

---

# **Qarnot Documentation**

***Release v2.4.1+0.gbdfd853.dirty***

**Qarnot computing**

**Jul 12, 2021**



---

## Contents

---

|          |                            |           |
|----------|----------------------------|-----------|
| <b>1</b> | <b>Reference Docs</b>      | <b>3</b>  |
| <b>2</b> | <b>Indices and tables</b>  | <b>47</b> |
|          | <b>Python Module Index</b> | <b>49</b> |
|          | <b>Index</b>               | <b>51</b> |



qarnot is a Python library designed to interact with cloud computing service. It allows users to launch, manage and monitor payloads running on distributed computing nodes deployed in Qarnot's digital heaters.

You can find samples and detailed information on <http://computing.qarnot.com>.



### 1.1 Installation

We recommend you to set up a Python virtual environment. To do so with Python2 you need *virtualenv*, you should be able to install it using for example one of the following commands:

```
$ apt-get install python-virtualenv
$ easy_install virtualenv
$ pip install virtualenv
```

Or for Python3:

```
$ apt-get install python3-venv
$ pip3 install virtualenv
```

Once *virtualenv* is installed you can create your own environment by running the following commands in the project directory:

```
$ virtualenv venv
New python executable in venv/bin/python
Installing setuptools, pip, wheel...done.
```

Or with Python3;

```
$ python3 -m venv venv
```

Then each time you want to use your virtual environment you have to activate it by running this command:

```
$ . venv/bin/activate
```

Finally you have to install in your environment the Qarnot SDK:

```
pip install qarnot
```

If you plan to send large files to the API, we advise you to install the optional requests-toolbelt dependency in order not to overuse your memory:

```
pip install requests-toolbelt
```

You are now ready to use the Qarnot SDK.

## 1.2 Basic usage

### 1.2.1 Configuration

A basic usage of the Qarnot require a configuration file (eg: *qarnot.conf*). Here is a basic one, check [Connection](#) for details.

```
1 [cluster]
2 # url of the REST API
3 url=https://api.qarnot.com
4 [client]
5 # auth string of the client
6 token=token
```

### 1.2.2 Script

And here is a little sample to start a task running your *myscript.py* Python script.

```
1 import qarnot
2
3 conn = qarnot.connection.Connection('qarnot.conf')
4 task = conn.create_task('hello world', 'ubuntu')
5 task.constants['DOCKER_CMD'] = 'echo hello world from ${FRAME_ID}!'
6 task.run()
7 print(task.stdout())
```

## 1.3 SDK Documentation

### 1.3.1 Connection

Module describing a connection.

```
class qarnot.connection.Connection(fileconf=None, client_token=None, cluster_url=None,
                                     cluster_unsafe=False, cluster_timeout=None, stor-
                                     age_url=None, storage_unsafe=False, retry_count=5,
                                     retry_wait=1.0, cluster_custom_certificate=None, stor-
                                     age_custom_certificate=None)
```

Bases: `object`

Represents the couple cluster/user to which submit tasks.

```
__init__(fileconf=None, client_token=None, cluster_url=None, cluster_unsafe=False, clus-
          ter_timeout=None, storage_url=None, storage_unsafe=False, retry_count=5,
          retry_wait=1.0, cluster_custom_certificate=None, storage_custom_certificate=None)
```

Create a connection to a cluster with given config file, options or environment variables.



Available environment variable are `QARNOT_CLUSTER_URL`, `QARNOT_CLUSTER_UNSAFE`, `QARNOT_CLUSTER_TIMEOUT` and `QARNOT_CLIENT_TOKEN`.

#### Parameters

- `fileconf` (*str* or *dict*) – path to a qarnot configuration file or a corresponding dict
- `client_token` (*str*) – API Token
- `cluster_url` (*str*) – (optional) Cluster url.
- `cluster_unsafe` (*bool*) – (optional) Disable certificate check
- `cluster_timeout` (*int*) – (optional) Timeout value for every request
- `storage_url` (*str*) – (optional) Storage service url.
- `storage_unsafe` (*bool*) – (optional) Disable certificate check
- `retry_count` (*int*) – (optional) ConnectionError retry count. Default to 5.
- `retry_wait` (*float*) – (optional) Retry on error wait time, progressive. (wait \* (retry\_count - retry\_num)). Default to 1s

Configuration sample:

```
[cluster]
# url of the REST API
url=https://localhost
# No SSL verification ?
unsafe=False
[client]
# auth string of the client
token=login
[storage]
url=https://storage
unsafe=False
```

#### s3client

Pre-configured s3 client object.

**Return type** `list(S3.Client)`

**Returns** A list of ObjectSummary resources

#### s3resource

Pre-configured s3 resource object.

**Return type** `list(S3.ServiceResource)`

**Returns** A list of ObjectSummary resources

#### user\_info

Get information of the current user on the cluster.

**Return type** `UserInfo`

**Returns** Requested information.

**Raises**

- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details

**buckets ()**

Get the list of buckets.

**Return type** `list(class:~qarnot.bucket.Bucket)`.

**Returns** List of buckets

**pools (summary=True, tags\_intersect=None)**

Get the list of pools stored on this cluster for this user.

**Parameters**

- **summary** (*bool*) – only get the summaries.
- **tags\_intersect** (list of *str*, optional) – Desired filtering tags, all of them

**Return type** List of *Pool*.

**Returns** Pools stored on the cluster owned by the user.

**Raises**

- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details

**tasks (tags=None, summary=True, tags\_intersect=None)**

Get the list of tasks stored on this cluster for this user.

**Parameters**

- **tags** (list of *str*, optional) – Desired filtering tags, any of them
- **summary** (*bool*) – only get the summaries.
- **tags\_intersect** (list of *str*, optional) – Desired filtering tags, all of them

**Return type** List of *Task*.

**Returns** Tasks stored on the cluster owned by the user.

**Raises**

- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details

**jobs ()**

Get the list of jobs stored on this cluster for this user.

**Raises**

- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details

**retrieve\_pool (uuid)**

Retrieve a `qarnot.pool.Pool` from its uuid

**Parameters** **uuid** (*str*) – Desired pool uuid

**Return type** *Pool*

**Returns** Existing pool defined by the given uuid

**Raises**

- `garnot.exceptions.MissingPoolException` – pool does not exist
- `garnot.exceptions.UnauthorizedException` – invalid credentials
- `garnot.exceptions.QarnotGenericException` – API general error, see message for details

#### **retrieve\_task** (*uuid*)

Retrieve a `garnot.task.Task` from its uuid

**Parameters** **uuid** (*str*) – Desired task uuid

**Return type** *Task*

**Returns** Existing task defined by the given uuid

**Raises**

- `garnot.exceptions.MissingTaskException` – task does not exist
- `garnot.exceptions.UnauthorizedException` – invalid credentials
- `garnot.exceptions.QarnotGenericException` – API general error, see message for details

#### **retrieve\_job** (*uuid*)

Retrieve a `garnot.job.Job` from its uuid

**Parameters** **uuid** (*str*) – Desired job uuid

**Return type** *Job*

**Returns** Existing job defined by the given uuid

**Raises**

- `garnot.exceptions.MissingJobException` – job does not exist
- `garnot.exceptions.UnauthorizedException` – invalid credentials
- `garnot.exceptions.QarnotGenericException` – API general error, see message for details

#### **retrieve\_or\_create\_bucket** (*uuid*)

Retrieve a `Bucket` from its description, or create a new one.

**Parameters** **uuid** (*str*) – the bucket uuid (name)

**Return type** *Bucket*

**Returns** Existing or newly created bucket defined by the given name

#### **retrieve\_bucket** (*uuid*)

Retrieve a `Bucket` from its uuid (name)

**Parameters** **uuid** (*str*) – Desired bucket uuid (name)

**Return type** *Bucket*

**Returns** Existing bucket defined by the given uuid (name)

**Raises** `botocore.exceptions.ClientError`: Bucket does not exist, or invalid credentials

#### **create\_pool** (*name, profile, instancecount=1, shortname=None*)

Create a new `Pool`.

**Parameters**

- **name** (*str*) – given name of the pool

- **profile** (*str*) – which profile to use with this pool
- **instancecount** (*int*) – number of instances to run for the pool
- **shortname** (*str*) – optional unique friendly shortname of the pool

**Return type** *Pool*

**Returns** The created *Pool*.

---

**Note:** See available profiles with *profiles()*.

---

**create\_elastic\_pool** (*name*, *profile*, *minimum\_total\_slots*=0, *maximum\_total\_slots*=1, *minimum\_idle\_slots*=0, *minimum\_idle\_time\_seconds*=0, *resize\_factor*=1, *resize\_period*=90, *shortname*=None)

Create a new *Pool*.

**Parameters**

- **name** (*str*) – given name of the pool
- **profile** (*str*) – which profile to use with this pool
- **minimum\_total\_slots** (*int*) – minimum number of instances to run for the pool
- **maximum\_total\_slots** (*int*) – maximum number of instances to run for the pool
- **minimum\_idle\_slots** (*int*) – the number of instances that can be idle before considering shrinking the pool
- **minimum\_idle\_time\_seconds** (*int*) – the number of seconds before considering shrinking the pool
- **resize\_factor** (*float*) – the speed with which we grow the pool to meet the demand
- **resize\_period** (*int*) – the time between the load checks that decide if the pool grows or shrinks
- **shortname** (*str*) – optional unique friendly shortname of the pool

**Return type** *Pool*

**Returns** The created *Pool*.

---

**Note:** See available profiles with *profiles()*.

---

**create\_task** (*name*, *profile\_or\_pool*=None, *instancecount\_or\_range*=1, *shortname*=None, *job*=None)

Create a new *Task*.

