godocker_web Documentation

Release 1.0

Olivier Sallou

Nov 16, 2018

Contents

Contents:

CHAPTER 1

GO-D-Docker WEB API

Most API calls need authentication. To get a token, one should call endpoint /api/1.0/authenticate. With result token, token should be placed in authorization header of next calls, example:

```
import requests
headers=  {
    'Authorization':'Bearer XXXXX',
    'Content-type': 'application/json',
    'Accept':'application/json'
}
res = requests.get('http://godockerurli//api/1.0/task/active', headers=header)
```

Task object example:

```
task = {
    'id': task_id,  # Must not be provided in POST request to create a task
    'user': { # Filled by Auth plugin, must not be provided in POST request to create␣
↪a task
        'id': 'user_id',
        'uid': 1001,
        'gid': 1001,
        'project': 'default' # project assigned to the task, if none in particular,␣
↪use 'default'
    },
    'notify': {
        'email': false  # Optional email notification on task status modification␣
↪(running, over, ...)
    },
    'date': time.mktime(dt.timetuple()), # Must not be provided in POST request to␣
↪create a task
    'meta': {
        'name': 'some_name',
        'description': 'blabla'
    },
    'requirements': {
```

(continues on next page)

(continued from previous page)

```
        'cpu': 1,
        # In Gb
        'ram': 1,
        'array': {
            'values': None  # Job arrays  start:end:step
        },
        'label': ['storage==ssd'] or None, # See Docker labels
        'tasks': [] # Optional, Ids of parent tasks, current task will not be␣
↪scheduled before for the end of parent task,
        'ports': [] # Optional, list of ports to open
        'uris': [] # Optional, list of data/uris saved in file godocker-uris.txt in␣
↪task directory (one line per entry) to execute a generic script against different␣
↪data
        'failure_policy': 0 # Optional number of restart in case of node failure,␣
↪cannot exceed global configuration limit
        'network': 'public' # GoDocker >= 1.2, Optional network to use when CNI is␣
↪used by executor backend (public/user/project)
    },
    'container': {
        'image': 'centos:latest',
        'volumes': [],
        'network': True,
        'id': None,
        'meta': None, # Contains meta information provided by executor (hostname, ...)
        'stats': None,
        'ports': [], # Will be automatically filled for interactive jobs by mapped␣
↪ports
        'root': False
    },
    'command': {
        'interactive': False,
        'cmd': '/bin/sleep 30'
    },
    'status': {
        'primary': None,
        'secondary': None
    }
}
```

Task schema for task creation is available in source directory at doc/godocker.json

1.1 Go-d-docker web API reference

godweb.views.**admin_maintenance**(*request*)

GET /admin/maintenance

Response JSON Object

- **list** – general status and per node status

Status Codes

- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized

godweb.views.**admin_set_maintenance**(*request*)

POST /admin/maintenance/(**str:** *node*)**/**

str: *status* Admin only, set maintenance status for system or a node node: *all* for system, else node name When system is in maintenance, new tasks will be rejected with a 503 error. Existing tasks (running, pending, ...) are not impacted but pending tasks will not be scheduled.
Note: for the moment, only system maintenance is available, setting a specific node will have no impact and should be used for information only. See issue #32.

Response JSON Object
- **list** – general status and per node status

Status Codes
- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized

godweb.views.**api_app_token_add**(*request*)
Generate a new application token

godweb.views.**api_app_token_delete**(*request*)
Delete an application token

godweb.views.**api_app_tokens**(*request*)
Get applications token

godweb.views.**api_app_user_token**(*request*)
Get user info for a token application

Header must contain a JWT encoded bearer containing same app_token as requested token. Bearer must be decoded using shared_secret_passphrase.

godweb.views.**api_archive**(*request*)

GET /api/1.0/archive/(**int:** *days*)
Archive all tasks older than *days*

Response JSON Object
- **dict** – Task

Status Codes
- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_archive_delete**(*request*)

DELETE /api/1.0/archive/(**int:** *days*)**?tag=**
string: *tag* Delete all tasks older than *days*, optionally matching tag

Response JSON Object
- **dict** – Task

Status Codes
- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_auth**(*request*)

