--image <image>] <server>
```

Reboots a server into rescue mode, which starts the machine from either the initial image or a specified image, attaching the current boot disk as secondary.

Positional arguments:

<server> Name or ID of server.

Optional arguments:

--password <password> The admin password to be set in the rescue environment.

--image <image> The image to rescue with.

nova reset-network

```
usage: nova reset-network <server>
```

Reset network of a server.

Positional arguments:

`<server>` Name or ID of server.

nova reset-state

```
usage: nova reset-state [--all-tenants] [--active] <server> [<server> ...]
```

Reset the state of a server.

Positional arguments:

`<server>` Name or ID of server(s).

Optional arguments:

`--all-tenants` Reset state server(s) in another tenant by name (Admin only).

`--active` Request the server be reset to active state instead of error state (the default).

nova resize

```
usage: nova resize [--poll] <server> <flavor>
```

Resize a server.

Positional arguments:

`<server>` Name or ID of server.

`<flavor>` Name or ID of new flavor.

Optional arguments:

`--poll` Report the server resize progress until it completes.

nova resize-confirm

```
usage: nova resize-confirm <server>
```

Confirm a previous resize.

Positional arguments:

`<server>` Name or ID of server.

nova resize-revert

```
usage: nova resize-revert <server>
```

Revert a previous resize (and return to the previous VM).

Positional arguments:

`<server>` Name or ID of server.

nova restore

```
usage: nova restore <server>
```

Restore a soft-deleted server.

Positional arguments:

`<server>` Name or ID of server.

nova resume

```
usage: nova resume <server>
```

Resume a server.

Positional arguments:

`<server>` Name or ID of server.

nova server-group-create

```
usage: nova server-group-create [--rules <key=value>] <name> <policy>
```

Create a new server group with the specified details.

Positional arguments:

`<name>` Server group name.

`<policy>` Policy for the server groups.

Optional arguments:

`--rule` Policy rules for the server groups. (Supported by API versions 2.64 - 2.latest. Currently, only the `max_server_per_host` rule is supported for the `anti-affinity` policy. The `max_server_per_host` rule allows specifying how many members of the anti-affinity group can reside on the same compute host. If not specified, only one member from the same anti-affinity group can reside on a given host.

nova server-group-delete

```
usage: nova server-group-delete <id> [<id> ...]
```

Delete specific server group(s).

Positional arguments:

<id> Unique ID(s) of the server group to delete.

nova server-group-get

```
usage: nova server-group-get <id>
```

Get a specific server group.

Positional arguments:

<id> Unique ID of the server group to get.

nova server-group-list

```
usage: nova server-group-list [--limit <limit>] [--offset <offset>]
                              [--all-projects]
```

Print a list of all server groups.

Optional arguments:

--limit <limit> Maximum number of server groups to display. If limit is bigger than CONF.api.max_limit option of Nova API, limit CONF.api.max_limit will be used instead.

--offset <offset> The offset of groups list to display; use with limit to return a slice of server groups.

--all-projects Display server groups from all projects (Admin only).

nova server-migration-list

```
usage: nova server-migration-list <server>
```

Get the migrations list of specified server. (Supported by API versions 2.23 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

New in version 3.3.0.

Positional arguments:

<server> Name or ID of server.

nova server-migration-show

```
usage: nova server-migration-show <server> <migration>
```

Get the migration of specified server. (Supported by API versions 2.23 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

New in version 3.3.0.

Positional arguments:

<server> Name or ID of server.

<migration> ID of migration.

nova server-tag-add

```
usage: nova server-tag-add <server> <tag> [<tag> ...]
```

Add one or more tags to a server. (Supported by API versions 2.26 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

New in version 4.1.0.

Positional arguments:

<server> Name or ID of server.

<tag> Tag(s) to add.

nova server-tag-delete

```
usage: nova server-tag-delete <server> <tag> [<tag> ...]
```

Delete one or more tags from a server. (Supported by API versions 2.26 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

New in version 4.1.0.

Positional arguments:

<server> Name or ID of server.

<tag> Tag(s) to delete.

nova server-tag-delete-all

```
usage: nova server-tag-delete-all <server>
```

Delete all tags from a server. (Supported by API versions 2.26 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

New in version 4.1.0.

Positional arguments:

<server> Name or ID of server.

nova server-tag-list

```
usage: nova server-tag-list <server>
```

Get list of tags from a server. (Supported by API versions 2.26 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

New in version 4.1.0.

Positional arguments:

<server> Name or ID of server.

nova server-tag-set

```
usage: nova server-tag-set <server> <tags> [<tags> ...]
```

Set list of tags to a server. (Supported by API versions 2.26 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

New in version 4.1.0.

Positional arguments:

<server> Name or ID of server.

<tags> Tag(s) to set.

nova server-topology

```
usage: nova server-topology <server>
```

Retrieve server NUMA topology information. Host specific fields are only visible to users with the administrative role. (Supported by API versions 2.78 - 2.latest)

New in version 15.1.0.

Positional arguments:

<server> Name or ID of server.

nova service-delete

```
usage: nova service-delete <id>
```

Delete the service.

Important: If deleting a nova-compute service, be sure to stop the actual `nova-compute` process on the physical host *before* deleting the service with this command. Failing to do so can lead to the running service re-creating orphaned **compute_nodes** table records in the database.

Positional arguments:

<id> ID of service as a UUID. (Supported by API versions 2.53 - 2.latest)

nova service-disable

```
usage: nova service-disable [--reason <reason>] <id>
```

Disable the service.

Positional arguments:

<id> ID of the service as a UUID. (Supported by API versions 2.53 - 2.latest)

Optional arguments:

--reason <reason> Reason for disabling the service.

nova service-enable

```
usage: nova service-enable <id>
```

Enable the service.

Positional arguments:

<id> ID of the service as a UUID. (Supported by API versions 2.53 - 2.latest)

nova service-force-down

```
usage: nova service-force-down [--unset] <id>
```

Force service to down. (Supported by API versions 2.11 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

New in version 2.27.0.

Positional arguments:

<id> ID of the service as a UUID. (Supported by API versions 2.53 - 2.latest)

Optional arguments:

--unset Unset the forced_down state of the service.

nova service-list

```
usage: nova service-list [--host <hostname>] [--binary <binary>]
```

Show a list of all running services. Filter by host & binary.

Note that from microversion 2.69, during partial infrastructure failures in the deployment, the output of this command may return partial results for the services present in the failure domain.

Optional arguments:

`--host <hostname>` Name of host.

`--binary <binary>` Service binary.

nova set-password

```
usage: nova set-password <server>
```

Change the admin password for a server.

Positional arguments:

`<server>` Name or ID of server.

nova shelve

```
usage: nova shelve <server>
```

Shelve a server.

Positional arguments:

`<server>` Name or ID of server.

nova shelve-offload

```
usage: nova shelve-offload <server>
```

Remove a shelved server from the compute node.

Positional arguments:

`<server>` Name or ID of server.

nova show

```
usage: nova show [--minimal] [--wrap <integer>] <server>
```

Show details about the given server.

Note that from microversion 2.69, during partial infrastructure failures in the deployment, the output of this command may return partial results for the server if it exists in the failure domain.

Positional arguments:

`<server>` Name or ID of server.

Optional arguments:

`--minimal` Skips flavor/image lookups when showing servers.

`--wrap <integer>` Wrap the output to a specified length, or 0 to disable.

nova ssh

```
usage: nova ssh [--port PORT] [--address-type ADDRESS_TYPE]
                [--network <network>] [--ipv6] [--login <login>] [-i IDENTITY]
                [--extra-opts EXTRA]
                <server>
```

SSH into a server.

Positional arguments:

`<server>` Name or ID of server.

Optional arguments:

`--port PORT` Optional flag to indicate which port to use for ssh. (Default=22)

`--address-type ADDRESS_TYPE` Optional flag to indicate which IP type to use. Possible values includes fixed and floating (the Default).

`--network <network>` Network to use for the ssh.

`--ipv6` Optional flag to indicate whether to use an IPv6 address attached to a server. (Defaults to IPv4 address)

`--login <login>` Login to use.

`-i IDENTITY, --identity IDENTITY` Private key file, same as the -i option to the ssh command.

`--extra-opts EXTRA` Extra options to pass to ssh. see: man ssh.

nova start

```
usage: nova start [--all-tenants] <server> [<server> ...]
```

Start the server(s).

Positional arguments:

`<server>` Name or ID of server(s).

Optional arguments:

`--all-tenants` Start server(s) in another tenant by name (Admin only).

nova stop

```
usage: nova stop [--all-tenants] <server> [<server> ...]
```

Stop the server(s).

Positional arguments:

`<server>` Name or ID of server(s).

Optional arguments:

`--all-tenants` Stop server(s) in another tenant by name (Admin only).

nova suspend

```
usage: nova suspend <server>
```

Suspend a server.

Positional arguments:

`<server>` Name or ID of server.

nova trigger-crash-dump

```
usage: nova trigger-crash-dump <server>
```

Trigger crash dump in an instance. (Supported by API versions 2.17 - 2.latest) [hint: use os-compute-api-version flag to show help message for proper version]

New in version 3.3.0.

Positional arguments:

`<server>` Name or ID of server.

nova unlock

```
usage: nova unlock <server>
```

Unlock a server.

Positional arguments:

<server> Name or ID of server.

nova unpause

```
usage: nova unpause <server>
```

Unpause a server.

Positional arguments:

<server> Name or ID of server.

nova unrescue

```
usage: nova unrescue <server>
```

Restart the server from normal boot disk again.

Positional arguments:

<server> Name or ID of server.

nova unshelve

```
usage: nova unshelve [--availability-zone <availability_zone>] <server>
```

Unshelve a server.

Positional arguments:

<server> Name or ID of server.

Optional arguments:

--availability-zone <availability_zone> Name of the availability zone in which to unshelve a `SHELVED_OFFLOADED` server. (Supported by API versions 2.77 - 2.latest)

nova update

```
usage: nova update [--name <name>] [--description <description>]
                   [--hostname <hostname>]
                   <server>
```

Update attributes of a server.

Positional arguments:

<server> Name (old name) or ID of server.

Optional arguments:

--name <name> New name for the server.

--description <description> New description for the server. If it equals to empty string (i.g.), the server description will be removed. (Supported by API versions 2.19 - 2.latest)

--hostname <hostname> New hostname for the instance. This only updates the hostname stored in the metadata server: a utility running on the guest is required to propagate these changes to the guest. (Supported by API versions 2.90 - 2.latest)

nova usage

```
usage: nova usage [--start <start>] [--end <end>] [--tenant <tenant-id>]
```

Show usage data for a single tenant.

Optional arguments:

--start <start> Usage range start date ex 2012-01-20. (default: 4 weeks ago)

--end <end> Usage range end date, ex 2012-01-20. (default: tomorrow)

--tenant <tenant-id> UUID of tenant to get usage for.

nova usage-list

```
usage: nova usage-list [--start <start>] [--end <end>]
```

List usage data for all tenants.

Optional arguments:

--start <start> Usage range start date ex 2012-01-20. (default: 4 weeks ago)

--end <end> Usage range end date, ex 2012-01-20. (default: tomorrow)

nova version-list

```
usage: nova version-list
```

List all API versions.

nova volume-attach

```
usage: nova volume-attach [--delete-on-termination] [--tag <tag>]
                          <server> <volume> [<device>]
```

Attach a volume to a server.

Positional arguments:

<server> Name or ID of server.

<volume> ID of the volume to attach.

<device> Name of the device e.g. /dev/vdb. Use auto for autoassign (if supported). Libvirt driver will use default device name.

Optional arguments:

--tag <tag> Tag for the attached volume. (Supported by API versions 2.49 - 2.latest)

--delete-on-termination Specify if the attached volume should be deleted when the server is destroyed. By default the attached volume is not deleted when the server is destroyed. (Supported by API versions 2.79 - 2.latest)

nova volume-attachments

```
usage: nova volume-attachments <server>
```

List all the volumes attached to a server.

Positional arguments:

<server> Name or ID of server.

nova volume-detach

```
usage: nova volume-detach <server> <volume>
```

Detach a volume from a server.

Positional arguments:

<server> Name or ID of server.

<volume> ID of the volume to detach.

nova volume-update

```
usage: nova volume-update [--[no-]delete-on-termination]
                          <server> <src_volume> <dest_volume>
```

Update the attachment on the server. Migrates the data from an attached volume to the specified available volume and swaps out the active attachment to the new volume.

Positional arguments:

`<server>` Name or ID of server.

`<src_volume>` ID of the source (original) volume.

`<dest_volume>` ID of the destination volume.

Optional arguments:

`--delete-on-termination` Specify that the volume should be deleted when the server is destroyed. It is mutually exclusive with no-delete-on-termination. (Supported by API versions 2.85 - 2.latest)

`--no-delete-on-termination` Specify that the attached volume should not be deleted when the server is destroyed. It is mutually exclusive with delete-on-termination. (Supported by API versions 2.85 - 2.latest)

nova bash-completion