**Parameters**

- **name** (*str*) – given name of the task
- **profile\_or\_pool** (*str* or *Pool* or None) – which profile to use with this task, or which Pool to run task, or which job to attach it to
- **instancecount\_or\_range** (*int* or *str*) – number of instances, or ranges on which to run task. Defaults to 1.
- **shortname** (*str*) – optional unique friendly shortname of the task
- **job** (*Job*) – which job to attach the task to

**Return type** *Task*

**Returns** The created *Task*.

---

**Note:** See available profiles with *profiles()*.

---

**submit\_tasks** (*tasks*)

Submit a list of *Task*.

:param List of *Task*. :raises qarnot.exceptions.QarnotGenericException: API general error, see message for details

---

**Note:** Will ensure all added files are on the resource bucket regardless of their uploading mode.

---

**profiles** ()

Get list of profiles available on the cluster.

**Return type** list of *Profile*

**Raises**

- *qarnot.exceptions.UnauthorizedException* – invalid credentials
- *qarnot.exceptions.QarnotGenericException* – API general error, see message for details

**retrieve\_profile** (*name*)

Get details of a profile from its name.

**Return type** *Profile*

**Raises**

- *qarnot.exceptions.UnauthorizedException* – invalid credentials
- *qarnot.exceptions.QarnotGenericException* – API general error, see message for details

**create\_bucket** (*name*)

Create a new *Bucket*. If the bucket already exist, retrieve the existing bucket.

**Parameters** *name* (*str*) – bucket name

**Return type** *qarnot.bucket.Bucket*

**Returns** The created or existing *Bucket*.

**create\_job** (*name*, *pool=None*, *shortname=None*, *useDependencies=False*)

Create a new *Job*.

**Parameters**

- *name* (*str*) – given name of the job
- *pool* (*Pool* or *None*) – which Pool to submit the job in,
- *shortname* (*str*) – userfriendly job name
- *use\_dependencies* – allow dependencies between tasks in this job

**Returns** The created *Job*.

```
class garnot.connection.UserInfo(info)
    Bases: object

    Information about a garnot user.

    __init__(info)
        x.__init__(...) initializes x; see help(type(x)) for signature

    email = None
        Type str
        User email address.

    max_bucket = None
        Type int
        Maximum number of buckets allowed (resource and result buckets).

    quota_bytes_bucket = None
        Type int
        Total storage space allowed for the user's buckets (in Bytes).

    used_quota_bytes_bucket = None
        Type int
        Total storage space used by the user's buckets (in Bytes).

    task_count = None
        Type int
        Total number of tasks belonging to the user.

    max_task = None
        Type int
        Maximum number of tasks the user is allowed to create.

    running_task_count = None
        Type int
        Number of tasks currently in 'Submitted' state.

    max_running_task = None
        Type int
        Maximum number of running tasks.

    max_instances = None
        Type int
        Maximum number of instances.

class garnot.connection.Profile(info)
    Bases: object

    Information about a profile.

    __init__(info)
        x.__init__(...) initializes x; see help(type(x)) for signature
```

**name** = None

Type `str`

Name of the profile.

**constants** = None

Type List of (`str`, `str`)

List of couples (name, value) representing constants for this profile and their default values.

`__repr__()` <==> `repr(x)`

## 1.3.2 Compute

### Task

Module to handle a task.

**class** `qarnot.task.Task`(`connection`, `name`, `profile_or_pool=None`, `instancecount_or_range=1`, `shortname=None`, `job=None`)

Bases: `object`

Represents a Qarnot task.

---

**Note:** A `Task` must be created with `qarnot.connection.Connection.create_task()` or retrieved with `qarnot.connection.Connection.tasks()` or `qarnot.connection.Connection.retrieve_task()`.

---

`__init__`(`connection`, `name`, `profile_or_pool=None`, `instancecount_or_range=1`, `shortname=None`, `job=None`)  
Create a new `Task`.

#### Parameters

- **connection** (`qarnot.connection.Connection`) – the cluster on which to send the task
- **name** (`str`) – given name of the task
- **profile\_or\_pool** (`str` or `Pool` or `None`) – which profile to use with this task, or which Pool to run task,
- **instancecount\_or\_range** (`int` or `str`) – number of instances or ranges on which to run task
- **shortname** (`str`) – userfriendly task name
- **job** (`Job`) – which job to attach the task to

**run** (`output_dir=None`, `job_timeout=None`, `live_progress=False`, `results_progress=None`)  
Submit a task, wait for the results and download them if required.

#### Parameters

- **output\_dir** (`str`) – (optional) path to a directory that will contain the results
- **job\_timeout** (`float`) – (optional) Number of seconds before the task `abort()` if it is not already finished
- **live\_progress** (`bool`) – (optional) display a live progress

- **results\_progress** (*bool* or *function(float, float, str)*) – (optional) can be a callback (read,total,filename) or True to display a progress bar

#### Raises

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.MaxTaskException` – Task quota reached
- `qarnot.exceptions.NotEnoughCreditsException` – Not enough credits
- `qarnot.exceptions.UnauthorizedException` – invalid credentials

---

**Note:** Will ensure all added file are on the resource bucket regardless of their uploading mode.

---

---

**Note:** If this function is interrupted (script killed for example), but the task is submitted, the task will still be executed remotely (results will not be downloaded)

---

**Warning:** Will override `output_dir` content.

**resume** (*output\_dir*, *job\_timeout=None*, *live\_progress=False*, *results\_progress=None*)

Resume waiting for this task if it is still in submitted mode. Equivalent to `wait()` + `download_results()`.

#### Parameters

- **output\_dir** (*str*) – path to a directory that will contain the results
- **job\_timeout** (*float*) – Number of seconds before the task `abort()` if it is not already finished
- **live\_progress** (*bool*) – display a live progress
- **results\_progress** (*bool* or *function(float, float, str)*) – can be a callback (read,total,filename) or True to display a progress bar

#### Raises

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.MissingTaskException` – task does not exist

---

**Note:** Do nothing if the task has not been submitted.

---

**Warning:** Will override `output_dir` content.

**submit** ()

Submit task to the cluster if it is not already submitted.

#### Raises



- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.MaxTaskException` – Task quota reached
- `qarnot.exceptions.NotEnoughCreditsException` – Not enough credits
- `qarnot.exceptions.UnauthorizedException` – invalid credentials

---

**Note:** Will ensure all added files are on the resource bucket regardless of their uploading mode.

---

---

**Note:** To get the results, call `download_results()` once the job is done.

---

#### **abort ()**

Abort this task if running.

##### **Raises**

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.MissingTaskException` – task does not exist

#### **update\_resources ()**

Update resources for a running task.

**The typical workflow is as follows:**

1. Upload new files on your resource bucket,
2. Call this method,
3. The new files will appear on all the compute nodes in the `$DOCKER_WORKDIR` folder

Note: There is no way to know when the files are effectively transferred. This information is available on the compute node only. Note: The update is additive only: files deleted from the bucket will NOT be deleted from the task's resources directory.

##### **Raises**

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.MissingTaskException` – task does not exist

#### **delete (purge\_resources=False, purge\_results=False)**

Delete this task on the server.

##### **Parameters**

- **purge\_resources** (*bool*) – parameter value is used to determine if the bucket is also deleted. Defaults to False.
- **purge\_results** (*bool*) – parameter value is used to determine if the bucket is also deleted. Defaults to False.

##### **Raises**

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.MissingTaskException` – task does not exist

**update** (*flushcache=False*)

Update the task object from the REST Api. The `flushcache` parameter can be used to force the update, otherwise a cached version of the object will be served when accessing properties of the object. Some methods will flush the cache, like `submit()`, `abort()`, `wait()` and `instant()`. Cache behavior is configurable with `auto_update` and `update_cache_time`.

**Raises**

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.MissingTaskException` – task does not represent a valid one

**classmethod from\_json** (*connection, json\_task, is\_summary=False*)

Create a Task object from a json task.

**Parameters**

- **connection** (`qarnot.connection.Connection`) – the cluster connection
- **json\_task** (*dict*) – Dictionary representing the task

**Returns** The created `Task`.

**commit** ()

Replicate local changes on the current object instance to the REST API

**Raises**

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials

---

**Note:** When updating buckets' properties, auto update will be disabled until commit is called.

---

**wait** (*timeout=None, live\_progress=False*)

Wait for this task until it is completed.

**Parameters**

- **timeout** (*float*) – maximum time (in seconds) to wait before returning (None => no timeout)
- **live\_progress** (*bool*) – display a live progress

**Return type** `bool`

**Returns** Is the task finished

**Raises**

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details

- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.MissingTaskException` – task does not represent a valid one

#### **snapshot** (*interval*)

Start snapshotting results. If called, this task’s results will be periodically updated, instead of only being available at the end.

Snapshots will be taken every *interval* second from the time the task is submitted.

**Parameters** `interval` (*int*) – the interval in seconds at which to take snapshots

#### **Raises**

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.MissingTaskException` – task does not represent a valid one

---

**Note:** To get the temporary results, call `download_results()`.

---

#### **instant** ()

Make a snapshot of the current task.

#### **Raises**

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.MissingTaskException` – task does not exist

---

**Note:** To get the temporary results, call `download_results()`.

---

#### **state**

**Type** `str`

**Getter** return this task’s state

State of the task.

#### **Value is in**

- UnSubmitted
- Submitted
- PartiallyDispatched
- FullyDispatched
- PartiallyExecuting
- FullyExecuting
- UploadingResults
- DownloadingResults

- Cancelled
- Success
- Failure

**Warning:** this is the state of the task when the object was retrieved, call `update()` for up to date value.

#### resources

**Type** `list(Bucket)`

**Getter** Returns this task's resources bucket

**Setter** Sets this task's resources bucket

Represents resource files.

#### results

**Type** `Bucket`

**Getter** Returns this task's results bucket

**Setter** Sets this task's results bucket

Represents results files.

**download\_results** (*output\_dir*, *progress=None*)

Download results in given *output\_dir*.