POST /api/1.0/authenticate

Authenticate user to get an Authorization bearer token used in future API calls

Request JSON Object

- **body** (*dict*) – JSON credentials {'apikey': xx, 'user': user_id}

Response JSON Object

- **dict** – {'token': generated_token}

Status Codes

- 200 OK – no error
- 401 Unauthorized – need login/authentication

godweb.views.**api_config**(*request*)

GET /api/1.0/config

Get general configuration

Response JSON Object

- **dict** – configuration object with volumes etc…

Status Codes

- 200 OK – no error

godweb.views.**api_count_running**(*request*)

GET /api/1.0/task/status/(**str:** *status*)**/count**

Count number of tasks for all users

status request parameter should be in ['pending', 'running', 'kill', 'all']

Response JSON Object

- **dict** – {'total': number of tasks, 'status': requested status}

Status Codes

- 200 OK – no error

godweb.views.**api_image_count**(*request*)

GET /api/1.0/image/(**str:** *image*)

List used images

Response JSON Object

- **dict** – List of used images

Status Codes

- 200 OK – no error

godweb.views.**api_images**(*request*)

GET /api/1.0/image

List used images

Response JSON Object

- **dict** – List of used images

Status Codes

- 200 OK – no error

godweb.views.**api_marketplace_recipe**(*request*)

GET /api/1.0/marketplace/recipe/{str:recipe}

Get a public or owned recipe

Response JSON Object
- **dict** – {'id': recipe_id, 'description': recipe_description, 'task': task_originating_id}

Status Codes
- 200 OK – no error
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_marketplace_recipe_delete**(*request*)

DELETE /api/1.0/marketplace/recipe/{str:recipe}

Delete a owned recipe

Status Codes
- 200 OK – no error
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_marketplace_recipe_edit**(*request*)

PUT /api/1.0/marketplace/recipe/{str:recipe}

Update an owned recipe. Put body can contain description and/or task elements

Status Codes
- 200 OK – no error
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_marketplace_recipe_new**(*request*)

POST /api/1.0/marketplace/recipe/{str:recipe}

Create a recipe. Post body must contain description and task elements

Status Codes
- 200 OK – no error
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_marketplace_recipes**(*request*)

GET /api/1.0/marketplace/recipe

Get public recipes and user private recipes if logged

Response JSON Object
- **list** – {'id': recipe_id, 'description': recipe_description}

Status Codes
- 200 OK – no error

godweb.views.**api_ping**(*request*)

GET /api/1.0/ping

Ping web site

Response JSON Object
- **dict** – {'msg': 'pong'}

Status Codes
- 200 OK – no error

godweb.views.**api_project_create**(*request*)

POST /api/1.0/project

Create a project (admin only)

:<json dict:project :statuscode 200: no error :statuscode 401: need login/authentication :statuscode 403: not authorized

godweb.views.**api_project_delete**(*request*)

DELETE /api/1.0/project/{str:project}

Delete project details (admin only)

Statuscode 200 no error

Statuscode 401 need login/authentication

Statuscode 403 not authorized

godweb.views.**api_project_edit**(*request*)

POST /api/1.0/project/{str:project}

Update a project (owner and admin only)

Fields: description, members, prio, quota_time, quota_cpu, quota_ram, quota_gpu

:<json dict:project :statuscode 200: no error :statuscode 401: need login/authentication :statuscode 403: not authorized

godweb.views.**api_project_get**(*request*)

GET /api/1.0/project/{str:project}

Get project details (Admin only)

:>json dict:project :statuscode 200: no error :statuscode 401: need login/authentication :statuscode 403: not authorized

godweb.views.**api_project_list**(*request*)

GET /api/1.0/project

Get the list of projects (Admin only) or projects owner by user

>json list List of projects

Statuscode 200 no error

Statuscode 401 need login/authentication

Statuscode 403 not authorized

godweb.views.**api_project_quota_reset**(*request*)

DELETE /api/1.0/project/(**str:** *id*)**/quota**

Reset project usage quota (admin)

Status Codes

- 200 OK – no error

godweb.views.**api_project_usage**(*request*)

GET /api/1.0/project/(**str:** *id*)**/usage**

Get project usage (in project or admin)

Response JSON Object

- **list** – project usage {'cpu','ram','time'}

Status Codes

- 200 OK – no error

godweb.views.**api_prometheus**(*request*)