```
usage: nova bash-completion
```

Prints all of the commands and options to stdout so that the nova.bash_completion script doesnt have to hard code them.

CHAPTER

THREE

REFERENCE

3.1 novaclient

3.1.1 novaclient package

Subpackages

novaclient.v2 package

Submodules

novaclient.v2.agents module

agent interface

class novaclient.v2.agents.**Agent**(*manager*, *info*, *loaded=False*, *resp=None*)

Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

class novaclient.v2.agents.**AgentsManager**(*api*)

Bases: *novaclient.base.ManagerWithFind*

create(*os*, *architecture*, *version*, *url*, *md5hash*, *hypervisor*)

Create a new agent build.

delete(*id*)

Deletes an existing agent build.

Parameters **id** The agents id to delete

Returns An instance of novaclient.base.TupleWithMeta

list(*hypervisor=None*)

List all agent builds.

resource_class
alias of *novaclient.v2.agents.Agent*

update(*id*, *version*, *url*, *md5hash*)
Update an existing agent build.

novaclient.v2.aggregates module

Aggregate interface.

class novaclient.v2.aggregates.**Aggregate**(*manager*, *info*, *loaded=False*, *resp=None*)
Bases: *novaclient.base.Resource*

An aggregates is a collection of compute hosts.

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

add_host(*host*)

cache_images(*images*)

delete()
Delete the own aggregate.

Returns An instance of novaclient.base.TupleWithMeta

remove_host(*host*)

set_metadata(*metadata*)

update(*values*)
Update the name and/or availability zone.

class novaclient.v2.aggregates.**AggregateManager**(*api*)
Bases: *novaclient.base.ManagerWithFind*

add_host(*aggregate*, *host*)
Add a host into the Host Aggregate.

cache_images(*aggregate*, *images*)
Request images be cached on a given aggregate.

Parameters

- **aggregate** The aggregate to target
- **images** A list of image IDs to request caching

Returns An instance of novaclient.base.TupleWithMeta

create(*name*, *availability_zone*)
Create a new aggregate.

delete(*aggregate*)
Delete the specified aggregate.

Parameters **aggregate** The aggregate to delete

Returns An instance of novaclient.base.TupleWithMeta

get(*aggregate*)
Get details of the specified aggregate.

get_details(*aggregate*)
Get details of the specified aggregate.

list()
Get a list of os-aggregates.

remove_host(*aggregate*, *host*)
Remove a host from the Host Aggregate.

resource_class
alias of *novaclient.v2.aggregates.Aggregate*

set_metadata(*aggregate*, *metadata*)
Set aggregate metadata, replacing the existing metadata.

update(*aggregate*, *values*)
Update the name and/or availability zone.

novaclient.v2.assisted_volume_snapshots module

Assisted volume snapshots - to be used by Cinder and not end users.

class novaclient.v2.assisted_volume_snapshots.**AssistedSnapshotManager**(*api*)
Bases: *novaclient.base.Manager*

create(*volume_id*, *create_info*)

delete(*snapshot*, *delete_info*)
Delete a specified assisted volume snapshot.

Parameters

- **snapshot** an assisted volume snapshot to delete
- **delete_info** Information for snapshot deletion

Returns An instance of novaclient.base.TupleWithMeta

resource_class
alias of *novaclient.v2.assisted_volume_snapshots.Snapshot*

class novaclient.v2.assisted_volume_snapshots.**Snapshot**(*manager*, *info*, *loaded=False*, *resp=None*)
Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object

- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

delete()
Delete this snapshot.

Returns An instance of novaclient.base.TupleWithMeta

novaclient.v2.availability_zones module

Availability Zone interface

class novaclient.v2.availability_zones.**AvailabilityZone**(*manager*, *info*, *loaded=False*, *resp=None*)

Bases: *novaclient.base.Resource*

An availability zone object.

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

NAME_ATTR = 'display_name'

class novaclient.v2.availability_zones.**AvailabilityZoneManager**(*api*)

Bases: *novaclient.base.ManagerWithFind*

Manage *AvailabilityZone* resources.

list(*detailed=True*)
Get a list of all availability zones.

Parameters detailed If True, list availability zones with details.

Returns list of *AvailabilityZone*

resource_class
alias of *novaclient.v2.availability_zones.AvailabilityZone*

return_parameter_name = 'availabilityZoneInfo'

novaclient.v2.client module

class novaclient.v2.client.**Client**(*api_version=None*, *auth=None*, *auth_token=None*, *auth_url=None*, *cacert=None*, *cert=None*, *direct_use=True*, *endpoint_override=None*, *endpoint_type='publicURL'*, *extensions=None*, *http_log_debug=False*, *insecure=False*, *logger=None*, *os_cache=False*, *password=None*, *project_domain_id=None*, *project_domain_name=None*, *project_id=None*, *project_name=None*, *region_name=None*, *service_name=None*, *service_type='compute'*, *session=None*, *timeout=None*, *timings=False*, *user_domain_id=None*, *user_domain_name=None*, *user_id=None*, *username=None*, ***kwargs*)

Bases: `object`

Top-level object to access the OpenStack Compute API.

> **Warning:** All scripts and projects should not initialize this class directly. It should be done via *novaclient.client.Client* interface.

Initialization of Client object.

Parameters

- **api_version** (`novaclient.api_versions.APIVersion`) Compute API version
- **auth** (*str*) Auth
- **auth_token** (*str*) Auth token
- **auth_url** (*str*) Auth URL
- **cacert** (*str*) ca-certificate
- **cert** (*str*) certificate
- **direct_use** (*bool*) Inner variable of novaclient. Do not use it outside novaclient. Its restricted.
- **endpoint_override** (*str*) Bypass URL
- **endpoint_type** (*str*) Endpoint Type
- **extensions** (*str*) Extensions
- **http_log_debug** (*bool*) Enable debugging for HTTP connections
- **insecure** (*bool*) Allow insecure
- **logger** (*logging.Logger*) Logger instance to be used for all logging stuff
- **password** (*str*) User password
- **os_cache** (*bool*) OS cache
- **project_domain_id** (*str*) ID of project domain

- **project_domain_name** (*str*) Name of project domain
- **project_id** (*str*) Project/Tenant ID
- **project_name** (*str*) Project/Tenant name
- **region_name** (*str*) Region Name
- **service_name** (*str*) Service Name
- **service_type** (*str*) Service Type
- **session** (*str*) Session
- **timeout** (*float*) API timeout, None or 0 disables
- **timings** (*bool*) Timings
- **user_domain_id** (*str*) ID of user domain
- **user_domain_name** (*str*) Name of user domain
- **user_id** (*str*) User ID
- **username** (*str*) Username

property api_version

get_timings()

reset_timings()

novaclient.v2.flavor_access module

Flavor access interface.

class novaclient.v2.flavor_access.**FlavorAccess**(*manager*, *info*, *loaded=False*, *resp=None*)

Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

class novaclient.v2.flavor_access.**FlavorAccessManager**(*api*)

Bases: *novaclient.base.ManagerWithFind*

Manage *FlavorAccess* resources.

add_tenant_access(*flavor*, *tenant*)

Add a tenant to the given flavor access list.

list(***kwargs*)

remove_tenant_access(*flavor*, *tenant*)

Remove a tenant from the given flavor access list.

resource_class
alias of *novaclient.v2.flavor_access.FlavorAccess*

novaclient.v2.flavors module

Flavor interface.

class novaclient.v2.flavors.Flavor(*manager*, *info*, *loaded=False*, *resp=None*)
Bases: *novaclient.base.Resource*

A flavor is an available hardware configuration for a server.

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

HUMAN_ID = True

delete()
Delete this flavor.

Returns An instance of novaclient.base.TupleWithMeta

property ephemeral
Provide a user-friendly accessor to OS-FLV-EXT-DATA:ephemeral.

get_keys()
Get extra specs from a flavor.

Returns An instance of novaclient.base.DictWithMeta

property is_public
Provide a user-friendly accessor to os-flavor-access:is_public.

set_keys(*metadata*)
Set extra specs on a flavor.

Parameters metadata A dict of key/value pairs to be set

unset_keys(*keys*)
Unset extra specs on a flavor.

Parameters keys A list of keys to be unset

Returns An instance of novaclient.base.TupleWithMeta

update(*description=None*)
Update the description for this flavor.

Parameters description The description to set on the flavor.

Returns *Flavor*

class novaclient.v2.flavors.**FlavorManager**(*api*)

Bases: *novaclient.base.ManagerWithFind*

Manage *Flavor* resources.

create(*name*, *ram*, *vcpus*, *disk*, *flavorid='auto'*, *ephemeral=0*, *swap=0*, *rxtx_factor=1.0*, *is_public=True*, *description=None*)

Create a flavor.

Parameters

- **name** Descriptive name of the flavor
- **ram** Memory in MiB for the flavor
- **vcpus** Number of VCPUs for the flavor
- **disk** Size of local disk in GiB
- **flavorid** ID for the flavor (optional). You can use the reserved value `"auto"` to have Nova generate a UUID for the flavor in cases where you cannot simply pass `None`.
- **ephemeral** Ephemeral disk space in GiB.
- **swap** Swap space in MiB
- **rxtx_factor** RX/TX factor
- **is_public** Whether or not the flavor is public.
- **description** A free form description of the flavor. Limited to 65535 characters in length. Only printable characters are allowed. (Available starting with microversion 2.55)

Returns *Flavor*

delete(*flavor*)

Delete a specific flavor.

Parameters **flavor** Instance of *Flavor* to delete or ID of the flavor to delete.

Returns An instance of novaclient.base.TupleWithMeta

get(*flavor*)

Get a specific flavor.

Parameters **flavor** The ID of the *Flavor* to get.

Returns *Flavor*

is_alphanum_id_allowed = True

list(*detailed=True*, *is_public=True*, *marker=None*, *min_disk=None*, *min_ram=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

Get a list of all flavors.

Parameters

- **detailed** Whether flavor needs to be return with details (optional).
- **is_public** Filter flavors with provided access type (optional). None means give all flavors and only admin has query access to all flavor types.

- **marker** Begin returning flavors that appear later in the flavor list than that represented by this flavor id (optional).
- **min_disk** Filters the flavors by a minimum disk space, in GiB.
- **min_ram** Filters the flavors by a minimum RAM, in MiB.
- **limit** maximum number of flavors to return (optional). Note the API server has a configurable default limit. If no limit is specified here or limit is larger than default, the default limit will be used.
- **sort_key** Flavors list sort key (optional).
- **sort_dir** Flavors list sort direction (optional).

Returns list of *Flavor*.

resource_class
alias of *novaclient.v2.flavors.Flavor*

update(*flavor*, *description=None*)
Update the description of the flavor.

Parameters

- **flavor** The *Flavor* (or its ID) to update.
- **description** The description to set on the flavor.

novaclient.v2.hypervisors module

Hypervisors interface

class novaclient.v2.hypervisors.**Hypervisor**(*manager*, *info*, *loaded=False*, *resp=None*)
Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

NAME_ATTR = 'hypervisor_hostname'

class novaclient.v2.hypervisors.**HypervisorManager**(*api*)
Bases: *novaclient.base.ManagerWithFind*

get(*hypervisor*)
Get a specific hypervisor.

Parameters hypervisor Either a Hypervisor object or an ID. Starting with microversion 2.53 the ID must be a UUID value.

is_alphanum_id_allowed = True

list(*detailed=True*, *marker=None*, *limit=None*)

Get a list of hypervisors.

Parameters

- **detailed** Include a detailed response.
- **marker** Begin returning hypervisors that appear later in the hypervisors list than that represented by this hypervisor ID. Starting with microversion 2.53 the marker must be a UUID hypervisor ID. (optional).
- **limit** maximum number of hypervisors to return (optional). Note the API server has a configurable default limit. If no limit is specified here or limit is larger than default, the default limit will be used.

resource_class

alias of *novaclient.v2.hypervisors.Hypervisor*

search(*hypervisor_match*, *servers=False*, *detailed=False*)

Get a list of matching hypervisors.

Parameters

- **hypervisor_match** The hypervisor host name or a portion of it. The hypervisor hosts are selected with the host name matching this pattern.
- **servers** If True, server information is also retrieved.
- **detailed** If True, detailed hypervisor information is returned. This requires API version 2.53 or greater.

statistics()

Get hypervisor statistics over all compute nodes.

Kept for backwards compatibility, new code should call hypervisor_stats.statistics() instead of hypervisors.statistics()

uptime(*hypervisor*)

Get the uptime for a specific hypervisor.

Parameters hypervisor Either a Hypervisor object or an ID. Starting with microversion 2.53 the ID must be a UUID value.

class novaclient.v2.hypervisors.**HypervisorStats**(*manager*, *info*, *loaded=False*, *resp=None*)

Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

class novaclient.v2.hypervisors.**HypervisorStatsManager**(*api*)

Bases: *novaclient.base.Manager*

resource_class
alias of *novaclient.v2.hypervisors.HypervisorStats*

statistics()

novaclient.v2.images module

class novaclient.v2.images.**GlanceManager**(*api*)
Bases: *novaclient.base.Manager*

Use glance directly from service catalog.

This is used to do name to id lookups for images and listing images for the image-with option to the boot command. Do not use it for anything else besides that. You have been warned.

find_image(*name_or_id*)
Find an image by name or id (user provided input).

find_images(*names_or_ids*)
Find multiple images by name or id (user provided input).

Parameters **names_or_ids** A list of strings to use to find images.

Returns novaclient.v2.images.Image objects for each images found

Raises

- *exceptions.NotFound* If one or more images is not found
- *exceptions.ClientException* If the image service returns any unexpected images.

NOTE: This method always makes two calls to the image service, even if only one image is provided by ID and is returned in the first query.

list()
Get a detailed list of all images.

Return type list of *Image*

resource_class
alias of *novaclient.v2.images.Image*

class novaclient.v2.images.**Image**(*manager*, *info*, *loaded=False*, *resp=None*)
Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

HUMAN_ID = True

novaclient.v2.instance_action module

class novaclient.v2.instance_action.**InstanceAction**(*manager*, *info*, *loaded=False*, *resp=None*)

Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

class novaclient.v2.instance_action.**InstanceActionManager**(*api*)

Bases: *novaclient.base.ManagerWithFind*

get(*server*, *request_id*)

Get details of an action performed on an instance.

Parameters **request_id** The request_id of the action to get.

list(*server*, *marker=None*, *limit=None*, *changes_since=None*, *changes_before=None*)

Get a list of actions performed on a server.

Parameters

- **server** The `Server` (or its ID)
- **marker** Begin returning actions that appear later in the action list than that represented by this action request id (optional).
- **limit** Maximum number of actions to return. (optional).
- **changes_since** List only instance actions changed later or equal to a certain point of time. The provided time should be an ISO 8061 formatted time. e.g. 2016-03-04T06:27:59Z . (optional).
- **changes_before** List only instance actions changed earlier or equal to a certain point of time. The provided time should be an ISO 8061 formatted time. e.g. 2016-03-05T06:27:59Z . (optional).

resource_class

alias of *novaclient.v2.instance_action.InstanceAction*

novaclient.v2.instance_usage_audit_log module

class novaclient.v2.instance_usage_audit_log.**InstanceUsageAuditLog**(*manager*, *info*, *loaded=False*, *resp=None*)

Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

class novaclient.v2.instance_usage_audit_log.**InstanceUsageAuditLogManager**(*api*)
Bases: novaclient.base.Manager

get(*before=None*)
Get server usage audits.

Parameters **before** Filters the response by the date and time before which to list usage audits.

resource_class
alias of novaclient.v2.instance_usage_audit_log.InstanceUsageAuditLog

novaclient.v2.keypairs module

Keypair interface

class novaclient.v2.keypairs.**Keypair**(*manager*, *info*, *loaded=False*, *resp=None*)
Bases: novaclient.base.Resource

A keypair is a ssh key that can be injected into a server on launch.

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

delete()
Delete this keypair.

Returns An instance of novaclient.base.TupleWithMeta

property id

class novaclient.v2.keypairs.**KeypairManager**(*api*)
Bases: novaclient.base.ManagerWithFind

create(*name*, *public_key=None*, *key_type='ssh'*, *user_id=None*)
Create a keypair

Parameters

- **name** name for the keypair to create
- **public_key** existing public key to import
- **key_type** keypair type to create
- **user_id** user to add.

`delete`(*key*, *user_id=None*)

Delete a keypair

Parameters

- **key** The *Keypair* (or its ID) to delete.
- **user_id** Id of key-pair owner (Admin only).

Returns An instance of novaclient.base.TupleWithMeta

`get`(*keypair*, *user_id=None*)

Get a keypair.

Parameters

- **keypair** The ID of the keypair to get.
- **user_id** Id of key-pair owner (Admin only).

Return type *Keypair*

`is_alphanum_id_allowed = True`

`keypair_prefix = 'os-keypairs'`

`list`(*user_id=None*, *marker=None*, *limit=None*)

Get a list of keypairs.

Parameters

- **user_id** Id of key-pairs owner (Admin only).
- **marker** Begin returning keypairs that appear later in the keypair list than that represented by this keypair name (optional).
- **limit** maximum number of keypairs to return (optional). Note the API server has a configurable default limit. If no limit is specified here or limit is larger than default, the default limit will be used.

`resource_class`

alias of *novaclient.v2.keypairs.Keypair*

novaclient.v2.limits module

class `novaclient.v2.limits.AbsoluteLimit`(*name*, *value*)

Bases: `object`

Data model that represents a single absolute limit.

class `novaclient.v2.limits.Limits`(*manager*, *info*, *loaded=False*, *resp=None*)

Bases: *novaclient.base.Resource*

A collection of RateLimit and AbsoluteLimit objects.

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes

- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

property absolute

property rate

class novaclient.v2.limits.**LimitsManager**(*api*)

Bases: *novaclient.base.Manager*

Manager object used to interact with limits resource.

get(*reserved=False*, *tenant_id=None*)

Get a specific extension.

Return type *Limits*

resource_class

alias of *novaclient.v2.limits.Limits*

class novaclient.v2.limits.**RateLimit**(*verb*, *uri*, *regex*, *value*, *remain*, *unit*, *next_available*)

Bases: object

Data model that represents a flattened view of a single rate limit.

novaclient.v2.migrations module

migration interface

class novaclient.v2.migrations.**Migration**(*manager*, *info*, *loaded=False*, *resp=None*)

Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

class novaclient.v2.migrations.**MigrationManager**(*api*)

Bases: *novaclient.base.ManagerWithFind*

list(*host=None*, *status=None*, *instance_uuid=None*, *marker=None*, *limit=None*, *changes_since=None*, *changes_before=None*, *migration_type=None*, *source_compute=None*, *user_id=None*, *project_id=None*)

Get a list of migrations. :param host: filter migrations by host name (optional). :param status: filter migrations by status (optional). :param instance_uuid: filter migrations by instance uuid (optional). :param marker: Begin returning migrations that appear later in the migrations list than that represented by this migration UUID (optional). :param limit: maximum number of migrations to return (optional). Note the API server has a configurable default limit. If no limit is specified here or limit is larger than default, the default limit will be used. :param changes_since: Only return migrations changed later or equal to a certain point of time. The provided time should be an ISO 8061 formatted time. e.g. 2016-03-04T06:27:59Z . (optional). :param changes_before: Only return migrations changed earlier or equal to a certain point of time. The provided time should be an ISO 8061 formatted time. e.g. 2016-03-05T06:27:59Z . (optional). :param migration_type: Filter migrations by type. Valid values are: evacuation, live-migration, migration, resize :param source_compute: Filter migrations by source compute host name. :param user_id: filter migrations by user (optional). :param project_id: filter migrations by project (optional).

resource_class
alias of *novaclient.v2.migrations.Migration*

novaclient.v2.networks module

Network interface.

`class novaclient.v2.networks.`**`Network`**`(`*manager*, *info*, *loaded=False*, *resp=None*`)`
Bases: *novaclient.base.Resource*

A network as defined in the Networking (Neutron) API.

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

HUMAN_ID = True

NAME_ATTR = 'name'

`class novaclient.v2.networks.`**`NeutronManager`**`(`*api*`)`
Bases: *novaclient.base.Manager*

A manager for name -> id lookups for neutron networks.

This uses neutron directly from service catalog. Do not use it for anything else besides that. You have been warned.

find_network(*name*)
Find a network by name (user provided input).

resource_class
alias of *novaclient.v2.networks.Network*

novaclient.v2.quota_classes module

`class novaclient.v2.quota_classes.`**`QuotaClassSet`**`(`*manager*, *info*, *loaded=False*, *resp=None*`)`
Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object

- `info` dictionary representing resource attributes
- `loaded` prevent lazy-loading if set to True
- `resp` Response or list of Response objects

`update`(**args*, ***kwargs*)

class novaclient.v2.quota_classes.`QuotaClassSetManager`(*api*)

Bases: `novaclient.base.Manager`

`get`(*class_name*)

`resource_class`

alias of `novaclient.v2.quota_classes.QuotaClassSet`

`update`(*class_name*, *instances=None*, *cores=None*, *ram=None*, *metadata_items=None*, *key_pairs=None*, *server_groups=None*, *server_group_members=None*)

novaclient.v2.quotas module

class novaclient.v2.quotas.`QuotaSet`(*manager*, *info*, *loaded=False*, *resp=None*)

Bases: `novaclient.base.Resource`

Populate and bind to a manager.

Parameters

- `manager` BaseManager object
- `info` dictionary representing resource attributes
- `loaded` prevent lazy-loading if set to True
- `resp` Response or list of Response objects

`update`(**args*, ***kwargs*)

class novaclient.v2.quotas.`QuotaSetManager`(*api*)

Bases: `novaclient.base.Manager`

`defaults`(*tenant_id*)

`delete`(*tenant_id*, *user_id=None*)

Delete quota for a tenant or for a user.

Parameters

- `tenant_id` A tenant for which quota is to be deleted
- `user_id` A user for which quota is to be deleted

Returns An instance of novaclient.base.TupleWithMeta

`get`(*tenant_id*, *user_id=None*, *detail=False*)

`resource_class`

alias of `novaclient.v2.quotas.QuotaSet`

`update`(*tenant_id*, *user_id=None*, *force=False*, *instances=None*, *cores=None*, *ram=None*, *metadata_items=None*, *key_pairs=None*, *server_groups=None*, *server_group_members=None*)

novaclient.v2.server_external_events module

External event triggering for servers, not to be used by users.

class novaclient.v2.server_external_events.**Event**(*manager*, *info*, *loaded=False*, *resp=None*)

Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

class novaclient.v2.server_external_events.**ServerExternalEventManager**(*api*)

Bases: *novaclient.base.Manager*

create(*events*)

Create one or more server events.

Param:events A list of dictionaries containing server_uuid, name, status, and tag (which may be absent)

resource_class

alias of *novaclient.v2.server_external_events.Event*

novaclient.v2.server_groups module

Server group interface.

class novaclient.v2.server_groups.**ServerGroup**(*manager*, *info*, *loaded=False*, *resp=None*)

Bases: *novaclient.base.Resource*

A server group.

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

delete()

Delete this server group.

Returns An instance of novaclient.base.TupleWithMeta

class novaclient.v2.server_groups.**ServerGroupsManager**(*api*)

Bases: *novaclient.base.ManagerWithFind*

Manage *ServerGroup* resources.

`create`(*name*, *policy*, *rules=None*)
Create (allocate) a server group.

Parameters

- **name** The name of the server group.
- **policy** Policy name to associate with the server group.
- **rules** The rules of policy which is a dict, can be applied to the policy, now only `max_server_per_host` for `anti-affinity` policy would be supported (optional).

Return type list of *ServerGroup*

`delete`(*id*)
Delete a specific server group.

Parameters **id** The ID of the *ServerGroup* to delete.

Returns An instance of novaclient.base.TupleWithMeta

`get`(*id*)
Get a specific server group.

Parameters **id** The ID of the *ServerGroup* to get.

Return type *ServerGroup*

`list`(*all_projects=False*, *limit=None*, *offset=None*)
Get a list of all server groups.

Parameters

- **all_projects** Lists server groups for all projects. (optional)
- **limit** Maximum number of server groups to return. (optional) Note the API server has a configurable default limit. If no limit is specified here or limit is larger than default, the default limit will be used.
- **offset** Use with *limit* to return a slice of server groups. *offset* is where to start in the groups list. (optional)

Returns list of *ServerGroup*.

`resource_class`
alias of *novaclient.v2.server_groups.ServerGroup*

novaclient.v2.server_migrations module

class novaclient.v2.server_migrations.**ServerMigration**(*manager*, *info*, *loaded=False*, *resp=None*)
Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes

- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

`class novaclient.v2.server_migrations.`**`ServerMigrationsManager`**(*api*)

Bases: `novaclient.base.ManagerWithFind`

`get`(*server*, *migration*)

Get a migration of a specified server

Parameters

- **server** The `Server` (or its ID)
- **migration** Migration id that will be gotten.

Returns An instance of novaclient.v2.server_migrations.ServerMigration

`list`(*server*)

Get a migrations list of a specified server

Parameters **server** The `Server` (or its ID)

Returns An instance of novaclient.base.ListWithMeta

`live_migrate_force_complete`(*server*, *migration*)

Force on-going live migration to complete

Parameters

- **server** The `Server` (or its ID)
- **migration** Migration id that will be forced to complete

Returns An instance of novaclient.base.TupleWithMeta

`live_migration_abort`(*server*, *migration*)

Cancel an ongoing live migration

Parameters

- **server** The `Server` (or its ID)
- **migration** Migration id that will be cancelled

Returns An instance of novaclient.base.TupleWithMeta

`resource_class`

alias of `novaclient.v2.server_migrations.ServerMigration`

novaclient.v2.servers module

Server interface.

`class novaclient.v2.servers.`**`NetworkInterface`**(*manager*, *info*, *loaded=False*, *resp=None*)

Bases: `novaclient.base.Resource`

Populate and bind to a manager.

Parameters

- **manager** BaseManager object

- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

property id

class novaclient.v2.servers.**SecurityGroup**(*manager*, *info*, *loaded=False*, *resp=None*)

Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

class novaclient.v2.servers.**Server**(*manager*, *info*, *loaded=False*, *resp=None*)

Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

HUMAN_ID = True

add_security_group(*security_group*)

Add a security group to an instance.

Parameters security_group The name of security group to add

Returns An instance of novaclient.base.DictWithMeta

add_tag(*tag*)

Add single tag to an instance.

backup(*backup_name*, *backup_type*, *rotation*)

Backup a server instance.

Parameters

- **backup_name** Name of the backup image
- **backup_type** The backup type, like daily or weekly
- **rotation** Int parameter representing how many backups to keep around.

Returns An instance of novaclient.base.TupleWithMeta

change_password(*password*)

Update the admin password for a server.

Parameters `password` string to set as the admin password on the server

Returns An instance of novaclient.base.TupleWithMeta

`clear_password()`
Get password for a Server.

`confirm_resize()`
Confirm that the resize worked, thus removing the original server.

Returns An instance of novaclient.base.TupleWithMeta

`create_image`(*image_name*, *metadata=None*)
Create an image based on this server.

Parameters

- `image_name` The name to assign the newly create image.
- `metadata` Metadata to assign to the image.

`delete()`
Delete (i.e. shut down and delete the image) this server.

Returns An instance of novaclient.base.TupleWithMeta

`delete_all_tags()`
Remove all tags from an instance.

`delete_tag`(*tag*)
Remove single tag from an instance.

`diagnostics()`
Diagnostics Retrieve server diagnostics.

`evacuate`(*host=None*, *password=None*)
Evacuate an instance from failed host to specified host.

Parameters

- `host` Name of the target host
- `password` string to set as admin password on the evacuated server.

Returns An instance of novaclient.base.TupleWithMeta

`force_delete()`
Force delete Force delete a server.

Returns An instance of novaclient.base.TupleWithMeta

`get_console_output`(*length=None*)
Get text console log output from Server.

Parameters `length` The number of lines you would like to retrieve (as int)

`get_console_url`(*console_type*)
Retrieve a console of a particular protocol and console_type

Parameters `console_type` Type of console

`get_mks_console()`
Get mks console for a Server.

`get_password`(*private_key=None*)
Get password for a Server.

Returns the clear password of an instance if private_key is provided, returns the ciphered password otherwise.

Parameters `private_key` Path to private key file for decryption (optional)

`get_rdp_console`(*console_type*)
Get rdp console for a Server.

Parameters `console_type` Type of console (rdp-html5)

`get_serial_console`(*console_type*)
Get serial console for a Server.

Parameters `console_type` Type of console (serial)

`get_spice_console`(*console_type*)
Get spice console for a Server.

Parameters `console_type` Type of console (spice-html5)

`get_vnc_console`(*console_type*)
Get vnc console for a Server.

Parameters `console_type` Type of console (novnc or xvpvnc)

`interface_attach`(*port_id*, *net_id*, *fixed_ip*, *tag=None*)
Attach a network interface to an instance with an optional tag.

`interface_detach`(*port_id*)
Detach a network interface from an instance.

`interface_list`()
List interfaces attached to an instance.

`list_security_group`()
List security group(s) of an instance.

`live_migrate`(*host=None*, *block_migration=None*)
Migrates a running instance to a new machine.

Parameters

- `host` destination host name.
- `block_migration` if True, do block_migration, the default value is None which is mapped to auto.

Returns An instance of novaclient.base.TupleWithMeta

`lock`(*reason=None*)
Lock Lock the instance from certain operations.

Parameters `reason` (Optional) The lock reason.

Returns An instance of novaclient.base.TupleWithMeta

`migrate`(*host=None*)
Migrate a server to a new host.

Parameters `host` (Optional) The target host.

Returns An instance of novaclient.base.TupleWithMeta

`property networks`
Generate a simplified list of addresses

Returns An OrderedDict, keyed by network name, and sorted by network name in ascending order.

`pause()`
Pause Pause the running server.

Returns An instance of novaclient.base.TupleWithMeta

`reboot`(*reboot_type='SOFT'*)
Reboot the server.

Parameters **`reboot_type`** either `REBOOT_SOFT` for a software-level reboot, or *REBOOT_HARD* for a virtual power cycle hard reboot.

Returns An instance of novaclient.base.TupleWithMeta

`rebuild`(*image*, *password=None*, *preserve_ephemeral=False*, ***kwargs*)
Rebuild shut down and then re-image this server.

Parameters

- **`image`** the `Image` (or its ID) to re-image with.
- **`password`** string to set as the admin password on the rebuilt server.
- **`preserve_ephemeral`** If True, request that any ephemeral device be preserved when rebuilding the instance. Defaults to False.

`remove_security_group`(*security_group*)
Remove a security group from an instance.

Returns An instance of novaclient.base.TupleWithMeta

`rescue`(*password=None*, *image=None*)
Rescue Rescue the problematic server.

Parameters

- **`password`** The admin password to be set in the rescue instance.
- **`image`** The `Image` to rescue with.

Returns An instance of novaclient.base.TupleWithMeta

`reset_network()`
Reset network of an instance.

Returns An instance of novaclient.base.TupleWithMeta

`reset_state`(*state='error'*)
Reset the state of an instance to active or error.

Returns An instance of novaclient.base.TupleWithMeta

`resize`(*flavor*, ***kwargs*)
Resize the servers resources.

Parameters **`flavor`** the `Flavor` (or its ID) to resize to.

Returns An instance of novaclient.base.TupleWithMeta

Until a resize event is confirmed with `confirm_resize()`, the old server will be kept around and youll be able to roll back to the old flavor quickly with `revert_resize()`. All resizes are automatically confirmed after 24 hours.

restore()
Restore Restore a server in soft-deleted state.

Returns An instance of novaclient.base.TupleWithMeta

resume()
Resume Resume the suspended server.

Returns An instance of novaclient.base.TupleWithMeta

revert_resize()
Revert a previous resize, switching back to the old server.

Returns An instance of novaclient.base.TupleWithMeta

set_tags(*tags*)
Set list of tags to an instance.

shelve()
Shelve Shelve the server.

Returns An instance of novaclient.base.TupleWithMeta

shelve_offload()
Shelve_offload Remove a shelved server from the compute node.

Returns An instance of novaclient.base.TupleWithMeta

start()
Start Start the paused server.

Returns An instance of novaclient.base.TupleWithMeta

stop()
Stop Stop the running server.

Returns An instance of novaclient.base.TupleWithMeta

suspend()
Suspend Suspend the running server.

Returns An instance of novaclient.base.TupleWithMeta

tag_list()
Get list of tags from an instance.

topology()
Retrieve server topology.

trigger_crash_dump()
Trigger crash dump in an instance

unlock()
Unlock Remove instance lock.

Returns An instance of novaclient.base.TupleWithMeta

unpause()

Unpause Unpause the paused server.

Returns An instance of novaclient.base.TupleWithMeta

unrescue()

Unrescue Unrescue the rescued server.

Returns An instance of novaclient.base.TupleWithMeta

unshelve(*availability_zone=None*)

Unshelve Unshelve the server.

Parameters **availability_zone** The specified availability zone name (Optional)

Returns An instance of novaclient.base.TupleWithMeta

update(*name=None*, *description=None*, *hostname=None*)

Update attributes of this server.

Parameters

- **name** Update the servers name.
- **description** Update the servers description.
- **hostname** Update the servers hostname.

Returns *Server*

class novaclient.v2.servers.**ServerManager**(*api*)

Bases: *novaclient.base.BootingManagerWithFind*

add_security_group(*server*, *security_group*)

Add a Security Group to an instance

Parameters

- **server** ID of the instance.
- **security_group** The name of security group to add.

Returns An instance of novaclient.base.DictWithMeta

add_tag(*server*, *tag*)

Add single tag to an instance.

backup(*server*, *backup_name*, *backup_type*, *rotation*)

Backup a server instance.

Parameters

- **server** The *Server* (or its ID) to share onto.
- **backup_name** Name of the backup image
- **backup_type** The backup type, like daily or weekly
- **rotation** Int parameter representing how many backups to keep around.

Returns An instance of novaclient.base.TupleWithMeta if the request microversion is < 2.45, otherwise novaclient.base.DictWithMeta.

`change_password`(*server*, *password*)
Update the password for a server.

Parameters `server` The `Server` (or its ID) for which the admin password is to be changed

Returns An instance of novaclient.base.TupleWithMeta

`clear_password`(*server*)
Clear the admin password of an instance

Remove the admin password for an instance from the metadata server.

Parameters `server` The `Server` (or its ID) for which the admin password is to be cleared

`confirm_resize`(*server*)
Confirm that the resize worked, thus removing the original server.

Parameters `server` The `Server` (or its ID) to share onto.

Returns An instance of novaclient.base.TupleWithMeta

`create`(*name*, *image*, *flavor*, *meta=None*, *files=None*, *reservation_id=False*, *min_count=None*, *max_count=None*, *security_groups=None*, *userdata=None*, *key_name=None*, *availability_zone=None*, *block_device_mapping=None*, *block_device_mapping_v2=None*, *nics=None*, *scheduler_hints=None*, *config_drive=None*, *disk_config=None*, *admin_pass=None*, *access_ip_v4=None*, *access_ip_v6=None*, *trusted_image_certificates=None*, *host=None*, *hypervisor_hostname=None*, *hostname=None*, ***kwargs*)
Create (boot) a new server.

In order to create a server with pre-existing ports that contain a `resource_request` value, such as for guaranteed minimum bandwidth quality of service support, microversion `2.72` is required.

Parameters

- **`name`** Something to name the server.
- **`image`** The `Image` to boot with.
- **`flavor`** The `Flavor` to boot onto.
- **`meta`** A dict of arbitrary key/value metadata to store for this server. Both keys and values must be <=255 characters.
- **`files`** A dict of files to overwrite on the server upon boot. Keys are file names (i.e. `/etc/passwd`) and values are the file contents (either as a string or as a file-like object). A maximum of five entries is allowed, and each file must be 10k or less. (deprecated starting with microversion 2.57)
- **`reservation_id`** return a reservation_id for the set of servers being requested, boolean.
- **`min_count`** (optional extension) The minimum number of servers to launch.
- **`max_count`** (optional extension) The maximum number of servers to launch.
- **`security_groups`** A list of security group names

- **userdata** user data to pass to be exposed by the metadata server this can be a file type object as well or a string.
- **key_name** (optional extension) name of previously created keypair to inject into the instance.
- **availability_zone** Name of the availability zone for instance placement.
- **block_device_mapping** (optional extension) A dict of block device mappings for this server.
- **block_device_mapping_v2** (optional extension) A list of block device mappings (dicts) for this server.
- **nics** An ordered list of nics (dicts) to be added to this server, with information about connected networks, fixed IPs, port etc. Beginning in microversion 2.37 this field is required and also supports a single string value of auto or none. The auto value means the Compute service will automatically allocate a network for the project if one is not available. This is the same behavior as not passing anything for nics before microversion 2.37. The none value tells the Compute service to not allocate any networking for the server.
- **scheduler_hints** (optional extension) arbitrary key-value pairs specified by the client to help boot an instance
- **config_drive** (optional extension) a boolean value to enable config drive
- **disk_config** (optional extension) control how the disk is partitioned when the server is created. possible values are AUTO or MANUAL.
- **admin_pass** (optional extension) add a user supplied admin password.
- **access_ip_v4** (optional extension) add alternative access ip v4
- **access_ip_v6** (optional extension) add alternative access ip v6
- **description** optional description of the server (allowed since microversion 2.19)
- **tags** A list of arbitrary strings to be added to the server as tags (allowed since microversion 2.52)
- **trusted_image_certificates** A list of trusted certificate IDs (allowed since microversion 2.63)
- **host** requested host to create servers (allowed since microversion 2.74)
- **hypervisor_hostname** requested hypervisor hostname to create servers (allowed since microversion 2.74)
- **hostname** requested hostname of server (allowed since microversion 2.90)

`create_image`(*server*, *image_name*, *metadata=None*)

Snapshot a server.

Parameters

- **server** The *Server* (or its ID) to share onto.
- **image_name** Name to give the snapshot image
- **metadata** Metadata to give newly-created image entity

Returns An instance of novaclient.base.StrWithMeta (The snapshot images UUID)

delete(*server*)
Delete (i.e. shut down and delete the image) this server.

Parameters **server** The *Server* (or its ID) to delete

Returns An instance of novaclient.base.TupleWithMeta

delete_all_tags(*server*)
Remove all tags from an instance.

delete_meta(*server*, *keys*)
Delete metadata from a server

Parameters

- **server** The *Server* to add metadata to
- **keys** A list of metadata keys to delete from the server

Returns An instance of novaclient.base.TupleWithMeta

delete_tag(*server*, *tag*)
Remove single tag from an instance.

diagnostics(*server*)
Retrieve server diagnostics.

Parameters **server** The *Server* (or its ID) for which diagnostics to be returned

Returns An instance of novaclient.base.TupleWithMeta

evacuate(*server*, *host=None*, *password=None*)
Evacuate a server instance.

Parameters

- **server** The *Server* (or its ID) to evacuate to.
- **host** Name of the target host.
- **password** string to set as password on the evacuated server.

Returns An instance of novaclient.base.TupleWithMeta

force_delete(*server*)
Force delete the server.

Parameters **server** The *Server* (or its ID) to force delete

Returns An instance of novaclient.base.TupleWithMeta

get(*server*)
Get a server.

Parameters **server** ID of the *Server* to get.

Return type *Server*

get_console_output(*server*, *length=None*)
Get text console log output from Server.

Parameters

- **server** The *Server* (or its ID) whose console output you would like to retrieve.
- **length** The number of tail loglines you would like to retrieve.

Returns An instance of novaclient.base.StrWithMeta or novaclient.base.UnicodeWithMeta

`get_console_url`(*server*, *console_type*)
Retrieve a console url of a server.

Parameters

- **server** server to get console url for
- **console_type** type can be novnc/xvpvnc for protocol vnc; spice-html5 for protocol spice; rdp-html5 for protocol rdp; serial for protocol serial. webmks for protocol mks (since version 2.8).

`get_mks_console`(*server*)
Get a mks console for an instance

Parameters **server** The *Server* (or its ID) to get console for.

Returns An instance of novaclient.base.DictWithMeta

`get_password`(*server*, *private_key=None*)
Get admin password of an instance

Returns the admin password of an instance in the clear if private_key is provided, returns the ciphered password otherwise.

Requires that openssl is installed and in the path

Parameters

- **server** The *Server* (or its ID) for which the admin password is to be returned
- **private_key** The private key to decrypt password (optional)

Returns An instance of novaclient.base.StrWithMeta or novaclient.base.BytesWithMeta or novaclient.base.UnicodeWithMeta

`get_rdp_console`(*server*, *console_type*)
Get a rdp console for an instance

Parameters

- **server** The *Server* (or its ID) to get console for.
- **console_type** Type of rdp console to get (rdp-html5)

Returns An instance of novaclient.base.DictWithMeta

`get_serial_console`(*server*, *console_type*)
Get a serial console for an instance

Parameters

- **server** The *Server* (or its ID) to get console for.
- **console_type** Type of serial console to get (serial)

Returns An instance of novaclient.base.DictWithMeta

`get_spice_console`(*server*, *console_type*)
Get a spice console for an instance

Parameters

- **`server`** The *`Server`* (or its ID) to get console for.
- **`console_type`** Type of spice console to get (spice-html5)

Returns An instance of novaclient.base.DictWithMeta

`get_vnc_console`(*server*, *console_type*)
Get a vnc console for an instance

Parameters

- **`server`** The *`Server`* (or its ID) to get console for.
- **`console_type`** Type of vnc console to get (novnc or xvpvnc)

Returns An instance of novaclient.base.DictWithMeta

`interface_attach`(*server*, *port_id*, *net_id*, *fixed_ip*, *tag=None*)
Attach a network_interface to an instance.

Parameters

- **`server`** The *`Server`* (or its ID) to attach to.
- **`port_id`** The port to attach. The port_id and net_id parameters are mutually exclusive.
- **`net_id`** The ID of the network to attach.
- **`fixed_ip`** The fixed IP addresses. If the fixed_ip is specified, the net_id has to be specified at the same time.
- **`tag`** The tag.

`interface_detach`(*server*, *port_id*)
Detach a network_interface from an instance.

Parameters

- **`server`** The *`Server`* (or its ID) to detach from.
- **`port_id`** The port to detach.

Returns An instance of novaclient.base.TupleWithMeta

`interface_list`(*server*)
List attached network interfaces

Parameters **`server`** The *`Server`* (or its ID) to query.

`ips`(*server*)
Return IP Addresses associated with the server.

Often a cheaper call then getting all the details for a server.

Parameters **`server`** The *`Server`* (or its ID) for which the IP adresses are to be returned

Returns An instance of novaclient.base.DictWithMeta

`list`(*detailed=True*, *search_opts=None*, *marker=None*, *limit=None*, *sort_keys=None*, *sort_dirs=None*)

Get a list of servers.

Parameters

- **`detailed`** Whether to return detailed server info (optional).
- **`search_opts`** Search options to filter out servers which dont match the search_opts (optional). The search opts format is a dictionary of key / value pairs that will be appended to the query string. For a complete list of keys see: https://docs.openstack.org/api-ref/compute/#list-servers
- **`marker`** Begin returning servers that appear later in the server list than that represented by this server id (optional).
- **`limit`** Maximum number of servers to return (optional). Note the API server has a configurable default limit. If no limit is specified here or limit is larger than default, the default limit will be used. If limit == -1, all servers will be returned.
- **`sort_keys`** List of sort keys
- **`sort_dirs`** List of sort directions

Return type list of *`Server`*

Examples:

client.servers.list() - returns detailed list of servers

client.servers.list(search_opts={status: ERROR}) - returns list of servers in error state.

client.servers.list(limit=10) - returns only 10 servers

`list_security_group`(*server*)

List Security Group(s) of an instance

Parameters **`server`** ID of the instance.

`live_migrate`(*server*, *host*, *block_migration*)

Migrates a running instance to a new machine.

Parameters

- **`server`** instance id which comes from nova list.
- **`host`** destination host name.
- **`block_migration`** if True, do block_migration, can be set as auto

Returns An instance of novaclient.base.TupleWithMeta

`lock`(*server*, *reason=None*)

Lock the server.

Parameters

- **`server`** The *`Server`* (or its ID) to lock
- **`reason`** (Optional) The lock reason.

Returns An instance of novaclient.base.TupleWithMeta

`migrate`(*server*, *host=None*)
Migrate a server to a new host.

Parameters

- **server** The `Server` (or its ID).
- **host** (Optional) The target host.

Returns An instance of novaclient.base.TupleWithMeta

`pause`(*server*)
Pause the server.

Parameters **server** The `Server` (or its ID) to pause

Returns An instance of novaclient.base.TupleWithMeta

`reboot`(*server*, *reboot_type='SOFT'*)
Reboot a server.

Parameters

- **server** The `Server` (or its ID) to share onto.
- **reboot_type** either `REBOOT_SOFT` for a software-level reboot, or *REBOOT_HARD* for a virtual power cycle hard reboot.

Returns An instance of novaclient.base.TupleWithMeta

`rebuild`(*server*, *image*, *password=None*, *disk_config=None*, *preserve_ephemeral=False*, *name=None*, *meta=None*, *files=None*, ***kwargs*)
Rebuild shut down and then re-image a server.

Parameters

- **server** The `Server` (or its ID) to share onto.
- **image** The `Image` (or its ID) to re-image with.
- **password** String to set as password on the rebuilt server.
- **disk_config** Partitioning mode to use on the rebuilt server. Valid values are AUTO or MANUAL
- **preserve_ephemeral** If True, request that any ephemeral device be preserved when rebuilding the instance. Defaults to False.
- **name** Something to name the server.
- **meta** A dict of arbitrary key/value metadata to store for this server. Both keys and values must be <=255 characters.
- **files** A dict of files to overwrite on the server upon boot. Keys are file names (i.e. `/etc/passwd`) and values are the file contents (either as a string or as a file-like object). A maximum of five entries is allowed, and each file must be 10k or less. (deprecated starting with microversion 2.57)
- **description** Optional description of the server. If None is specified, the existing description will be unset. (starting from microversion 2.19)

- **key_name** Optional key pair name for rebuild operation. If None is specified, the existing key will be unset. (starting from microversion 2.54)
- **userdata** Optional user data to pass to be exposed by the metadata server; this can be a file type object as well or a string. If None is specified, the existing user_data is unset. (starting from microversion 2.57)
- **trusted_image_certificates** A list of trusted certificate IDs or None to unset/reset the servers trusted image certificates (starting from microversion 2.63)
- **hostname** Optional hostname to configure for the instance. If None is specified, the existing hostname will be unset. (starting from microversion 2.90)

Returns `Server`

remove_security_group(*server*, *security_group*)
Remove a Security Group to an instance

Parameters

- **server** ID of the instance.
- **security_group** The name of security group to remove.

Returns An instance of novaclient.base.TupleWithMeta

rescue(*server*, *password=None*, *image=None*)
Rescue the server.

Parameters

- **server** The `Server` to rescue.
- **password** The admin password to be set in the rescue instance.
- **image** The `Image` to rescue with.

Returns An instance of novaclient.base.TupleWithMeta

reset_network(*server*)
Reset network of an instance.

Parameters **server** The `Server` for network is to be reset

Returns An instance of novaclient.base.TupleWithMeta

reset_state(*server*, *state='error'*)
Reset the state of an instance to active or error.

Parameters

- **server** ID of the instance to reset the state of.
- **state** Desired state; either active or error. Defaults to error.

Returns An instance of novaclient.base.TupleWithMeta

resize(*server*, *flavor*, *disk_config=None*, ***kwargs*)
Resize a servers resources.

Parameters

- **server** The `Server` (or its ID) to share onto.

- **flavor** the Flavor (or its ID) to resize to.
- **disk_config** partitioning mode to use on the rebuilt server. Valid values are AUTO or MANUAL

Returns An instance of novaclient.base.TupleWithMeta

Until a resize event is confirmed with *confirm_resize()*, the old server will be kept around and youll be able to roll back to the old flavor quickly with *revert_resize()*. All resizes are automatically confirmed after 24 hours.

resource_class
alias of *novaclient.v2.servers.Server*

restore(*server*)
Restore soft-deleted server.

Parameters server The *Server* (or its ID) to restore

Returns An instance of novaclient.base.TupleWithMeta

resume(*server*)
Resume the server.

Parameters server The *Server* (or its ID) to resume

Returns An instance of novaclient.base.TupleWithMeta

revert_resize(*server*)
Revert a previous resize, switching back to the old server.

Parameters server The *Server* (or its ID) to share onto.

Returns An instance of novaclient.base.TupleWithMeta

set_meta(*server*, *metadata*)
Set a servers metadata :param server: The *Server* to add metadata to :param metadata: A dict of metadata to be added to the server

set_meta_item(*server*, *key*, *value*)
Updates an item of server metadata :param server: The *Server* to add metadata to :param key: metadata key to update :param value: string value

set_tags(*server*, *tags*)
Set list of tags to an instance.

shelve(*server*)
Shelve the server.

Parameters server The *Server* (or its ID) to shelve

Returns An instance of novaclient.base.TupleWithMeta

shelve_offload(*server*)
Remove a shelved instance from the compute node.

Parameters server The *Server* (or its ID) to shelve offload

Returns An instance of novaclient.base.TupleWithMeta

start(*server*)
Start the server.

Parameters **server** The *Server* (or its ID) to start

Returns An instance of novaclient.base.TupleWithMeta

`stop`(*server*)

Stop the server.

Parameters **server** The *Server* (or its ID) to stop

Returns An instance of novaclient.base.TupleWithMeta

`suspend`(*server*)

Suspend the server.

Parameters **server** The *Server* (or its ID) to suspend

Returns An instance of novaclient.base.TupleWithMeta

`tag_list`(*server*)

Get list of tags from an instance.

`topology`(*server*)

Retrieve server topology.

Parameters **server** The *Server* (or its ID) for which topology to be returned

Returns An instance of novaclient.base.DictWithMeta

`static transform_userdata`(*userdata*)

`trigger_crash_dump`(*server*)

Trigger crash dump in an instance

`unlock`(*server*)

Unlock the server.

Parameters **server** The *Server* (or its ID) to unlock

Returns An instance of novaclient.base.TupleWithMeta

`unpause`(*server*)

Unpause the server.

Parameters **server** The *Server* (or its ID) to unpause

Returns An instance of novaclient.base.TupleWithMeta

`unrescue`(*server*)

Unrescue the server.

Parameters **server** The *Server* (or its ID) to unrescue

Returns An instance of novaclient.base.TupleWithMeta

`unshelve`(*server*, *availability_zone=None*)

Unshelve the server.

Parameters

- **server** The *Server* (or its ID) to unshelve
- **availability_zone** The specified availability zone name (Optional)

Returns An instance of novaclient.base.TupleWithMeta

update(*server*, *name=None*, *description=None*, *hostname=None*)
Update attributes of a server.

Parameters

- **server** The *Server* (or its ID) to update.
- **name** Update the servers name.