#### Parameters

- **output\_dir** (*str*) – local directory for the retrieved files.
- **progress** (*bool* or *function(float, float, str)*) – can be a callback (read,total,filename) or True to display a progress bar

#### Raises

- `qarnot.exceptions.MissingBucketException` – the bucket is not on the server
- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials

**Warning:** Will override *output\_dir* content.

#### stdout ()

Get the standard output of the task since the submission of the task.

**Return type** `str`

**Returns** The standard output.

#### Raises

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details

- `garnot.exceptions.UnauthorizedException` – invalid credentials
- `garnot.exceptions.MissingTaskException` – task does not exist

---

**Note:** The buffer is circular, if stdout is too big, prefer calling `fresh_stdout()` regularly.

---

#### **fresh\_stdout()**

Get what has been written on the standard output since last time this function was called or since the task has been submitted.

**Return type** `str`

**Returns** The new output since last call.

**Raises**

- `garnot.exceptions.QarnotGenericException` – API general error, see message for details
- `garnot.exceptions.UnauthorizedException` – invalid credentials
- `garnot.exceptions.MissingTaskException` – task does not exist

#### **stderr()**

Get the standard error of the task since the submission of the task.

**Return type** `str`

**Returns** The standard error.

**Raises**

- `garnot.exceptions.QarnotGenericException` – API general error, see message for details
- `garnot.exceptions.UnauthorizedException` – invalid credentials
- `garnot.exceptions.MissingTaskException` – task does not exist

---

**Note:** The buffer is circular, if stderr is too big, prefer calling `fresh_stderr()` regularly.

---

#### **fresh\_stderr()**

Get what has been written on the standard error since last time this function was called or since the task has been submitted.

**Return type** `str`

**Returns** The new error messages since last call.

**Raises**

- `garnot.exceptions.QarnotGenericException` – API general error, see message for details
- `garnot.exceptions.UnauthorizedException` – invalid credentials
- `garnot.exceptions.MissingTaskException` – task does not exist

#### **uuid**

**Type** `str`

**Getter** Returns this task's uuid

The task's uuid.

Automatically set when a task is submitted.

#### **name**

**Type** `str`

**Getter** Returns this task's name

**Setter** Sets this task's name

The task's name.

Can be set until task is submitted.

#### **shortname**

**Type** `str`

**Getter** Returns this task's shortname

**Setter** Sets this task's shortname

The task's shortname, must be DNS compliant and unique, if not provided, will default to `uuid`.

Can be set until task is submitted.

#### **tags**

**Type** `:class:list(str)`

**Getter** Returns this task's tags

**Setter** Sets this task's tags

Custom tags.

#### **pool**

**Type** `Pool`

**Getter** Returns this task's pool

**Setter** Sets this task's pool

The pool to run the task in.

Can be set until `run()` is called.

**Warning:** This property is mutually exclusive with `profile`

#### **profile**

**Type** `str`

**Getter** Returns this task's profile

**Setter** Sets this task's profile

The profile to run the task with.

Can be set until `run()` or `submit()` is called.

**Warning:** This property is mutually exclusive with `pool`

**instancecount****Type** `int`**Getter** Returns this task's instance count**Setter** Sets this task's instance count

Number of instances needed for the task.

Can be set until `run()` or `submit()` is called.**Raises** **AttributeError** – if `advanced_range` is not None when setting this property**Warning:** This property is mutually exclusive with `advanced_range`**running\_core\_count****Type** `int`**Getter** Returns this task's running core count

Number of core running inside the task.

**running\_instance\_count****Type** `int`**Getter** Returns this task's running instance count

Number of instances running inside the task.

**advanced\_range****Type** `str`**Getter** Returns this task's advanced range**Setter** Sets this task's advanced range

Advanced instances range selection.

Allows to select which instances will be computed. Should be None or match the following extended regular expression “`(([0-9]+|[0-9]+-[0-9]+)(,[0-9]+|[0-9]+-[0-9]+))+`” eg: 1,3-8,9,12-19Can be set until `run()` is called.**Raises** **AttributeError** – if `instancecount` is not 0 when setting this property**Warning:** This property is mutually exclusive with `instancecount`**snapshot\_whitelist****Type** `str`**Getter** Returns this task's snapshot whitelist**Setter** Sets this task's snapshot whitelistSnapshot white list (regex) for `snapshot()` and `instant()`

Can be set until task is submitted.

**snapshot\_blacklist**

**Type** `str`

**Getter** Returns this task's snapshot blacklist

**Setter** Sets this task's snapshot blacklist

Snapshot black list (regex) for `snapshot()` `instant()`

Can be set until task is submitted.

#### **results\_whitelist**

**Type** `str`

**Getter** Returns this task's results whitelist

**Setter** Sets this task's results whitelist

Results whitelist (regex)

Can be set until task is submitted.

#### **results\_blacklist**

**Type** `str`

**Getter** Returns this task's results blacklist

**Setter** Sets this task's results blacklist

Results blacklist (regex)

Can be set until task is submitted.

#### **status**

**Type** `garnot.status.Status`

**Getter** Returns this task's status

Status of the task

#### **completed\_instances**

**Type** `list(CompletedInstance)`

**Getter** Return this task's completed instances

#### **creation\_date**

**Type** `str`

**Getter** Returns this task's creation date

Creation date of the task (UTC Time)

#### **errors**

**Type** `list(Error)`

**Getter** Returns this task's errors if any.

Error reason if any, empty string if none

#### **constants**

**Type** `dictionary{str: str}`

**Getter** Returns this task's constants dictionary.

**Setter** set the task's constants dictionary.



Update the constants if needed Constants are the parametrizer of the profiles. Use them to adjust your profile parameter.

**constraints**

**Type** `dictionary{str: str}`

**Getter** Returns this task's constraints dictionary.

**Setter** set the task's constraints dictionary.

Update the constraints if needed advance usage

**wait\_for\_pool\_resources\_synchronization**

**Type** `bool` or `None`

**Getter** Returns this task's wait\_for\_pool\_resources\_synchronization.

**Setter** set the task's wait\_for\_pool\_resources\_synchronization.

**Raises** `qarnot.exceptions.AttributeError` – can't set this attribute on a launched task

**auto\_update**

**Type** `bool`

**Getter** Returns this task's auto update state

**Setter** Sets this task's auto update state

Auto update state, default to True When auto update is disabled properties will always return cached value for the object and a call to `update()` will be required to get latest values from the REST Api.

**update\_cache\_time**

**Type** `int`

**Getter** Returns this task's auto update state

**Setter** Sets this task's auto update state

Cache expiration time, default to 5s

**set\_task\_dependencies\_from\_uuids** (*uuids*)

Setter for the task dependencies using uuid

**set\_task\_dependencies\_from\_tasks** (*tasks*)

Setter for the task dependencies using tasks

**auto\_delete**

Autodelete this Task if it is finished and your max number of task is reach

Can be set until `run()` or `submit()` is called.

**Type** `bool`

**Getter** Returns is this task must autodelete

**Setter** Sets this task's autodelete

**Default\_value** "False"

**Raises** `AttributeError` – if you try to reset the auto\_delete after the task is submit

#### `completion_ttl`

The task will be auto delete *completion\_ttl* after it is finished

Can be set until *run()* or *submit()* is called.

**Getter** Returns this task's completed time to live.

**Type** `str`

**Setter** Sets this task's this task's completed time to live.

**Type** `str` or `datetime.timedelta`

**Default\_value** `""`

**Raises** `AttributeError` – if you try to set it after the task is submitted

The *completion\_ttl* must be a timedelta or a time span format string (example: 'd.hh:mm:ss' or 'hh:mm:ss')

`__repr__()`  $\Leftrightarrow$  `repr(x)`

**class** `qarnot.task.CompletedInstance(json)`

Bases: `object`

Completed Instance Information

---

**Note:** Read-only class

---

`__init__(json)`

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

**instance\_id** = `None`

**Type** `int`

Instance number.

**state** = `None`

**Type** `str`

Instance final state.

**wall\_time\_sec** = `None`

**Type** `float`

Instance wall time in seconds.

**exec\_time\_sec** = `None`

**Type** `float`

Execution time in seconds.

**exec\_time\_sec\_ghz** = `None`

**Type** `float`

Execution time in seconds GHz.

**peak\_memory\_mb** = `None`

**Type** `int`

Peak memory size in MB.

**average\_ghz** = `None`

**Type** `float`

Instance execution time GHz

**results** = `None`

**Type** `:class:list(str)`

Instance produced results

**\_\_repr\_\_** ()  $\leq \Rightarrow repr(x)$

**class** `qarnot.task.BulkTaskResponse(json)`

Bases: `object`

Bulk Task Response Information

---

**Note:** Read-only class

---

**\_\_init\_\_** (`json`)

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

**status\_code** = `None`

**Type** `int`

Status code.

**uuid** = `None`

**Type** `str`

Created Task Uuid.

**message** = `None`

**Type** `str`

User friendly error message.

**is\_success** ()

Check that the task submit has been successful.

**Return type** `bool`

**Returns** The task creation success(depending on received uuid and the status code).

**\_\_repr\_\_** ()  $\leq \Rightarrow repr(x)$

## Job

Module to handle a job.

**class** `qarnot.job.Job(connection, name, pool=None, shortname=None, use_dependencies=False)`

Bases: `object`

Represents a Qarnot job.

---

**Note:** A *Job* must be created with `qarnot.connection.Connection.create_job()` or retrieved with `qarnot.connection.Connection.jobs()` or `qarnot.connection.Connection.retrieve_job()`.

---

`__init__(connection, name, pool=None, shortname=None, use_dependencies=False)`

Create a new *Job*.

#### Parameters

- **connection** (*qarnot.connection.Connection*) – the cluster on which to send the job
- **name** (*str*) – given name of the job
- **pool** (*Pool* or *None*) – which Pool to submit the job in,
- **shortname** (*str*) – userfriendly job name
- **use\_dependencies** – allow dependencies between tasks in this job

#### `auto_update`

**Type** *bool*

**Getter** Returns this job's auto update state

**Setter** Sets this job's auto update state

Auto update state, default to True When auto update is disabled properties will always return cached value for the object and a call to *update()* will be required to get latest values from the REST Api.