POST /api/1.0/prometheus

Push prometheus metrics, admin only via prometheus_key config parameter

Status Codes

- 200 OK – no error

godweb.views.**api_task**(*request*)

PUT /api/1.0/task/(**str:** *id*)

Update a task, only requirements dynamic fields are supported dynamic_fields in config file. Other parameters cannot be modified once task is create

Response JSON Object

- **list** – Update object with name/value pair, example {"name" : "maxlifespan", "value": "6d"}

Status Codes

- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized

godweb.views.**api_task_active**(*request*)

GET /api/1.0/task/active

List running or pending tasks for user

Response JSON Object

- **dict** – List of tasks

Status Codes

- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized

godweb.views.**api_task_active_all**(*request*)

GET /api/1.0/task/active/all

List running or pending tasks for all users Restricted to administrators

Response JSON Object

- **dict** – List of tasks

Status Codes

- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized

godweb.views.**api_task_add**(*request*)

POST /api/1.0/task

Create a new task for user

Request JSON Object

- **body** (*dict*) – JSON task to add

Response JSON Object

- **dict** – {'msg': 'Task added', 'id': task_id}

Status Codes

- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 503 Service Unavailable – service unavailable

godweb.views.**api_task_archive**(*request*)

GET /api/1.0/task/(**int:** *task*)**/archive**

Archive a task over

Response JSON Object

- **dict** – Task

Status Codes

- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_task_archive_delete**(*request*)

DELETE /api/1.0/task/(**int:** *task*)**/archive**

Deletes a task over

Response JSON Object

- **dict** – Task

Status Codes

- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_task_file**(*request*)

GET /api/1.0/task/(**int:** *task*)**/files/*file**

List task files if *file is empty or a subdirectory of task directory, else download file matching *file

Accept request header X-GODOCKER-SHARE containing a share token to access files

Response JSON Object

- **list** – List of files and directory in *file if *file is a directory

::>octet-stream: File content if *file is a file :statuscode 200: no error :statuscode 401: need login/authentication :statuscode 403: not authorized :statuscode 404: not found :statuscode 503: service unavailable

godweb.views.**api_task_file_share**(*request*)

GET /api/1.0/task/(**int:** *task*)**/share/{str:token}/files/*file**

List shared task files if *file is empty or a subdirectory of task directory, else download file matching *file

Response JSON Object

- **list** – List of files and directory in *file if *file is a directory

::>octet-stream: File content if *file is a file :statuscode 200: no error :statuscode 401: need login/authentication :statuscode 403: not authorized :statuscode 404: not found :statuscode 503: service unavailable

godweb.views.**api_task_get**(*request*)

GET /api/1.0/task/(**int:** *task*)

Get a task X-GODOCKER-SHARE in headers to access via a share token.

Accept

Response JSON Object

- **dict** – Task

Status Codes

- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_task_kill**(*request*)

DELETE /api/1.0/task/(**int:** *task*)

Kill a running task

Response JSON Object

- **dict** – Task

Status Codes

- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_task_monitor**(*request*)

GET /api/1.0/task/(**int:** *container*)**/monitor**

Get monitoring info for container

Response JSON Object

- **dict** – Task

Status Codes

- 200 OK – no error
- 404 Not Found – not found

godweb.views.**api_task_over**(*request*)

GET /api/1.0/task/over

List last 100 finished tasks, or the last X tasks, if *limit* request parameter is set

Support for pagination with following optional request parameters: draw: set this to manage pagination and filtering start_date: timestamp to limit search in time end_date: timestamp to limit search in time start: number of tasks to skip limit: number of results to return reverse: set to *true* to reverse the results orderBy: field used to sort the results [id, name, container_image, user_id] regex: text to use to filter the results container regex value (case insensitive)

If pagination is enabled with *draw* parameter, result is an array of 1 element with fields: data: list of tasks matching the conditions recordsTotal: total number of records

Response JSON Object
- **list** – List of tasks

Status Codes
- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized

godweb.views.**api_task_over_all**(*request*)

GET /api/1.0/task/over/all

List over for all users (last 100) Restricted to administrators

Response JSON Object
- **dict** – List of tasks

Status Codes
- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized

godweb.views.**api_task_over_all_cursor**(*request*)