- **description** Update the servers description. If it equals to empty string(i.g.), the server description will be removed.
- **hostname** Update the servers hostname as recorded by the metadata service. Note that a separate utility running on the guest will be necessary to reflect these changes in the guest itself.

Returns *Server*

novaclient.v2.services module

Service interface.

class novaclient.v2.services.**Service**(*manager*, *info*, *loaded=False*, *resp=None*)
Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

class novaclient.v2.services.**ServiceManager**(*api*)
Bases: *novaclient.base.ManagerWithFind*

delete(*service_id*)
Delete a service.

Parameters **service_id** Before microversion 2.53, this must be an integer id and may not uniquely the service in a multi-cell deployment. Starting with microversion 2.53 this must be a UUID.

disable(*service_uuid*)
Disable the service specified by the service UUID ID.

Parameters **service_uuid** The UUID ID of the service to disable.

disable_log_reason(*service_uuid*, *reason*)
Disable the service with a reason.

Parameters

- **service_uuid** The UUID ID of the service to disable.
- **reason** The reason for disabling a service. The minimum length is 1 and the maximum length is 255.

enable(*service_uuid*)
Enable the service specified by the service UUID ID.

Parameters **service_uuid** The UUID ID of the service to enable.

force_down(*service_uuid*, *force_down*)
Update the services `forced_down` field specified by the service UUID ID.

Parameters

- **service_uuid** The UUID ID of the service.
- **force_down** Whether or not this service was forced down manually by an administrator. This value is useful to know that some 3rd party has verified the service should be marked down.

list(*host=None*, *binary=None*)
Get a list of services.

Parameters **host** destination host name.

resource_class
alias of *novaclient.v2.services.Service*

novaclient.v2.shell module

class novaclient.v2.shell.**Console**(*console_dict*)
Bases: `object`

class novaclient.v2.shell.**EvacuateHostResponse**(*manager*, *info*, *loaded=False*, *resp=None*)
Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

class novaclient.v2.shell.**HostServersMigrateResponse**(*manager*, *info*, *loaded=False*, *resp=None*)
Bases: *novaclient.base.Resource*

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

novaclient.v2.shell.**console_dict_accessor**(*cs*, *data*)

novaclient.v2.shell.**do_add_secgroup**(*cs*, *args*)
Add a Security Group to a server.

novaclient.v2.shell.**do_agent_create**(*cs*, *args*)
DEPRECATED Create new agent build.

novaclient.v2.shell.**do_agent_delete**(*cs*, *args*)
DEPRECATED Delete existing agent build.

novaclient.v2.shell.**do_agent_list**(*cs*, *args*)
DEPRECATED List all builds.

novaclient.v2.shell.**do_agent_modify**(*cs*, *args*)
DEPRECATED Modify existing agent build.

novaclient.v2.shell.**do_aggregate_add_host**(*cs*, *args*)
Add the host to the specified aggregate.

novaclient.v2.shell.**do_aggregate_cache_images**(*cs*, *args*)
Request images be cached.

novaclient.v2.shell.**do_aggregate_create**(*cs*, *args*)
Create a new aggregate with the specified details.

novaclient.v2.shell.**do_aggregate_delete**(*cs*, *args*)
Delete the aggregate.

novaclient.v2.shell.**do_aggregate_list**(*cs*, *args*)
Print a list of all aggregates.

novaclient.v2.shell.**do_aggregate_remove_host**(*cs*, *args*)
Remove the specified host from the specified aggregate.

novaclient.v2.shell.**do_aggregate_set_metadata**(*cs*, *args*)
Update the metadata associated with the aggregate.

novaclient.v2.shell.**do_aggregate_show**(*cs*, *args*)
Show details of the specified aggregate.

novaclient.v2.shell.**do_aggregate_update**(*cs*, *args*)
Update the aggregates name and optionally availability zone.

novaclient.v2.shell.**do_availability_zone_list**(*cs*, *_args*)
List all the availability zones.

novaclient.v2.shell.**do_backup**(*cs*, *args*)
Backup a server by creating a backup type snapshot.

novaclient.v2.shell.**do_boot**(*cs*, *args*)
Boot a new server.

novaclient.v2.shell.**do_clear_password**(*cs*, *args*)
Clear the admin password for a server from the metadata server. This action does not actually change the instance server password.

novaclient.v2.shell.**do_console_log**(*cs*, *args*)
Get console log output of a server.

novaclient.v2.shell.**do_delete**(*cs*, *args*)
Immediately shut down and delete specified server(s).

novaclient.v2.shell.**do_diagnostics**(*cs*, *args*)
Retrieve server diagnostics.

novaclient.v2.shell.**do_evacuate**(*cs*, *args*)
Evacuate server from failed host.

novaclient.v2.shell.**do_flavor_access_add**(*cs*, *args*)
Add flavor access for the given tenant.

novaclient.v2.shell.**do_flavor_access_list**(*cs*, *args*)
Print access information about the given flavor.

novaclient.v2.shell.**do_flavor_access_remove**(*cs*, *args*)
Remove flavor access for the given tenant.

novaclient.v2.shell.**do_flavor_create**(*cs*, *args*)
Create a new flavor.

novaclient.v2.shell.**do_flavor_delete**(*cs*, *args*)
Delete a specific flavor

novaclient.v2.shell.**do_flavor_key**(*cs*, *args*)
Set or unset extra_spec for a flavor.

novaclient.v2.shell.**do_flavor_list**(*cs*, *args*)
Print a list of available flavors (sizes of servers).

novaclient.v2.shell.**do_flavor_show**(*cs*, *args*)
Show details about the given flavor.

novaclient.v2.shell.**do_flavor_update**(*cs*, *args*)
Update the description of an existing flavor.

novaclient.v2.shell.**do_force_delete**(*cs*, *args*)
Force delete a server.

novaclient.v2.shell.**do_get_mks_console**(*cs*, *args*)
Get an MKS console to a server.

novaclient.v2.shell.**do_get_password**(*cs*, *args*)
Get the admin password for a server. This operation calls the metadata service to query metadata information and does not read password information from the server itself.

novaclient.v2.shell.**do_get_rdp_console**(*cs*, *args*)
Get a rdp console to a server.

novaclient.v2.shell.**do_get_serial_console**(*cs*, *args*)
Get a serial console to a server.

novaclient.v2.shell.**do_get_spice_console**(*cs*, *args*)
Get a spice console to a server.

novaclient.v2.shell.**do_get_vnc_console**(*cs*, *args*)
Get a vnc console to a server.

novaclient.v2.shell.**do_host_evacuate**(*cs*, *args*)
Evacuate all instances from failed host.

novaclient.v2.shell.**do_host_evacuate_live**(*cs*, *args*)
Live migrate all instances off the specified host to other available hosts.

novaclient.v2.shell.**do_host_meta**(*cs*, *args*)
Set or Delete metadata on all instances of a host.

novaclient.v2.shell.**do_host_servers_migrate**(*cs*, *args*)
Cold migrate all instances off the specified host to other available hosts.

novaclient.v2.shell.**do_hypervisor_list**(*cs*, *args*)
List hypervisors.

novaclient.v2.shell.**do_hypervisor_servers**(*cs*, *args*)
List servers belonging to specific hypervisors.

novaclient.v2.shell.**do_hypervisor_show**(*cs*, *args*)
Display the details of the specified hypervisor.

novaclient.v2.shell.**do_hypervisor_stats**(*cs*, *args*)

novaclient.v2.shell.**do_hypervisor_uptime**(*cs*, *args*)
Display the uptime of the specified hypervisor.

novaclient.v2.shell.**do_image_create**(*cs*, *args*)
Create a new image by taking a snapshot of a running server.

novaclient.v2.shell.**do_instance_action**(*cs*, *args*)
Show an action.

novaclient.v2.shell.**do_instance_action_list**(*cs*, *args*)
List actions on a server.

novaclient.v2.shell.**do_instance_usage_audit_log**(*cs*, *args*)
List/Get server usage audits.

novaclient.v2.shell.**do_interface_attach**(*cs*, *args*)
Attach a network interface to a server.

novaclient.v2.shell.**do_interface_detach**(*cs*, *args*)
Detach a network interface from a server.

novaclient.v2.shell.**do_interface_list**(*cs*, *args*)
List interfaces attached to a server.

novaclient.v2.shell.**do_keypair_add**(*cs*, *args*)
Create a new key pair for use with servers.

novaclient.v2.shell.**do_keypair_delete**(*cs*, *args*)
Delete keypair given by its name.

novaclient.v2.shell.**do_keypair_list**(*cs*, *args*)
Print a list of keypairs for a user

novaclient.v2.shell.**do_keypair_show**(*cs*, *args*)
Show details about the given keypair.

novaclient.v2.shell.**do_limits**(*cs*, *args*)
Print rate and absolute limits.

novaclient.v2.shell.**do_list**(*cs*, *args*)
List servers.

novaclient.v2.shell.**do_list_secgroup**(*cs*, *args*)
List Security Group(s) of a server.

novaclient.v2.shell.**do_live_migration**(*cs*, *args*)
Migrate running server to a new machine.

novaclient.v2.shell.**do_live_migration_abort**(*cs*, *args*)
Abort an on-going live migration.

novaclient.v2.shell.**do_live_migration_force_complete**(*cs*, *args*)
Force on-going live migration to complete.

novaclient.v2.shell.**do_lock**(*cs*, *args*)
Lock a server. A normal (non-admin) user will not be able to execute actions on a locked server.

novaclient.v2.shell.**do_meta**(*cs*, *args*)
Set or delete metadata on a server.

novaclient.v2.shell.**do_migrate**(*cs*, *args*)
Migrate a server.

novaclient.v2.shell.**do_migration_list**(*cs*, *args*)
Print a list of migrations.

novaclient.v2.shell.**do_pause**(*cs*, *args*)
Pause a server.

novaclient.v2.shell.**do_quota_class_show**(*cs*, *args*)
List the quotas for a quota class.

novaclient.v2.shell.**do_quota_class_update**(*cs*, *args*)
Update the quotas for a quota class.

novaclient.v2.shell.**do_quota_defaults**(*cs*, *args*)
List the default quotas for a tenant.

novaclient.v2.shell.**do_quota_delete**(*cs*, *args*)
Delete quota for a tenant/user so their quota will Revert back to default.

novaclient.v2.shell.**do_quota_show**(*cs*, *args*)
List the quotas for a tenant/user.

novaclient.v2.shell.**do_quota_update**(*cs*, *args*)
Update the quotas for a tenant/user.

novaclient.v2.shell.**do_reboot**(*cs*, *args*)
Reboot a server.

novaclient.v2.shell.**do_rebuild**(*cs*, *args*)
Shutdown, re-image, and re-boot a server.

novaclient.v2.shell.**do_refresh_network**(*cs*, *args*)
Refresh server network information.

novaclient.v2.shell.**do_remove_secgroup**(*cs*, *args*)
Remove a Security Group from a server.

novaclient.v2.shell.**do_rescue**(*cs*, *args*)
Reboots a server into rescue mode, which starts the machine from either the initial image or a specified image, attaching the current boot disk as secondary.

novaclient.v2.shell.**do_reset_network**(*cs*, *args*)
Reset network of a server.

novaclient.v2.shell.**do_reset_state**(*cs*, *args*)
Reset the state of a server.

novaclient.v2.shell.**do_resize**(*cs*, *args*)
Resize a server.

novaclient.v2.shell.**do_resize_confirm**(*cs*, *args*)
Confirm a previous resize.

novaclient.v2.shell.**do_resize_revert**(*cs*, *args*)
Revert a previous resize (and return to the previous VM).

novaclient.v2.shell.**do_restore**(*cs*, *args*)
Restore a soft-deleted server.

novaclient.v2.shell.**do_resume**(*cs*, *args*)
Resume a server.

novaclient.v2.shell.**do_server_group_create**(*cs*, *args*)
Create a new server group with the specified details.

novaclient.v2.shell.**do_server_group_delete**(*cs*, *args*)
Delete specific server group(s).

novaclient.v2.shell.**do_server_group_get**(*cs*, *args*)
Get a specific server group.

novaclient.v2.shell.**do_server_group_list**(*cs*, *args*)
Print a list of all server groups.

novaclient.v2.shell.**do_server_migration_list**(*cs*, *args*)
Get the migrations list of specified server.

novaclient.v2.shell.**do_server_migration_show**(*cs*, *args*)
Get the migration of specified server.

novaclient.v2.shell.**do_server_tag_add**(*cs*, *args*)
Add one or more tags to a server.

novaclient.v2.shell.**do_server_tag_delete**(*cs*, *args*)
Delete one or more tags from a server.

novaclient.v2.shell.**do_server_tag_delete_all**(*cs*, *args*)
Delete all tags from a server.

novaclient.v2.shell.**do_server_tag_list**(*cs*, *args*)
Get list of tags from a server.

novaclient.v2.shell.**do_server_tag_set**(*cs*, *args*)
Set list of tags to a server.

novaclient.v2.shell.**do_server_topology**(*cs*, *args*)
Retrieve server topology.

novaclient.v2.shell.**do_service_delete**(*cs*, *args*)
Delete the service by UUID ID.

If deleting a nova-compute service, be sure to stop the actual nova-compute process on the physical host before deleting the service with this command. Failing to do so can lead to the running service re-creating orphaned compute_nodes table records in the database.

`novaclient.v2.shell.do_service_disable(cs, args)`
Disable the service.

`novaclient.v2.shell.do_service_enable(cs, args)`
Enable the service.

`novaclient.v2.shell.do_service_force_down(cs, args)`
Force service to down.

`novaclient.v2.shell.do_service_list(cs, args)`
Show a list of all running services. Filter by host & binary.

`novaclient.v2.shell.do_set_password(cs, args)`
Change the admin password for a server.

`novaclient.v2.shell.do_shelve(cs, args)`
Shelve a server.

`novaclient.v2.shell.do_shelve_offload(cs, args)`
Remove a shelved server from the compute node.

`novaclient.v2.shell.do_show(cs, args)`
Show details about the given server.

`novaclient.v2.shell.do_ssh(cs, args)`
SSH into a server.

`novaclient.v2.shell.do_start(cs, args)`
Start the server(s).

`novaclient.v2.shell.do_stop(cs, args)`
Stop the server(s).

`novaclient.v2.shell.do_suspend(cs, args)`
Suspend a server.

`novaclient.v2.shell.do_trigger_crash_dump(cs, args)`
Trigger crash dump in an instance.

`novaclient.v2.shell.do_unlock(cs, args)`
Unlock a server.

`novaclient.v2.shell.do_unpause(cs, args)`
Unpause a server.

`novaclient.v2.shell.do_unrescue(cs, args)`
Restart the server from normal boot disk again.

`novaclient.v2.shell.do_unshelve(cs, args)`
Unshelve a server.

`novaclient.v2.shell.do_update(cs, args)`
Update the name or the description for a server.

`novaclient.v2.shell.do_usage(cs, args)`
Show usage data for a single tenant.

`novaclient.v2.shell.do_usage_list(cs, args)`
List usage data for all tenants.

`novaclient.v2.shell.`**`do_version_list`**`(`*cs*, *args*`)`
List all API versions.

`novaclient.v2.shell.`**`do_volume_attach`**`(`*cs*, *args*`)`
Attach a volume to a server.

`novaclient.v2.shell.`**`do_volume_attachments`**`(`*cs*, *args*`)`
List all the volumes attached to a server.

`novaclient.v2.shell.`**`do_volume_detach`**`(`*cs*, *args*`)`
Detach a volume from a server.

`novaclient.v2.shell.`**`do_volume_update`**`(`*cs*, *args*`)`
Update the attachment on the server.

If dest_volume is the same as the src_volume then the command migrates the data from the attached volume to the specified available volume and swaps out the active attachment to the new volume. Otherwise it only updates the parameters of the existing attachment.

`novaclient.v2.shell.`**`emit_duplicated_image_with_warning`**`(`*img*, *image_with*`)`

`novaclient.v2.shell.`**`print_console`**`(`*cs*, *data*`)`

novaclient.v2.usage module

Usage interface.

`class` `novaclient.v2.usage.`**`Usage`**`(`*manager*, *info*, *loaded=False*, *resp=None*`)`
Bases: *`novaclient.base.Resource`*

Usage contains information about a tenants physical resource usage

Populate and bind to a manager.

Parameters

- **`manager`** BaseManager object
- **`info`** dictionary representing resource attributes
- **`loaded`** prevent lazy-loading if set to True
- **`resp`** Response or list of Response objects

`get``()`
Support for lazy loading details.

Some clients, such as novaclient have the option to lazy load the details, details which can be loaded with this function.

`class` `novaclient.v2.usage.`**`UsageManager`**`(`*api*`)`
Bases: *`novaclient.base.ManagerWithFind`*

Manage *`Usage`* resources.

`get``(`*tenant_id*, *start*, *end*, *marker=None*, *limit=None*`)`
Get usage for a specific tenant.

Parameters

- **`tenant_id`** Tenant ID to fetch usage for

- **start** `datetime.datetime` Start date in UTC
- **end** `datetime.datetime` End date in UTC
- **marker** Begin returning usage data for instances that appear later in the instance list than that represented by this instance UUID (optional).
- **limit** Maximum number of instances to include in the usage (optional). Note the API server has a configurable default limit. If no limit is specified here or limit is larger than default, the default limit will be used.

Return type *Usage*

list(*start*, *end*, *detailed=False*, *marker=None*, *limit=None*)
Get usage for all tenants

Parameters

- **start** `datetime.datetime` Start date in UTC
- **end** `datetime.datetime` End date in UTC
- **detailed** Whether to include information about each instance whose usage is part of the report
- **marker** Begin returning usage data for instances that appear later in the instance list than that represented by this instance UUID (optional).
- **limit** Maximum number of instances to include in the usage (optional). Note the API server has a configurable default limit. If no limit is specified here or limit is larger than default, the default limit will be used.

Return type list of *Usage*.

resource_class
alias of *novaclient.v2.usage.Usage*

usage_prefix = 'os-simple-tenant-usage'

novaclient.v2.versions module

version interface

class novaclient.v2.versions.**Version**(*manager*, *info*, *loaded=False*, *resp=None*)
Bases: *novaclient.base.Resource*

Compute REST API information

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

class novaclient.v2.versions.**VersionManager**(*api*)
Bases: *novaclient.base.ManagerWithFind*

get_current()

list()
: List all versions.

resource_class
: alias of *novaclient.v2.versions.Version*

novaclient.v2.volumes module

Volume interface

class novaclient.v2.volumes.**Volume**(*manager*, *info*, *loaded=False*, *resp=None*)
: Bases: *novaclient.base.Resource*

A volume is an extra block level storage to the OpenStack instances.

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

NAME_ATTR = 'display_name'

class novaclient.v2.volumes.**VolumeManager**(*api*)
: Bases: *novaclient.base.Manager*

Manage *Volume* resources. This is really about volume attachments.

create_server_volume(*server_id*, *volume_id*, *device=None*, *tag=None*, *delete_on_termination=False*)
: Attach a volume identified by the volume ID to the given server ID

Parameters

- **server_id** The ID of the server.
- **volume_id** The ID of the volume to attach.
- **device** The device name (optional).
- **tag** The tag (optional).
- **delete_on_termination** Marked whether to delete the attached volume when the server is deleted (optional).

Return type *Volume*

delete_server_volume(*server_id*, *volume_id=None*, *attachment_id=None*)
: Detach a volume identified by the volume ID from the given server

Parameters

- **server_id** The ID of the server
- **volume_id** The ID of the volume to attach

Returns An instance of novaclient.base.TupleWithMeta

`get_server_volume`(*server_id*, *volume_id=None*, *attachment_id=None*)
Get the volume identified by the volume ID, that is attached to the given server ID

Parameters

- `server_id` The ID of the server
- `volume_id` The ID of the volume to attach

Return type `Volume`

`get_server_volumes`(*server_id*)
Get a list of all the attached volumes for the given server ID

Parameters `server_id` The ID of the server

Return type list of `Volume`

`resource_class`
alias of `novaclient.v2.volumes.Volume`

`update_server_volume`(*server_id*, *src_volid*, *dest_volid*, *delete_on_termination=None*)
Swaps the existing volume attachment to point to a new volume.

Takes a server, a source (attached) volume and a destination volume and performs a hypervisor assisted data migration from src to dest volume, detaches the original (source) volume and attaches the new destination volume. Note that not all backing hypervisor drivers support this operation and it may be disabled via policy.

Parameters

- `server_id` The ID of the server
- `source_volume` The ID of the src volume
- `dest_volume` The ID of the destination volume
- `delete_on_termination` Marked whether to delete the attached volume when the server is deleted (optional).

Return type `Volume`

Module contents

Submodules

novaclient.api_versions module

`class novaclient.api_versions.APIVersion`(*version_str=None*)
Bases: `object`

This class represents an API Version Request.

This class provides convenience methods for manipulation and comparison of version numbers that we need to do to implement microversions.

Create an API version object.

Parameters **`version_str`** String representation of APIVersionRequest. Correct format is X.Y, where X and Y are int values. None value should be used to create Null APIVersionRequest, which is equal to 0.0

`get_string()`
Version string representation.

Converts object to string representation which if used to create an APIVersion object results in the same version.

`is_latest()`

`is_null()`

`matches`(*min_version*, *max_version*)
Matches the version object.

Returns whether the version object represents a version greater than or equal to the minimum version and less than or equal to the maximum version.

Parameters

- **`min_version`** Minimum acceptable version.
- **`max_version`** Maximum acceptable version.

Returns boolean

If min_version is null then there is no minimum limit. If max_version is null then there is no maximum limit. If self is null then raise ValueError

class `novaclient.api_versions.`**`VersionedMethod`**(*name*, *start_version*, *end_version*, *func*)
Bases: `object`

Versioning information for a single method

Parameters

- **`name`** Name of the method
- **`start_version`** Minimum acceptable version
- **`end_version`** Maximum acceptable_version
- **`func`** Method to call

Minimum and maximums are inclusive

`novaclient.api_versions.`**`check_headers`**(*response*, *api_version*)
Checks that microversion header is in response.

`novaclient.api_versions.`**`check_major_version`**(*api_version*)
Checks major part of `APIVersion` obj is supported.

Raises *`novaclient.exceptions.UnsupportedVersion`* if major part is not supported

`novaclient.api_versions.`**`deprecated_after`**(*version*)

`novaclient.api_versions.`**`discover_version`**(*client*, *requested_version*)
Discover most recent version supported by API and client.

Checks `requested_version` and returns the most recent version supported by both the API and the client.

Parameters

- **client** client object
- **requested_version** requested version represented by APIVersion obj

Returns APIVersion

`novaclient.api_versions.`**`get_api_version`**`(`*version_string*`)`
Returns checked APIVersion object

`novaclient.api_versions.`**`get_available_major_versions`**`()`

`novaclient.api_versions.`**`get_substitutions`**`(`*func_name*, *api_version=None*`)`

`novaclient.api_versions.`**`update_headers`**`(`*headers*, *api_version*`)`
Set microversion headers if api_version is not null

`novaclient.api_versions.`**`wraps`**`(`*start_version*, *end_version=None*`)`

novaclient.base module

Base utilities to build API operation managers and objects on top of.

class `novaclient.base.`**`BootingManagerWithFind`**`(`*api*`)`
Bases: *novaclient.base.ManagerWithFind*

Like a *ManagerWithFind*, but has the ability to boot servers.

class `novaclient.base.`**`BytesWithMeta`**`(`*value*, *resp*`)`
Bases: `bytes`, *novaclient.base.RequestIdMixin*

class `novaclient.base.`**`DictWithMeta`**`(`*values*, *resp*`)`
Bases: `dict`, *novaclient.base.RequestIdMixin*

class `novaclient.base.`**`HookableMixin`**
Bases: `object`

Mixin so classes can register and run hooks.

classmethod **`add_hook`**`(`*hook_type*, *hook_func*`)`
Add a new hook of specified type.

Parameters

- **cls** class that registers hooks
- **hook_type** hook type, e.g., __pre_parse_args__
- **hook_func** hook function

classmethod **`run_hooks`**`(`*hook_type*, **args*, ***kwargs*`)`
Run all hooks of specified type.

Parameters

- **cls** class that registers hooks
- **hook_type** hook type, e.g., __pre_parse_args__
- **args** args to be passed to every hook function
- **kwargs** kwargs to be passed to every hook function

`class novaclient.base.`**`ListWithMeta`**`(`*values*, *resp*`)`

Bases: `list`, *`novaclient.base.RequestIdMixin`*

`class novaclient.base.`**`Manager`**`(`*api*`)`

Bases: *`novaclient.base.HookableMixin`*

Manager for API service.

Managers interact with a particular type of API (servers, flavors, images, etc.) and provide CRUD operations for them.

`alternate_service_type``(`*default*, *allowed_types=()*`)`

`property api_version`

`cache_lock = <unlocked _thread.RLock object owner=0 count=0>`

`property client`

`completion_cache``(`*cache_type*, *obj_class*, *mode*`)`

The completion cache for bash autocompletion.

The completion cache store items that can be used for bash autocompletion, like UUIDs or human-friendly IDs.

A resource listing will clear and repopulate the cache.

A resource create will append to the cache.

Delete is not handled because listings are assumed to be performed often enough to keep the cache reasonably up-to-date.

`convert_into_with_meta``(`*item*, *resp*`)`

`resource_class = None`

`write_to_completion_cache``(`*cache_type*, *val*`)`

`class novaclient.base.`**`ManagerWithFind`**`(`*api*`)`

Bases: *`novaclient.base.Manager`*

Like a *Manager*, but with additional *find()*/*findall()* methods.

`find``(`***kwargs*`)`

Find a single item with attributes matching `**kwargs`.

`findall``(`***kwargs*`)`

Find all items with attributes matching `**kwargs`.

`abstract list``()`

`class novaclient.base.`**`RequestIdMixin`**

Bases: `object`

Wrapper class to expose x-openstack-request-id to the caller.

`append_request_ids``(`*resp*`)`

Add request_ids as an attribute to the object

Parameters **`resp`** Response object or list of Response objects

`property request_ids`

`request_ids_setup``()`

class novaclient.base.**Resource**(*manager*, *info*, *loaded=False*, *resp=None*)
Bases: *novaclient.base.RequestIdMixin*

Base class for OpenStack resources (tenant, user, etc.).

This is pretty much just a bag for attributes.

Populate and bind to a manager.

Parameters

- **manager** BaseManager object
- **info** dictionary representing resource attributes
- **loaded** prevent lazy-loading if set to True
- **resp** Response or list of Response objects

HUMAN_ID = False

NAME_ATTR = 'name'

property api_version

get()
Support for lazy loading details.

Some clients, such as novaclient have the option to lazy load the details, details which can be loaded with this function.

property human_id
Human-readable ID which can be used for bash completion.

is_loaded()

set_info(*key*, *value*)

set_loaded(*val*)

to_dict()

class novaclient.base.**StrWithMeta**(*value*, *resp*)
Bases: str, *novaclient.base.RequestIdMixin*

class novaclient.base.**TupleWithMeta**(*values*, *resp*)
Bases: tuple, *novaclient.base.RequestIdMixin*

novaclient.base.**getid**(*obj*)
Get objects ID or object.

Abstracts the common pattern of allowing both an object or an objects ID as a parameter when dealing with relationships.

novaclient.client module

OpenStack Client interface. Handles the REST calls and responses.

`novaclient.client.`**`Client`**(*version*, *username=None*, *password=None*, *project_id=None*, *auth_url=None*, ***kwargs*)

Initialize client object based on given version.

HOW-TO: The simplest way to create a client instance is initialization with your credentials:

```
>>> from novaclient import client
>>> nova = client.Client(VERSION, USERNAME, PASSWORD,
...                      PROJECT_ID, AUTH_URL)
```

Here `VERSION` can be a string or `novaclient.api_versions.APIVersion` obj. If you prefer string value, you can use `1.1` (deprecated now), `2` or `2.X` (where X is a microversion).

Alternatively, you can create a client instance using the keystoneauth session API. See The novaclient Python API page at python-novaclients doc.

`class` `novaclient.client.`**`SessionClient`**(**args*, ***kwargs*)

Bases: `keystoneauth1.adapter.LegacyJsonAdapter`

`client_name = 'python-novaclient'`

`client_version = '17.7.1'`

`get_timings`()

`request`(*url*, *method*, ***kwargs*)

`reset_timings`()

`novaclient.client.`**`discover_extensions`**(**args*, ***kwargs*)

Returns the list of extensions, which can be discovered by python path and by entry-point novaclient.extension.

novaclient.crypto module

`exception` `novaclient.crypto.`**`DecryptionFailure`**

Bases: `Exception`

`novaclient.crypto.`**`decrypt_password`**(*private_key*, *password*)

Base64 decodes password and unencrypts it with private key.

Requires openssl binary available in the path.

novaclient.exceptions module

Exception definitions.

exception `novaclient.exceptions.`**`BadRequest`**(*code*, *message=None*, *details=None*, *request_id=None*, *url=None*, *method=None*)

Bases: *`novaclient.exceptions.ClientException`*

HTTP 400 - Bad request: you sent some malformed data.

`http_status = 400`

`message = 'Bad request'`

exception `novaclient.exceptions.`**`ClientException`**(*code*, *message=None*, *details=None*, *request_id=None*, *url=None*, *method=None*)

Bases: `Exception`

The base exception class for all exceptions this library raises.

`message = 'Unknown Error'`

exception `novaclient.exceptions.`**`CommandError`**

Bases: `Exception`

exception `novaclient.exceptions.`**`Conflict`**(*code*, *message=None*, *details=None*, *request_id=None*, *url=None*, *method=None*)

Bases: *`novaclient.exceptions.ClientException`*

HTTP 409 - Conflict

`http_status = 409`

`message = 'Conflict'`

exception `novaclient.exceptions.`**`Forbidden`**(*code*, *message=None*, *details=None*, *request_id=None*, *url=None*, *method=None*)

Bases: *`novaclient.exceptions.ClientException`*

HTTP 403 - Forbidden: your credentials dont give you access to this resource.

`http_status = 403`

`message = 'Forbidden'`

exception `novaclient.exceptions.`**`HTTPNotImplemented`**(*code*, *message=None*, *details=None*, *request_id=None*, *url=None*, *method=None*)

Bases: *`novaclient.exceptions.ClientException`*

HTTP 501 - Not Implemented: the server does not support this operation.

`http_status = 501`

`message = 'Not Implemented'`

exception `novaclient.exceptions.`**`InstanceInDeletedState`**

Bases: `Exception`

Instance is in the deleted state.

exception novaclient.exceptions.**InvalidUsage**

Bases: `RuntimeError`

This function call is invalid in the way you are using this client.

Due to the transition to using keystoneauth some function calls are no longer available. You should make a similar call to the session object instead.

exception novaclient.exceptions.**MethodNotAllowed**(*code*, *message=None*, *details=None*, *request_id=None*, *url=None*, *method=None*)

Bases: *novaclient.exceptions.ClientException*

HTTP 405 - Method Not Allowed

http_status = 405

message = 'Method Not Allowed'

exception novaclient.exceptions.**NoUniqueMatch**

Bases: `Exception`

exception novaclient.exceptions.**NotAcceptable**(*code*, *message=None*, *details=None*, *request_id=None*, *url=None*, *method=None*)

Bases: *novaclient.exceptions.ClientException*

HTTP 406 - Not Acceptable

http_status = 406

message = 'Not Acceptable'

exception novaclient.exceptions.**NotFound**(*code*, *message=None*, *details=None*, *request_id=None*, *url=None*, *method=None*)

Bases: *novaclient.exceptions.ClientException*

HTTP 404 - Not found

http_status = 404

message = 'Not found'

exception novaclient.exceptions.**OverLimit**(**args*, ***kwargs*)

Bases: *novaclient.exceptions.RetryAfterException*

HTTP 413 - Over limit: youre over the API limits for this time period.

http_status = 413

message = 'Over limit'

exception novaclient.exceptions.**RateLimit**(**args*, ***kwargs*)

Bases: *novaclient.exceptions.RetryAfterException*

HTTP 429 - Rate limit: youve sent too many requests for this time period.

http_status = 429

message = 'Rate limit'

exception novaclient.exceptions.**ResourceInErrorState**(*obj*)

Bases: `Exception`

Resource is in the error state.

exception novaclient.exceptions.**ResourceNotFound**

Bases: `Exception`

Error in getting the resource.

exception novaclient.exceptions.**RetryAfterException**(**args*, ***kwargs*)

Bases: `novaclient.exceptions.ClientException`

The base exception class for ClientExceptions that use Retry-After header.

exception novaclient.exceptions.**Unauthorized**(*code*, *message=None*, *details=None*, *request_id=None*, *url=None*, *method=None*)

Bases: `novaclient.exceptions.ClientException`

HTTP 401 - Unauthorized: bad credentials.

http_status = 401

message = 'Unauthorized'

exception novaclient.exceptions.**UnsupportedAttribute**(*argument_name*, *start_version*, *end_version=None*)

Bases: `AttributeError`

Indicates that the user is trying to transmit the argument to a method, which is not supported by selected version.

exception novaclient.exceptions.**UnsupportedConsoleType**(*console_type*)

Bases: `Exception`

Indicates that the user is trying to use an unsupported console type when retrieving console urls of servers.

exception novaclient.exceptions.**UnsupportedVersion**

Bases: `Exception`

Indicates that the user is trying to use an unsupported version of the API.

exception novaclient.exceptions.**VersionNotFoundForAPIMethod**(*version*, *method*)

Bases: `Exception`

msg_fmt = "API version '%(vers)s' is not supported on '%(method)s' method."

novaclient.exceptions.**from_response**(*response*, *body*, *url*, *method=None*)

Return an instance of an ClientException or subclass based on a requests response.

Usage:

```
resp, body = requests.request(...)
if resp.status_code != 200:
    raise exception_from_response(resp, rest.text)
```

novaclient.extension module

class novaclient.extension.**Extension**(*name*, *module*)

Bases: *novaclient.base.HookableMixin*

Extension descriptor.

SUPPORTED_HOOKS = ('__pre_parse_args__', '__post_parse_args__')

novaclient.i18n module

oslo_i18n integration module for novaclient.

See https://docs.openstack.org/oslo.i18n/latest/user/usage.html .

novaclient.shell module

Command-line interface to the OpenStack Nova API.

class novaclient.shell.**DeprecatedAction**(*option_strings*, *dest*, *help=None*, *real_action=None*, *use=None*, ***kwargs*)

Bases: `argparse.Action`

An argparse action for deprecated options.

This class is an `argparse.Action` subclass that allows command line options to be explicitly deprecated. It modifies the help text for the option to indicate that its deprecated (unless help has been suppressed using `argparse.SUPPRESS`), and provides a means to specify an alternate option to use using the `use` keyword argument to `argparse.ArgumentParser.add_argument()`. The original action may be specified with the `real_action` keyword argument, which has the same interpretation as the `action` argument to `argparse.ArgumentParser.add_argument()`, with the addition of the special nothing action which completely ignores the option (other than emitting the deprecation warning). Note that the deprecation warning is only emitted once per specific option string.

Note: If the `real_action` keyword argument specifies an unknown action, no warning will be emitted unless the action is used, due to limitations with the method used to resolve the action names.

Initialize a `DeprecatedAction` instance.

Parameters

- **option_strings** The recognized option strings.
- **dest** The attribute that will be set.
- **help** Help text. This will be updated to indicate the deprecation, and if `use` is provided, that text will be included as well.
- **real_action** The actual action to invoke. This is interpreted the same way as the `action` parameter.
- **use** Text explaining which option to use instead.

class novaclient.shell.**NovaClientArgumentParser**(**args*, ***kwargs*)

Bases: `argparse.ArgumentParser`

error(*message: string*)
Prints a usage message incorporating the message to stderr and exits.

class novaclient.shell.**OpenStackComputeShell**
Bases: object

do_bash_completion(*_args*)
Prints all of the commands and options to stdout so that the nova.bash_completion script doesnt have to hard code them.

do_help(*args*)
Display help about this program or one of its subcommands.

get_base_parser(*argv*)

get_subcommand_parser(*version*, *do_help=False*, *argv=None*)

main(*argv*)

setup_debugging(*debug*)

times = []

class novaclient.shell.**OpenStackHelpFormatter**(*prog*, *indent_increment=2*, *max_help_position=32*, *width=None*)
Bases: argparse.HelpFormatter

start_section(*heading*)

novaclient.shell.**main**()

novaclient.utils module

novaclient.utils.**add_arg**(*func*, **args*, ***kwargs*)
Bind CLI arguments to a shell.py *do_foo* function.

novaclient.utils.**arg**(**args*, ***kwargs*)
Decorator for CLI args.

Example:

```
>>> @arg("name", help="Name of the new entity")
... def entity_create(args):
...     pass
```

novaclient.utils.**do_action_on_many**(*action*, *resources*, *success_msg*, *error_msg*)
Helper to run an action on many resources.

novaclient.utils.**env**(**args*, ***kwargs*)
Returns the first environment variable set.

If all are empty, defaults to or keyword arg *default*.

novaclient.utils.**find_resource**(*manager*, *name_or_id*, *wrap_exception=True*, ***find_args*)
Helper for the _find_* methods.

novaclient.utils.**flatten_dict**(*data*)
Return a new dict whose sub-dicts have been merged into the original. Each of the parents keys are prepended to the childs to prevent collisions. Any string elements will be JSON parsed before flattening.

```
>>> flatten_dict({'service': {'host':'cloud9@compute-068', 'id': 143}})
{'service_host': colud9@compute-068', 'service_id': 143}
```

novaclient.utils.**format_security_groups**(*groups*)

novaclient.utils.**format_servers_list_networks**(*server*)

novaclient.utils.**get_service_type**(*f*)
Retrieves service type from function.

novaclient.utils.**get_url_with_filter**(*url*, *filters*)

novaclient.utils.**is_integer_like**(*val*)
Returns validation of a value as an integer.

novaclient.utils.**isunauthenticated**(*func*)
Checks if the function does not require authentication.

Mark such functions with the *@unauthenticated* decorator.

Returns bool

novaclient.utils.**make_field_formatter**(*attr*, *filters=None*)
Given an object attribute, return a formatted field name and a formatter suitable for passing to print_list.

Optionally pass a dict mapping attribute names to a function. The function will be passed the value of the attribute and should return the string to display.

novaclient.utils.**prepare_query_string**(*params*)
Convert dict params to query string

novaclient.utils.**pretty_choice_dict**(*d*)
Returns a formatted dict as key=value.

novaclient.utils.**pretty_choice_list**(*l*)

novaclient.utils.**print_dict**(*d*, *dict_property='Property'*, *dict_value='Value'*, *wrap=0*)

novaclient.utils.**print_list**(*objs*, *fields*, *formatters={}*, *sortby_index=None*)

novaclient.utils.**record_time**(*times*, *enabled*, **args*)
Record the time of a specific action.

Parameters

- **times** A list of tuples holds time data.
- **enabled** Whether timing is enabled.
- **args** Other data to be stored besides time data, these args will be joined to a string.

novaclient.utils.**safe_issubclass**(**args*)
Like issubclass, but will just return False if not a class.

novaclient.utils.**service_type**(*stype*)

Adds service_type attribute to decorated function.

Usage:

```
@service_type('volume')
def mymethod(f):
...
```

novaclient.utils.**unauthenticated**(*func*)

Adds unauthenticated attribute to decorated function.

Usage:

```
>>> @unauthenticated
... def mymethod(f):
...     pass
```

novaclient.utils.**validate_flavor_metadata_keys**(*keys*)

Module contents

CHAPTER

FOUR

CONTRIBUTOR GUIDE

4.1 Basic Information

4.1.1 So You Want to Contribute

For general information on contributing to OpenStack, please check out the contributor guide to get started. It covers all the basics that are common to all OpenStack projects: the accounts you need, the basics of interacting with our Gerrit review system, how we communicate as a community, etc.

Below will cover the more project specific information you need to get started with python-novaclient.

Communication

Please refer how-to-get-involved.

Contacting the Core Team

The overall structure of the Nova team including python-novaclient is documented on the wiki.

New Feature Planning

If you want to propose a new feature please read the blueprints page.

Task Tracking

We track our tasks in Launchpad.

If youre looking for some smaller, easier work item to pick up and get started on, search for the low-hanging-fruit tag.

Reporting a Bug

You found an issue and want to make sure we are aware of it? You can do so on Launchpad. More info about Launchpad usage can be found on OpenStack docs page.

Getting Your Patch Merged

All changes proposed to the python-novaclient requires two `Code-Review +2` votes from python-novaclient core reviewers before one of the core reviewers can approve patch by giving `Workflow +1` vote..

4.2 Developer Guide

4.2.1 Adding support for a new microversion

If a new microversion is added on the nova side, then support must be added on the *python-novaclient* side also. The following procedure describes how to add support for a new microversion in *python-novaclient*.

1. Update `API_MAX_VERSION`

   Set `API_MAX_VERSION` in `novaclient/__init__.py` to the version you are going to support.

   Note: Microversion support should be added one by one in order. For example, microversion 2.74 should be added right after microversion 2.73. Microversion 2.74 should not be added right after microversion 2.72 or earlier.

2. Update CLI and Python API

   Update CLI (`novaclient/v2/shell.py`) and/or Python API (`novaclient/v2/*.py`) to support the microversion.

3. Add tests

   Add unit tests for the change. Add unit tests for the previous microversion to check raising an error or an exception when new arguments or parameters are specified. Add functional tests if necessary.

   Add the microversion in the `exclusions` in the `test_versions` method of the `novaclient.tests.unit.v2.test_shell.ShellTest` class if there are no versioned wrapped method changes for the microversion. The versioned wrapped methods have `@api_versions.wraps` decorators.

   For example (microversion 2.72 example):

```
exclusions = set([
    (snipped...)
    72,  # There are no version-wrapped shell method changes for this.
])
```

4. Update the CLI reference

   Update the CLI reference (`doc/source/cli/nova.rst`) if the CLI commands and/or arguments are modified.

5. Add a release note

   Add a release note for the change. The release note should include a link to the description for the microversion in the Compute API Microversion History.

6. Commit message

   The description of the blueprint and dependency on the patch in nova side should be added in the commit message. For example:

   ```
   Implements: blueprint remove-force-flag-from-live-migrate-and-evacuate
   Depends-On: https://review.opendev.org/#/c/634600/
   ```

See the following examples:

- Microversion 2.71 - show server group
- API microversion 2.69: Handles Down Cells
- Microversion 2.68: Remove forced live migrations, evacuations
- Add support changes-before for microversion 2.66
- Microversion 2.64 - Use new format policy in server group

4.2.2 Testing

The preferred way to run the unit tests is using `tox`. There are multiple test targets that can be run to validate the code.

`tox -e pep8` Style guidelines enforcement.

`tox -e py38` Traditional unit testing (Python 3.8).

`tox -e functional` Live functional testing against an existing OpenStack instance. (Python 3.8)

`tox -e cover` Generate a coverage report on unit testing.

Functional testing assumes the existence of a *clouds.yaml* file as supported by os-client-config. It assumes the existence of a cloud named *devstack* that behaves like a normal DevStack installation with a demo and an admin user/tenant - or clouds named *functional_admin* and *functional_nonadmin*.

Refer to Consistent Testing Interface for more details.

4.2.3 Deprecating commands

There are times when commands need to be deprecated due to rename or removal. The process for command deprecation is:

1. Push up a change for review which deprecates the command(s).

   - The change should print a deprecation warning to `stderr` each time a deprecated command is used.
   - That warning message should include a rough timeline for when the command will be removed and what should be used instead, if anything.
   - The description in the help text for the deprecated command should mark that it is deprecated.
   - The change should include a release note with the `deprecations` section filled out.

   - The deprecation cycle is typically the first client release *after* the next *full* nova server release so that there is at least six months of deprecation.

2. Once the change is approved, have a member of the nova-release team release a new version of *python-novaclient*.

3. Example: https://review.opendev.org/#/c/185141/

   This change was made while the nova 12.0.0 Liberty release was in development. The current version of *python-novaclient* at the time was 2.25.0. Once the change was merged, *python-novaclient* 2.26.0 was released. Since there was less than six months before 12.0.0 would be released, the deprecation cycle ran through the 13.0.0 nova server release.

PYTHON MODULE INDEX