#### `update_cache_time`

**Type** *int*

**Getter** Returns this job's auto update state

**Setter** Sets this job's auto update state

Cache expiration time, default to 5s

#### `state`

**Type** *str*

**Getter** return this job's state

State of the job.

#### Value is in

- UnSubmitted
- Active,
- Terminating,
- Completed,
- Deleting

**Warning:** this is the state of the job when the object was retrieved, call *update()* for up to date value.

#### `tasks`

**Type** List of *Task*

**Getter** Returns this job tasks

The tasks submitted in this job.

#### **use\_dependencies**

**Type** `bool`

**Getter** task's job can have dependencies

**Setter** Set if there is task's job dependencies

Can be set until `submit()` is called.

#### **uuid**

**Type** `str`

**Getter** Returns this job's uuid

The job's uuid.

Automatically set when a job is submitted.

#### **name**

**Type** `str`

**Getter** Returns this job's name

**Setter** Sets this job's name

The job's name.

Can be set until job is submitted.

#### **shortname**

**Type** `str`

**Getter** Returns this job's shortname

**Setter** Sets this job's shortname

The job's shortname, must be DNS compliant and unique, if not provided, will default to `uuid`.

Can be set until job is submitted.

#### **creation\_date**

**Type** `str`

**Getter** Returns this job's creation date

Creation date of the job (UTC Time)

#### **max\_wall\_time**

**Type** `str`

**Getter** Returns this job's maximum wall time

**Setter** Sets this job's maximum wall time

The job's maximum wall time. It is a time span string. Format example: 'd.hh:mm:ss' or 'hh:mm:ss'

Can be set until job is submitted.

#### **pool**

**Type** `Pool`

**Getter** Returns this job's pool

**Setter** Sets this job's pool

The pool to run the job in.

Can be set until `submit()` is called.

**classmethod** `from_json(connection, payload)`

Create a Job object from a json job.

**Parameters**

- **connection** (`qarnot.connection.Connection`) – the cluster connection
- **json\_job** (`dict`) – Dictionary representing the job

**Returns** The created `Job`.

**submit()**

Submit job to the cluster if it is not already submitted.

**Raises**

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.MaxJobException` – Job quota reached
- `qarnot.exceptions.NotEnoughCreditsException` – Not enough credits
- `qarnot.exceptions.UnauthorizedException` – invalid credentials

**update** (`flushcache=False`)

Update the job object from the REST Api. The flushcache parameter can be used to force the update, otherwise a cached version of the object will be served when accessing properties of the object. Cache behavior is configurable with `auto_update` and `update_cache_time`.

**Raises**

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.MissingJobException` – job does not exist

**terminate()**

Terminate this job on the server and abort all remaining tasks in the job.

**Raises**

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.MissingJobException` – job does not exist

**delete** (`forceAbort=False`)

Delete this job on the server.

The forceAbort parameter can be used to force running task in the job to be aborted,

**Raises**

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials

- `qarnot.exceptions.UnauthorizedException` – job still contains running tasks
- `qarnot.exceptions.MissingJobException` – job does not exist

**auto\_delete**

Autodelete this job if it is finished and your max number of job is reach

Can be set until `submit()` is called.

**Type** `bool`

**Getter** Returns is this job must autodelete

**Setter** Sets this job's autodelete

**Default\_value** "False"

**Raises** `AttributeError` – if you try to reset the auto\_delete after the job is submit

**completion\_ttl**

The job will be auto delete `completion_ttl` after it is finished

Can be set until `submit()` is called.

**Getter** Returns this job's completed time to live.

**Type** `str`

**Setter** Sets this job's this job's completed time to live.

**Type** `str` or `datetime.timedelta`

**Default\_value** ""

**Raises** `AttributeError` – if you try to set it after the job is submitted

The `completion_ttl` must be a timedelta or a time span format string (example: 'd.hh:mm:ss' or 'hh:mm:ss')

`__repr__()`  $\Leftrightarrow$  `repr(x)`

## Pool

Module to handle a pool.

**class** `qarnot.pool.Pool` (`connection`, `name`, `profile`, `instancecount=1`, `shortname=None`)

Bases: `object`

Represents a Qarnot pool.

---

**Note:** A `Pool` must be created with `qarnot.connection.Connection.create_pool()` or retrieved with `qarnot.connection.Connection.pools()` or `qarnot.connection.Connection.retrieve_pool()`.

---

`__init__` (`connection`, `name`, `profile`, `instancecount=1`, `shortname=None`)

Create a new `Pool`.

**Parameters**

- **connection** (`qarnot.connection.Connection`) – the cluster on which to send the pool
- **name** (`str`) – given name of the pool

- **profile** (*str*) – which profile to use with this task
- **instancecount** (*int or str*) – number of instances or ranges on which to run pool
- **shortname** (*str*) – userfriendly pool name

**classmethod from\_json** (*connection, json\_pool, is\_summary=False*)

Create a Pool object from a json pool.

#### Parameters

- **connection** (*qarnot.connection.Connection*) – the cluster connection
- **json\_pool** (*dict*) – Dictionary representing the pool

**Returns** The created *Pool*.

**submit** ()

Submit pool to the cluster if it is not already submitted.

#### Raises

- *qarnot.exceptions.QarnotGenericException* – API general error, see message for details
- *qarnot.exceptions.MaxPoolException* – Pool quota reached
- *qarnot.exceptions.NotEnoughCreditsException* – Not enough credits
- *qarnot.exceptions.UnauthorizedException* – invalid credentials

---

**Note:** Will ensure all added files are on the resource bucket regardless of their uploading mode.

---

**update** (*flushcache=False*)

Update the pool object from the REST Api. The flushcache parameter can be used to force the update, otherwise a cached version of the object will be served when accessing properties of the object. Cache behavior is configurable with *auto\_update* and *update\_cache\_time*.

#### Raises

- *qarnot.exceptions.QarnotGenericException* – API general error, see message for details
- *qarnot.exceptions.UnauthorizedException* – invalid credentials
- *qarnot.exceptions.MissingTaskException* – pool does not represent a valid one

**commit** ()

Replicate local changes on the current object instance to the REST API

#### Raises

- *qarnot.exceptions.QarnotGenericException* – API general error, see message for details
- *qarnot.exceptions.UnauthorizedException* – invalid credentials

This function need to be call to apply the local elastic pool setting modifications. .. note:: When updating buckets' properties, auto update will be disabled until commit is called.

**setup\_elastic** (*minimum\_total\_slots=0, maximum\_total\_slots=1, minimum\_idle\_slots=0, minimum\_idle\_time\_seconds=0, resize\_factor=1, resize\_period=90*)

Setup the pool elastic properties



**Parameters**

- **minimum\_total\_slots** (*int*) – Minimum slot number for the pool in elastic mode. Defaults to 0.
- **maximum\_total\_slots** (*int*) – Maximum slot number for the pool in elastic mode. Defaults to 1.
- **minimum\_idle\_slots** (*int*) – Minimum idling slot number. Defaults to 0.
- **minimum\_idle\_time\_seconds** (*int*) – Wait time in seconds before closing an unused slot if the number of unused slots are upper than the `minimum_idle_slots`. Defaults to 0.
- **resize\_factor** (*float*) – Growing factor of the pool. It must be a number between 0 and 1. Defaults to 1.
- **resize\_period** (*int*) – Refresh rate of resizing the pool in elastic mode. Defaults to 90.

**delete** (*purge\_resources=False*)

Delete this pool on the server.

**Parameters** **purge\_resources** (*bool*) – parameter value is used to determine if the bucket is also deleted. Defaults to False.

**Raises**

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.MissingTaskException` – pool does not exist

**close** ()

Close this pool if running.

**Raises**

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials
- `qarnot.exceptions.MissingPoolException` – pool does not exist

**uuid**

**Type** `str`

**Getter** Returns this pool's uuid

The pool's uuid.

Automatically set when a pool is submitted.

**state**

**Type** `str`

**Getter** return this pool's state

State of the pool.

**Value is in**

- UnSubmitted

- Submitted
- PartiallyDispatched
- FullyDispatched
- PartiallyExecuting
- FullyExecuting
- Closing
- Closed
- Failure
- PendingDelete

**Warning:** this is the state of the pool when the object was retrieved, call `update()` for up to date value.

#### resources

**Type** `list(Bucket)`

**Getter** Returns this pool's resources bucket

**Setter** Sets this pool's resources bucket

Represents resource files.

#### name

**Type** `str`

**Getter** Returns this pool's name

**Setter** Sets this pool's name

The pool's name.

Can be set until pool is submitted.

#### shortname

**Type** `str`

**Getter** Returns this pool's shortname

**Setter** Sets this pool's shortname

The pool's shortname, must be DNS compliant and unique, if not provided, will default to `uuid`.

Can be set until pool is submitted.

#### tags

**Type** `:class:list(str)`

**Getter** Returns this pool's tags

**Setter** Sets this pool's tags

Custom tags.

#### profile

**Type** `str`

**Getter** Returns this pool's profile

**Setter** Sets this pool's profile

The profile to run the pool with.

Can be set until `submit()` is called.

#### **instancecount**

**Type** `int`

**Getter** Returns this pool's instance count

**Setter** Sets this pool's instance count

Number of instances needed for the pool.

Can be set until `submit()` is called.

#### **running\_core\_count**

**Type** `int`

**Getter** Returns this pool's running core count

Number of core running inside the pool.

#### **running\_instance\_count**

**Type** `int`

**Getter** Returns this pool's running instance count

Number of instances running inside the pool.

#### **errors**

**Type** `list(str)`

**Getter** Returns this pool's error list

Error reason if any, empty string if none

#### **creation\_date**

**Type** `str`

**Getter** Returns this pool's creation date

Creation date of the pool (UTC Time)

#### **status**

**Type** `qarnot.status.Status`

**Getter** Returns this pool's status

Status of the task

#### **auto\_update**

**Type** `bool`

**Getter** Returns this pool's auto update state

**Setter** Sets this pool's auto update state

Auto update state, default to True When auto update is disabled properties will always return cached value for the object and a call to `update()` will be required to get latest values from the REST Api.

#### **update\_cache\_time**

**Type** `int`

**Getter** Returns this pool's auto update state

**Setter** Sets this pool's auto update state

Cache expiration time, default to 5s

#### **is\_elastic**

**Type** `bool`

**Getter** Returns this pool's is\_elastic

**Setter** Sets this pool's is\_elastic

Define if you use a static or an elastic pool.

#### **elastic\_minimum\_slots**

**Type** `int`

**Getter** Returns this pool's elastic\_minimum\_slots

**Setter** Sets this pool's elastic\_minimum\_slots

The minimum slot number of the elastic pool. Define the minimum number of pool instances stay open during the pool execution.

#### **elastic\_maximum\_slots**

**Type** `int`

**Getter** Returns this pool's elastic\_maximum\_slots

**Setter** Sets this pool's elastic\_maximum\_slots

The maximum slot number of the elastic pool. Define the maximum number of pool instances opened during the pool execution.

#### **elastic\_minimum\_idle\_slots**

**Type** `int`

**Getter** Returns this pool's elastic\_minimum\_idle\_slots

**Setter** Sets this pool's elastic\_minimum\_idle\_slots

The minimum idle number of the elastic pool. Define the minimum number of the idle pool instances stay opened during the pool execution. It should be lower to elastic\_minimum\_slots to be usefull

#### **elastic\_minimum\_idle\_time**

**Type** `int`

**Getter** Returns this pool's elastic\_minimum\_idle\_time

**Setter** Sets this pool's elastic\_minimum\_idle\_time

Wait time in seconds before closing an unused slot if the number of unused slots are upper than the minimum\_idle\_slots.

#### **elastic\_resize\_factor**

**Type** `float`

**Getter** Returns this pool's elastic\_resize\_factor

**Setter** Sets this pool's `elastic_resize_factor`

The resize factor of the pool. It represent the resize factor of the slots. It's a decimal number upper than 0 and and equal or lower the 1

#### **`elastic_resize_period`**

**Type** `int`

**Getter** Returns this pool's `elastic_resize_period`

**Setter** Sets this pool's `elastic_resize_period`

The resize period of the elastic pool in second. This is the refresh rate of resizing the elastic pool.

#### **`preparation_command_line`**

**Type** `str`:

**Getter** Returns this pool's command line.

**Setter** set the pool's command line.

Update the pool command line if needed The command line is a command running before each task execution.

#### **`constants`**

**Type** `dictionary{str: str}`

**Getter** Returns this pool's constants dictionary.

**Setter** set the pool's constants dictionary.

Update the constants if needed Constants are the parametrizer of the profiles. Use them to adjust your profile parameter.

#### **`constraints`**

**Type** `dictionary{str: str}`

**Getter** Returns this pool's constraints dictionary.

**Setter** set the pool's constraints dictionary.

Update the constraints if needed advance usage

#### **`tasks_default_wait_for_pool_resources_synchronization`**

**Type** `bool`

**Getter** Returns this task's `tasks_default_wait_for_pool_resources_synchronization`.

**Setter** set the task's `tasks_default_wait_for_pool_resources_synchronization`.

**Raises** `qarnot.exceptions.AttributeError` – can't set this attribute on a launched task

#### **`auto_delete`**

Autodelete this pool if it is finished and your max number of pool is reach

Can be set until `submit()` is called.

**Type** `bool`

**Getter** Returns is this pool must autodelete

**Setter** Sets this pool's autodelete

**Default\_value** "False"

Raises **AttributeError** – if you try to reset the `auto_delete` after the pool is submit

**completion\_ttl**

The pool will be auto delete *completion\_ttl* after it is finished

Can be set until *submit()* is called.

**Getter** Returns this pool’s completed time to live.

**Type** `str`

**Setter** Sets this pool’s this pool’s completed time to live.

**Type** `str` or `datetime.timedelta`

**Default\_value** `“”`

Raises **AttributeError** – if you try to set it after the pool is submitted

The *completion\_ttl* must be a `timedelta` or a time span format string (example: `‘d.hh:mm:ss’` or `‘hh:mm:ss’`)

`__repr__()`  $\Leftrightarrow$  *repr(x)*

## Status

**class** `garnot.status.Status(json)`

Bases: `object`

The status object of the running pools and tasks. To retrieve the status of a pool, use:

- `my_pool.status`

To retrieve the status of a task, use:

- `my_task.status`

---

**Note:** Read-only class

---

`__init__(json)`

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

**download\_progress = None**

**Type** `float`

Resources download progress to the instances.

**execution\_progress = None**

**Type** `float`

Task execution progress.

**upload\_progress = None**

**Type** `float`

Task results upload progress to the API.

**instance\_count = None**

**Type** `int`

Number of running instances.

**download\_time = None**

Type `str`

Resources download time to the instances.

**download\_time\_sec = None**

Type `float`

Resources download time to the instances in seconds.

**environment\_time = None**

Type `str`

Environment time to the instances.

**environment\_time\_sec = None**

Type `float`

Environment time to the instances in seconds.

**execution\_time = None**

Type `str`

Task execution time.

**execution\_time\_sec = None**

Type `float`

Task execution time in seconds.

**upload\_time = None**

Type `str`

Task results upload time to the API.

**upload\_time\_sec = None**

Type `float`

Task results upload time to the API in seconds.

**wall\_time = None**

Type `str`

Wall time of the task.

**wall\_time\_sec = None**

Type `float`

Wall time of the task in seconds.

**succeeded\_range = None**

Type `str`

Successful instances range.

**executed\_range = None**

Type `str`

Executed instances range.

**failed\_range** = None

Type `str`

Failed instances range.

**running\_instances\_info** = None

Type `RunningInstancesInfo`

Running instances information.

**\_\_repr\_\_** () <==> *repr*(x)

**class** garnot.status.**RunningInstancesInfo** (*json*)

Bases: `object`

Running Instances Information

---

**Note:** Read-only class

---

**\_\_init\_\_** (*json*)

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature

**per\_running\_instance\_info** = None

Type list(`PerRunningInstanceInfo`)

Per running instances information.

**timestamp** = None

Type `str`

Last information update timestamp.

**average\_frequency\_ghz** = None

Type `float`

Average Frequency in GHz.

**max\_frequency\_ghz** = None

Type `float`

Maximum Frequency in GHz.

**min\_frequency\_ghz** = None

Type `float`

Minimum Frequency in GHz.

**average\_max\_frequency\_ghz** = None

Type `float`

Average Maximum Frequency in GHz.

**average\_cpu\_usage** = None

Type `float`

Average CPU Usage.

**cluster\_power\_indicator** = None



Type `float`

Cluster Power Indicator.

**average\_memory\_usage = None**

Type `float`

Average Memory Usage.

**average\_network\_in\_kbps = None**

Type `float`

Average Network Input in Kbps.

**average\_network\_out\_kbps = None**

Type `float`

Average Network Output in Kbps.

**total\_network\_in\_kbps = None**

Type `float`

Total Network Input in Kbps.

**total\_network\_out\_kbps = None**

Type `float`

Total Network Output in Kbps.

**\_\_repr\_\_** ()  $\leq ==>$  *repr*(x)

**class** qarnot.status.**PerRunningInstanceInfo** (*json*)

Bases: `object`

Per Running Instance Information

---

**Note:** Read-only class

---

**\_\_init\_\_** (*json*)

x.**\_\_init\_\_**(...) initializes x; see help(type(x)) for signature

**phase = None**

Type `str`

Instance phase.

**instance\_id = None**

Type `int`

Instance number.

**max\_frequency\_ghz = None**

Type `float`

Maximum CPU frequency in GHz.

**current\_frequency\_ghz = None**

Type `float`

Current CPU frequency in GHz.

**cpu\_usage = None**

Type `float`

Current CPU usage.

**max\_memory\_mb = None**

Type `int`

Maximum memory size in MB.

**current\_memory\_mb = None**

Type `int`

Current memory size in MB.

**memory\_usage = None**

Type `float`

Current memory usage.

**network\_in\_kbps = None**

Type `float`

Network Input in Kbps.

**network\_out\_kbps = None**

Type `float`

Network Output in Kbps.

**progress = None**

Type `float`

Instance progress.

**execution\_time\_sec = None**

Type `float`

Instance execution time in seconds.

**execution\_time\_ghz = None**

Type `float`

Instance execution time GHz

**cpu\_model = None**

Type `str`

CPU model

**active\_forward = None**

type: list(*TaskActiveForward*)

Active forwards list.

**vpn\_connections = None**

type: list(*TaskVpnConnection*)

Vpn connection list.

`__repr__()`  $\leq \Rightarrow$  *repr(x)*

**class** garnot.status.TaskActiveForward(*json*)

Bases: `object`

Task Active Forward

---

**Note:** Read-only class

---

`__init__`(*json*)

*x*.`__init__`(...) initializes *x*; see `help(type(x))` for signature

**application\_port** = None

Type `int`

Application Port.

**forwarder\_port** = None

Type `int`

Forwarder Port.

**forwarder\_host** = None

Type `str`

Forwarder Host.

**bind\_address** = None

Type `str`

Bind address of the listening socket on the forwarder host.

`__repr__()`  $\leq \Rightarrow$  *repr(x)*

**class** garnot.status.TaskVpnConnection(*json*)

Bases: `object`

Vpn Connection Information

---

**Note:** Read-only class

---

`__init__`(*json*)

*x*.`__init__`(...) initializes *x*; see `help(type(x))` for signature

**node\_ip\_address\_cidr** = None

Type `str`

Vpn classless inter-domain routing address.

**vpn\_name** = None

Type `str`

Vpn name.

`__repr__()`  $\leq \Rightarrow$  *repr(x)*

### 1.3.3 Storage

#### Storage

Storage prototype

**class** `garnot.storage.Storage`

Bases: `object`

Common architecture for storage providers

**get\_all\_files** (*output\_dir*, *progress=None*)

Get all files from the storage.

#### Parameters

- **output\_dir** (*str*) – local directory for the retrieved files.
- **progress** (*bool* or *function(float, float, str)*) – can be a callback (read,total,filename) or True to display a progress bar

#### Raises

- `garnot.exceptions.MissingBucketException` – the bucket is not on the server
- `garnot.exceptions.QarnotGenericException` – API general error, see message for details
- `garnot.exceptions.UnauthorizedException` – invalid credentials

**Warning:** Will override *output\_dir* content.

**\_\_init\_\_** ()

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature

**list\_files** ()

List files on the storage.

... **note:** File object returned *must* have a *key* property.

**Returns** List of the files on the storage.

**get\_file** (*remote*, *local=None*, *progress=None*)

Get a file from the storage. Create needed subfolders.

#### Parameters

- **remote** (*str*) – the name of the remote file
- **local** (*str*) – local name of the retrieved file (defaults to *remote*)
- **progress** (*bool* or *function(float, float, str)*) – can be a callback (read,total,filename) or True to display a progress bar

**Return type** `str`

**Returns** The name of the output file.

**Raises** `ValueError` – no such file

**copy\_file** (*source*, *dest*)

Create a copy of a file

**Parameters**

- **source** (*str*) – name of the existing file to duplicate
- **dest** (*str*) – name of the created file

**add\_directory** (*local*, *remote*)

Add a directory to the storage. Does not follow symlinks. File hierarchy is preserved.

**Parameters**

- **local** (*str*) – path of the local directory to add
- **remote** (*str*) – path of the directory on remote node (defaults to *local*)

**Raises** **IOError** – not a valid directory

**add\_file** (*local\_or\_file*, *remote*)

Add a local file or a Python File on the storage.

---

**Note:** You can also use **object[remote] = local**

---

**Parameters**

- **local\_or\_file** (*str* or *File*) – path of the local file or an opened Python File
- **remote** (*str*) – name of the remote file (defaults to *local\_or\_file*)

**delete\_file** (*remote*)

Delete a file from the storage.

**Parameters** **remote** (*str*) – the name of the remote file

**update** (*flush=None*)

Update object from remote endpoint

**Parameters** **flush** (*bool*) – bypass cache

**flush** ()

Ensure all background uploads are complete

**\_\_getitem\_\_** (*filename*)

`x.__getitem__(y) <==> x[y]`

**\_\_setitem\_\_** (*remote*, *filename*)

`x.__setitem__(i, y) <==> x[i]=y`

**\_\_delitem\_\_** (*filename*)

`x.__delitem__(y) <==> del x[y]`

**\_\_contains\_\_** (*k*) → True if D has a key k, else False

**\_\_iter\_\_** () <==> *iter(x)*

**\_\_eq\_\_** (*other*)

`x.__eq__(y) <==> x == y`

**\_\_ne\_\_** (*other*)

`x.__ne__(y) <==> x != y`

## Bucket

Module for bucket object.

**class** `garnot.bucket.Bucket` (*connection, name, create=True*)

Bases: `garnot.storage.Storage`

Represents a resource/result bucket.

This class is the interface to manage resources or results from a `garnot.bucket.Bucket`.

---

**Note:** A `Bucket` must be created with `garnot.connection.Connection.create_bucket()` or retrieved with `garnot.connection.Connection.buckets()`, `garnot.connection.Connection.retrieve_bucket()`, or `garnot.connection.Connection.retrieve_or_create_bucket()`.

---

---

**Note:** Paths given as ‘remote’ arguments, (or as path arguments for `Bucket.directory()`) **must** be valid unix-like paths.

---

`__init__` (*connection, name, create=True*)

`x.__init__(...)` initializes x; see `help(type(x))` for signature

**delete** ()

Delete the bucket represented by this `Bucket`.

**list\_files** ()

List files in the bucket

**Return type** `list(S3.ObjectSummary)`

**Returns** A list of `ObjectSummary` resources

**directory** (*directory=""*)

List files in a directory of the bucket according to prefix.

**Return type** `list(S3.ObjectSummary)`

**Returns** A list of `ObjectSummary` resources

**sync\_directory** (*directory, verbose=False, remote=None*)

Synchronize a local directory with the remote buckets.

**Parameters**

- **directory** (*str*) – The local directory to use for synchronization
- **verbose** (*bool*) – Print information about synchronization operations
- **remote** (*str*) – path of the directory on remote node (defaults to *local*)

**Warning:** Local changes are reflected on the server, a file present on the bucket but not in the local directory will be deleted from the bucket.

A file present in the directory but not in the bucket will be uploaded.

---

**Note:** The following parameters are used to determine whether synchronization is required :

- name
  - size
  - sha1sum
- 

**sync\_files** (*files*, *verbose=False*, *remote=None*)

Synchronize files with the remote buckets.

#### Parameters

- **files** (*dict*) – Dictionary of synchronized files
- **verbose** (*bool*) – Print information about synchronization operations
- **remote** (*str*) – path of the directory on remote node (defaults to *local*)

Dictionary key is the remote file path while value is the local file path.

**Warning:** Local changes are reflected on the server, a file present on the bucket but not in the local directory will be deleted from the bucket.

A file present in the directory but not in the bucket will be uploaded.

---

**Note:** The following parameters are used to determine whether synchronization is required :

- name
  - size
  - sha1sum
- 

**add\_file** (*local\_or\_file*, *remote=None*)

Add a local file or a Python File on the storage.

---

**Note:** You can also use **object[remote] = local**

---

#### Parameters

- **local\_or\_file** (*str or File*) – path of the local file or an opened Python File
- **remote** (*str*) – name of the remote file (defaults to *local\_or\_file*)

**get\_all\_files** (*output\_dir*, *progress=None*)

Get all files from the storage.

#### Parameters

- **output\_dir** (*str*) – local directory for the retrieved files.
- **progress** (*bool or function(float, float, str)*) – can be a callback (read,total,filename) or True to display a progress bar

#### Raises

- **qarnot.exceptions.MissingBucketException** – the bucket is not on the server

- `qarnot.exceptions.QarnotGenericException` – API general error, see message for details
- `qarnot.exceptions.UnauthorizedException` – invalid credentials

**Warning:** Will override `output_dir` content.

**get\_file** (*remote*, *local=None*, *progress=None*)

Get a file from the storage. Create needed subfolders.

**Parameters**

- **remote** (*str*) – the name of the remote file
- **local** (*str*) – local name of the retrieved file (defaults to *remote*)
- **progress** (*bool* or *function(float, float, str)*) – can be a callback (read,total,filename) or True to display a progress bar

**Return type** *str*

**Returns** The name of the output file.

**Raises** **ValueError** – no such file

**add\_directory** (*local*, *remote=""*)

Add a directory to the storage. Does not follow symlinks. File hierarchy is preserved.

**Parameters**

- **local** (*str*) – path of the local directory to add
- **remote** (*str*) – path of the directory on remote node (defaults to *local*)

**Raises** **IOError** – not a valid directory

**copy\_file** (*source*, *dest*)

Create a copy of a file

**Parameters**

- **source** (*str*) – name of the existing file to duplicate
- **dest** (*str*) – name of the created file

**flush** ()

Ensure all background uploads are complete

**update** (*flush=False*)

Update object from remote endpoint

**Parameters** **flush** (*bool*) – bypass cache

**delete\_file** (*remote*)

Delete a file from the storage.

**Parameters** **remote** (*str*) – the name of the remote file

**uuid**

Bucket identifier

**description**

Bucket identifier



### 1.3.4 Exceptions

Exceptions.

**exception** `qarnot.exceptions.QarnotException`

Bases: `exceptions.Exception`

Qarnot Exception

**exception** `qarnot.exceptions.QarnotGenericException(msg)`

Bases: `qarnot.exceptions.QarnotException`

General Connection exception

**exception** `qarnot.exceptions.BucketStorageUnavailableException`

Bases: `qarnot.exceptions.QarnotException`

API bucket storage is disabled.

**exception** `qarnot.exceptions.UnauthorizedException`

Bases: `qarnot.exceptions.QarnotException`

Invalid token.

**exception** `qarnot.exceptions.MissingTaskException`

Bases: `qarnot.exceptions.QarnotException`

Non existent task.

**exception** `qarnot.exceptions.MissingBucketException`

Bases: `qarnot.exceptions.QarnotException`

Non existent bucket.

**exception** `qarnot.exceptions.MissingPoolException`

Bases: `qarnot.exceptions.QarnotException`

Non existent pool.

**exception** `qarnot.exceptions.MaxTaskException`

Bases: `qarnot.exceptions.QarnotException`

Max number of tasks reached.

**exception** `qarnot.exceptions.MaxPoolException`

Bases: `qarnot.exceptions.QarnotException`

Max number of pools reached.

**exception** `qarnot.exceptions.NotEnoughCreditsException`

Bases: `qarnot.exceptions.QarnotException`

Not enough credits exception.

**exception** `qarnot.exceptions.MissingJobException`

Bases: `exceptions.Exception`

Non existent job.

**exception** `qarnot.exceptions.MaxJobException`

Bases: `exceptions.Exception`

Max number of jobs reached.



## CHAPTER 2

---

### Indices and tables

---

- `genindex`
- `modindex`



### q

- `qarnot.bucket`, [42](#)
- `qarnot.connection`, [4](#)
- `qarnot.exceptions`, [45](#)
- `qarnot.job`, [23](#)
- `qarnot.pool`, [27](#)
- `qarnot.status`, [34](#)
- `qarnot.storage`, [40](#)
- `qarnot.task`, [11](#)



## Symbols

`__contains__()` (*qarnot.storage.Storage* method), 41  
`__delitem__()` (*qarnot.storage.Storage* method), 41  
`__eq__()` (*qarnot.storage.Storage* method), 41  
`__getitem__()` (*qarnot.storage.Storage* method), 41  
`__init__()` (*qarnot.bucket.Bucket* method), 42  
`__init__()` (*qarnot.connection.Connection* method), 4  
`__init__()` (*qarnot.connection.Profile* method), 10  
`__init__()` (*qarnot.connection.UserInfo* method), 10  
`__init__()` (*qarnot.job.Job* method), 23  
`__init__()` (*qarnot.pool.Pool* method), 27  
`__init__()` (*qarnot.status.PerRunningInstanceInfo* method), 37  
`__init__()` (*qarnot.status.RunningInstancesInfo* method), 36  
`__init__()` (*qarnot.status.Status* method), 34  
`__init__()` (*qarnot.status.TaskActiveForward* method), 39  
`__init__()` (*qarnot.status.TaskVpnConnection* method), 39  
`__init__()` (*qarnot.storage.Storage* method), 40  
`__init__()` (*qarnot.task.BulkTaskResponse* method), 23  
`__init__()` (*qarnot.task.CompletedInstance* method), 22  
`__init__()` (*qarnot.task.Task* method), 11  
`__iter__()` (*qarnot.storage.Storage* method), 41  
`__ne__()` (*qarnot.storage.Storage* method), 41  
`__repr__()` (*qarnot.connection.Profile* method), 11  
`__repr__()` (*qarnot.job.Job* method), 27  
`__repr__()` (*qarnot.pool.Pool* method), 34  
`__repr__()` (*qarnot.status.PerRunningInstanceInfo* method), 38  
`__repr__()` (*qarnot.status.RunningInstancesInfo* method), 37  
`__repr__()` (*qarnot.status.Status* method), 36  
`__repr__()` (*qarnot.status.TaskActiveForward*

*method*), 39  
`__repr__()` (*qarnot.status.TaskVpnConnection* method), 39  
`__repr__()` (*qarnot.task.BulkTaskResponse* method), 23  
`__repr__()` (*qarnot.task.CompletedInstance* method), 23  
`__repr__()` (*qarnot.task.Task* method), 22  
`__setitem__()` (*qarnot.storage.Storage* method), 41

## A

`abort()` (*qarnot.task.Task* method), 13  
`active_forward` (*qarnot.status.PerRunningInstanceInfo* attribute), 38  
`add_directory()` (*qarnot.bucket.Bucket* method), 44  
`add_directory()` (*qarnot.storage.Storage* method), 41  
`add_file()` (*qarnot.bucket.Bucket* method), 43  
`add_file()` (*qarnot.storage.Storage* method), 41  
`advanced_range` (*qarnot.task.Task* attribute), 19  
`application_port` (*qarnot.status.TaskActiveForward* attribute), 39  
`auto_delete` (*qarnot.job.Job* attribute), 27  
`auto_delete` (*qarnot.pool.Pool* attribute), 33  
`auto_delete` (*qarnot.task.Task* attribute), 21  
`auto_update` (*qarnot.job.Job* attribute), 24  
`auto_update` (*qarnot.pool.Pool* attribute), 31  
`auto_update` (*qarnot.task.Task* attribute), 21  
`average_cpu_usage` (*qarnot.status.RunningInstancesInfo* attribute), 36  
`average_frequency_ghz` (*qarnot.status.RunningInstancesInfo* attribute), 36  
`average_ghz` (*qarnot.task.CompletedInstance* attribute), 22  
`average_max_frequency_ghz` (*qarnot.status.RunningInstancesInfo* attribute), 36

average\_memory\_usage  
(*qarnot.status.RunningInstancesInfo* attribute),  
37

average\_network\_in\_kbps  
(*qarnot.status.RunningInstancesInfo* attribute),  
37

average\_network\_out\_kbps  
(*qarnot.status.RunningInstancesInfo* attribute),  
37

## B

bind\_address (*qarnot.status.TaskActiveForward* attribute), 39

Bucket (class in *qarnot.bucket*), 42

buckets() (*qarnot.connection.Connection* method), 5

BucketStorageUnavailableException, 45

BulkTaskResponse (class in *qarnot.task*), 23

## C

close() (*qarnot.pool.Pool* method), 29

cluster\_power\_indicator  
(*qarnot.status.RunningInstancesInfo* attribute),  
36

commit() (*qarnot.pool.Pool* method), 28

commit() (*qarnot.task.Task* method), 14

completed\_instances (*qarnot.task.Task* attribute),  
20

CompletedInstance (class in *qarnot.task*), 22

completion\_ttl (*qarnot.job.Job* attribute), 27

completion\_ttl (*qarnot.pool.Pool* attribute), 34

completion\_ttl (*qarnot.task.Task* attribute), 21

Connection (class in *qarnot.connection*), 4

constants (*qarnot.connection.Profile* attribute), 11

constants (*qarnot.pool.Pool* attribute), 33

constants (*qarnot.task.Task* attribute), 20

constraints (*qarnot.pool.Pool* attribute), 33

constraints (*qarnot.task.Task* attribute), 21

copy\_file() (*qarnot.bucket.Bucket* method), 44

copy\_file() (*qarnot.storage.Storage* method), 40

cpu\_model (*qarnot.status.PerRunningInstanceInfo* attribute), 38

cpu\_usage (*qarnot.status.PerRunningInstanceInfo* attribute), 38

create\_bucket() (*qarnot.connection.Connection* method), 9

create\_elastic\_pool()  
(*qarnot.connection.Connection* method),  
8

create\_job() (*qarnot.connection.Connection* method), 9

create\_pool() (*qarnot.connection.Connection* method), 7

create\_task() (*qarnot.connection.Connection* method), 8

creation\_date (*qarnot.job.Job* attribute), 25

creation\_date (*qarnot.pool.Pool* attribute), 31

creation\_date (*qarnot.task.Task* attribute), 20

current\_frequency\_ghz  
(*qarnot.status.PerRunningInstanceInfo* attribute), 37

current\_memory\_mb  
(*qarnot.status.PerRunningInstanceInfo* attribute), 38

## D

delete() (*qarnot.bucket.Bucket* method), 42

delete() (*qarnot.job.Job* method), 26

delete() (*qarnot.pool.Pool* method), 29

delete() (*qarnot.task.Task* method), 13

delete\_file() (*qarnot.bucket.Bucket* method), 44

delete\_file() (*qarnot.storage.Storage* method), 41

description (*qarnot.bucket.Bucket* attribute), 44

directory() (*qarnot.bucket.Bucket* method), 42

download\_progress (*qarnot.status.Status* attribute),  
34

download\_results() (*qarnot.task.Task* method), 16

download\_time (*qarnot.status.Status* attribute), 34

download\_time\_sec (*qarnot.status.Status* attribute),  
35

## E

elastic\_maximum\_slots (*qarnot.pool.Pool* attribute), 32

elastic\_minimum\_idle\_slots (*qarnot.pool.Pool* attribute), 32

elastic\_minimum\_idle\_time (*qarnot.pool.Pool* attribute), 32

elastic\_minimum\_slots (*qarnot.pool.Pool* attribute), 32

elastic\_resize\_factor (*qarnot.pool.Pool* attribute), 32

elastic\_resize\_period (*qarnot.pool.Pool* attribute), 33

email (*qarnot.connection.UserInfo* attribute), 10

environment\_time (*qarnot.status.Status* attribute),  
35

environment\_time\_sec (*qarnot.status.Status* attribute), 35

errors (*qarnot.pool.Pool* attribute), 31

errors (*qarnot.task.Task* attribute), 20

exec\_time\_sec (*qarnot.task.CompletedInstance* attribute), 22

exec\_time\_sec\_ghz  
(*qarnot.task.CompletedInstance* attribute),  
22

executed\_range (*qarnot.status.Status* attribute), 35

execution\_progress (*qarnot.status.Status* attribute), 34



execution\_time (*qarnot.status.Status* attribute), 35  
 execution\_time\_ghz  
   (*qarnot.status.PerRunningInstanceInfo* attribute), 38  
 execution\_time\_sec  
   (*qarnot.status.PerRunningInstanceInfo* attribute), 38  
 execution\_time\_sec (*qarnot.status.Status* attribute), 35

## F

failed\_range (*qarnot.status.Status* attribute), 35  
 flush() (*qarnot.bucket.Bucket* method), 44  
 flush() (*qarnot.storage.Storage* method), 41  
 forwarder\_host (*qarnot.status.TaskActiveForward* attribute), 39  
 forwarder\_port (*qarnot.status.TaskActiveForward* attribute), 39  
 fresh\_stderr() (*qarnot.task.Task* method), 17  
 fresh\_stdout() (*qarnot.task.Task* method), 17  
 from\_json() (*qarnot.job.Job* class method), 26  
 from\_json() (*qarnot.pool.Pool* class method), 28  
 from\_json() (*qarnot.task.Task* class method), 14

## G

get\_all\_files() (*qarnot.bucket.Bucket* method), 43  
 get\_all\_files() (*qarnot.storage.Storage* method), 40  
 get\_file() (*qarnot.bucket.Bucket* method), 44  
 get\_file() (*qarnot.storage.Storage* method), 40

## I

instance\_count (*qarnot.status.Status* attribute), 34  
 instance\_id (*qarnot.status.PerRunningInstanceInfo* attribute), 37  
 instance\_id (*qarnot.task.CompletedInstance* attribute), 22  
 instancecount (*qarnot.pool.Pool* attribute), 31  
 instancecount (*qarnot.task.Task* attribute), 18  
 instant() (*qarnot.task.Task* method), 15  
 is\_elastic (*qarnot.pool.Pool* attribute), 32  
 is\_success() (*qarnot.task.BulkTaskResponse* method), 23

## J

Job (class in *qarnot.job*), 23  
 jobs() (*qarnot.connection.Connection* method), 6

## L

list\_files() (*qarnot.bucket.Bucket* method), 42  
 list\_files() (*qarnot.storage.Storage* method), 40

## M

max\_bucket (*qarnot.connection.UserInfo* attribute), 10  
 max\_frequency\_ghz  
   (*qarnot.status.PerRunningInstanceInfo* attribute), 37  
 max\_frequency\_ghz  
   (*qarnot.status.RunningInstancesInfo* attribute), 36  
 max\_instances (*qarnot.connection.UserInfo* attribute), 10  
 max\_memory\_mb (*qarnot.status.PerRunningInstanceInfo* attribute), 38  
 max\_running\_task (*qarnot.connection.UserInfo* attribute), 10  
 max\_task (*qarnot.connection.UserInfo* attribute), 10  
 max\_wall\_time (*qarnot.job.Job* attribute), 25  
 MaxJobException, 45  
 MaxPoolException, 45  
 MaxTaskException, 45  
 memory\_usage (*qarnot.status.PerRunningInstanceInfo* attribute), 38  
 message (*qarnot.task.BulkTaskResponse* attribute), 23  
 min\_frequency\_ghz  
   (*qarnot.status.RunningInstancesInfo* attribute), 36  
 MissingBucketException, 45  
 MissingJobException, 45  
 MissingPoolException, 45  
 MissingTaskException, 45

## N

name (*qarnot.connection.Profile* attribute), 10  
 name (*qarnot.job.Job* attribute), 25  
 name (*qarnot.pool.Pool* attribute), 30  
 name (*qarnot.task.Task* attribute), 18  
 network\_in\_kbps (*qarnot.status.PerRunningInstanceInfo* attribute), 38  
 network\_out\_kbps (*qarnot.status.PerRunningInstanceInfo* attribute), 38  
 node\_ip\_address\_cidr  
   (*qarnot.status.TaskVpnConnection* attribute), 39  
 NotEnoughCreditsException, 45

## P

peak\_memory\_mb (*qarnot.task.CompletedInstance* attribute), 22  
 per\_running\_instance\_info  
   (*qarnot.status.RunningInstancesInfo* attribute), 36  
 PerRunningInstanceInfo (class in *qarnot.status*), 37

phase (*qarnot.status.PerRunningInstanceInfo* attribute), 37  
 Pool (class in *qarnot.pool*), 27  
 pool (*qarnot.job.Job* attribute), 25  
 pool (*qarnot.task.Task* attribute), 18  
 pools () (*qarnot.connection.Connection* method), 6  
 preparation\_command\_line (*qarnot.pool.Pool* attribute), 33  
 Profile (class in *qarnot.connection*), 10  
 profile (*qarnot.pool.Pool* attribute), 30  
 profile (*qarnot.task.Task* attribute), 18  
 profiles () (*qarnot.connection.Connection* method), 9  
 progress (*qarnot.status.PerRunningInstanceInfo* attribute), 38

## Q

qarnot.bucket (module), 42  
 qarnot.connection (module), 4  
 qarnot.exceptions (module), 45  
 qarnot.job (module), 23  
 qarnot.pool (module), 27  
 qarnot.status (module), 34  
 qarnot.storage (module), 40  
 qarnot.task (module), 11  
 QarnotException, 45  
 QarnotGenericException, 45  
 quota\_bytes\_bucket (*qarnot.connection.UserInfo* attribute), 10

## R

resources (*qarnot.pool.Pool* attribute), 30  
 resources (*qarnot.task.Task* attribute), 16  
 results (*qarnot.task.CompletedInstance* attribute), 23  
 results (*qarnot.task.Task* attribute), 16  
 results\_blacklist (*qarnot.task.Task* attribute), 20  
 results\_whitelist (*qarnot.task.Task* attribute), 20  
 resume () (*qarnot.task.Task* method), 12  
 retrieve\_bucket () (*qarnot.connection.Connection* method), 7  
 retrieve\_job () (*qarnot.connection.Connection* method), 7  
 retrieve\_or\_create\_bucket () (*qarnot.connection.Connection* method), 7  
 retrieve\_pool () (*qarnot.connection.Connection* method), 6  
 retrieve\_profile () (*qarnot.connection.Connection* method), 9  
 retrieve\_task () (*qarnot.connection.Connection* method), 7  
 run () (*qarnot.task.Task* method), 11

running\_core\_count (*qarnot.pool.Pool* attribute), 31  
 running\_core\_count (*qarnot.task.Task* attribute), 19  
 running\_instance\_count (*qarnot.pool.Pool* attribute), 31  
 running\_instance\_count (*qarnot.task.Task* attribute), 19  
 running\_instances\_info (*qarnot.status.Status* attribute), 36  
 running\_task\_count (*qarnot.connection.UserInfo* attribute), 10  
 RunningInstancesInfo (class in *qarnot.status*), 36

## S

s3client (*qarnot.connection.Connection* attribute), 5  
 s3resource (*qarnot.connection.Connection* attribute), 5  
 set\_task\_dependencies\_from\_tasks () (*qarnot.task.Task* method), 21  
 set\_task\_dependencies\_from\_uuids () (*qarnot.task.Task* method), 21  
 setup\_elastic () (*qarnot.pool.Pool* method), 28  
 shortname (*qarnot.job.Job* attribute), 25  
 shortname (*qarnot.pool.Pool* attribute), 30  
 shortname (*qarnot.task.Task* attribute), 18  
 snapshot () (*qarnot.task.Task* method), 15  
 snapshot\_blacklist (*qarnot.task.Task* attribute), 19  
 snapshot\_whitelist (*qarnot.task.Task* attribute), 19  
 state (*qarnot.job.Job* attribute), 24  
 state (*qarnot.pool.Pool* attribute), 29  
 state (*qarnot.task.CompletedInstance* attribute), 22  
 state (*qarnot.task.Task* attribute), 15  
 Status (class in *qarnot.status*), 34  
 status (*qarnot.pool.Pool* attribute), 31  
 status (*qarnot.task.Task* attribute), 20  
 status\_code (*qarnot.task.BulkTaskResponse* attribute), 23  
 stderr () (*qarnot.task.Task* method), 17  
 stdout () (*qarnot.task.Task* method), 16  
 Storage (class in *qarnot.storage*), 40  
 submit () (*qarnot.job.Job* method), 26  
 submit () (*qarnot.pool.Pool* method), 28  
 submit () (*qarnot.task.Task* method), 12  
 submit\_tasks () (*qarnot.connection.Connection* method), 9  
 succeeded\_range (*qarnot.status.Status* attribute), 35  
 sync\_directory () (*qarnot.bucket.Bucket* method), 42  
 sync\_files () (*qarnot.bucket.Bucket* method), 43

## T

tags (*qarnot.pool.Pool* attribute), 30  
 tags (*qarnot.task.Task* attribute), 18  
 Task (class in *qarnot.task*), 11  
 task\_count (*qarnot.connection.UserInfo* attribute), 10  
 TaskActiveForward (class in *qarnot.status*), 39  
 tasks (*qarnot.job.Job* attribute), 24  
 tasks () (*qarnot.connection.Connection* method), 6  
 tasks\_default\_wait\_for\_pool\_resources\_synchronization (*qarnot.pool.Pool* attribute), 33  
 TaskVpnConnection (class in *qarnot.status*), 39  
 terminate () (*qarnot.job.Job* method), 26  
 timestamp (*qarnot.status.RunningInstancesInfo* attribute), 36  
 total\_network\_in\_kbps (*qarnot.status.RunningInstancesInfo* attribute), 37  
 total\_network\_out\_kbps (*qarnot.status.RunningInstancesInfo* attribute), 37

## U

UnauthorizedException, 45  
 update () (*qarnot.bucket.Bucket* method), 44  
 update () (*qarnot.job.Job* method), 26  
 update () (*qarnot.pool.Pool* method), 28  
 update () (*qarnot.storage.Storage* method), 41  
 update () (*qarnot.task.Task* method), 14  
 update\_cache\_time (*qarnot.job.Job* attribute), 24  
 update\_cache\_time (*qarnot.pool.Pool* attribute), 31  
 update\_cache\_time (*qarnot.task.Task* attribute), 21  
 update\_resources () (*qarnot.task.Task* method), 13  
 upload\_progress (*qarnot.status.Status* attribute), 34  
 upload\_time (*qarnot.status.Status* attribute), 35  
 upload\_time\_sec (*qarnot.status.Status* attribute), 35  
 use\_dependencies (*qarnot.job.Job* attribute), 24  
 used\_quota\_bytes\_bucket (*qarnot.connection.UserInfo* attribute), 10  
 user\_info (*qarnot.connection.Connection* attribute), 5  
 UserInfo (class in *qarnot.connection*), 9  
 uuid (*qarnot.bucket.Bucket* attribute), 44  
 uuid (*qarnot.job.Job* attribute), 25  
 uuid (*qarnot.pool.Pool* attribute), 29  
 uuid (*qarnot.task.BulkTaskResponse* attribute), 23  
 uuid (*qarnot.task.Task* attribute), 17

## V

vpn\_connections (*qarnot.status.PerRunningInstanceInfo* attribute), 38  
 vpn\_name (*qarnot.status.TaskVpnConnection* attribute), 39

## W

wait () (*qarnot.task.Task* method), 14  
 wait\_for\_pool\_resources\_synchronization (*qarnot.task.Task* attribute), 21  
 wall\_time (*qarnot.status.Status* attribute), 35  
 wall\_time\_sec (*qarnot.status.Status* attribute), 35  
 wall\_time\_sec (*qarnot.task.CompletedInstance* attribute), 22