POST /api/1.0/task/over/all/(**int:** *skip*)**/**
int: *limit*

Get finished tasks for all users (Administrator only)

Params: skip: number of records to skip in search, limit: maximum number of records to return

Json optional filter: {'start': timestamp, 'end': timestamp}

Request JSON Object
- **dict** – Optional filter on task start date (timestamp)

Response JSON Object
- **dict** – List of tasks

Status Codes
- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized

godweb.views.**api_task_over_cursor**(*request*)

POST /api/1.0/task/over/(**int:** *skip*)**/**
int: *limit*

Get finished tasks

Params: skip: number of records to skip in search, limit: maximum number of records to return

Json optional filter: {'start': timestamp, 'end': timestamp}

Request JSON Object
- **dict** – Optional filter on task start date (timestamp)

Response JSON Object
- **dict** – List of tasks

Status Codes
- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized

godweb.views.**api_task_pending**(*request*)

GET /api/1.0/task/(**int:** *task*)**/pending**
Switch a task in CREATED status to PENDING

Response JSON Object
- **dict** – Task

Status Codes
- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_task_reschedule**(*request*)

GET /api/1.0/task/(**int:** *task*)**/reschedule**
Reschedule a running task

Response JSON Object
- **dict** – Task

Status Codes
- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_task_resume**(*request*)

GET /api/1.0/task/(**int:** *task*)**/resume**
Resume a suspended task

Response JSON Object
- **dict** – Task

Status Codes
- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_task_share**(*request*)

GET /api/1.0/task/(**int:** *task*)**/share**
Generate a token to share task with other people

Response JSON Object
- **dict** – dict containing token to use with /api/1.0/task/XXX?share=token

Status Codes
- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized
- 404 Not Found – not found
- 503 Service Unavailable – service unavailable

godweb.views.**api_task_status**(*request*)

GET /api/1.0/task/(**int:** *task*)**/status**
Get a task status

Response JSON Object
- **dict** – Task

Status Codes
- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_task_suspend**(*request*)

GET /api/1.0/task/(**int:** *task*)**/suspend**

Suspend a running task

Response JSON Object
- **dict** – Task

Status Codes
- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized
- 404 Not Found – not found

godweb.views.**api_task_token**(*request*)

GET /api/1.0/task/(**int:** *task*)**/token**

Get a JWT encoded task to securely exchange a task with other services.

>json dict {'token': generated_token}

Statuscode 200 no error

Statuscode 401 need login/authentication

godweb.views.**api_task_unshare**(*request*)

GET /api/1.0/task/(**int:** *task*)**/unshare**

Generate a token to share task with other people

Response JSON Object
- **dict** – dict containing token to use with /api/1.0/task/XXX?share=token

Status Codes
- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized
- 404 Not Found – not found
- 503 Service Unavailable – service unavailable

godweb.views.**api_usage**(*request*)

GET /api/1.0/usage

Get framework resource usage

Response JSON Object
- **list** – list of nodes with resources

Status Codes
- 200 OK – no error

godweb.views.**api_user_apikey**(*request*)

DELETE /api/1.0/user/{str:user}/apikey

Generate a new api key

>json dict new apikey

Statuscode 200 no error

Statuscode 401 need login/authentication

godweb.views.**api_user_billing**(*request*)

GET /api/1.0/(**str:** *type*)**/**
str: *id***/billing/:year/:month** Get user usage (self or admin) for month and year, per project, or a per project whatever the users if project belongs to user type=user/project Optional *force* query parameter to force a refresh and not using cache (force=0/1)

Response JSON Object

- **list** – user usage { "_id" : { "user" : "XX", "project" : "default" }, "cpu" : 758, "ram" : 1031, 'gpus': 3, 'totaltime': 12434 }

Status Codes

- 200 OK – no error

godweb.views.**api_user_file**(*request*)

GET /api/1.0/user/(**int:** *id*)**/files/*file**
List user files from personal storage if *file is empty or a subdirectory of task directory, else download file matching *file

Response JSON Object

- **list** – List of files and directory in *file if *file is a directory

::>octet-stream: File content if *file is a file :statuscode 200: no error :statuscode 401: need login/authentication :statuscode 403: not authorized :statuscode 404: not found :statuscode 503: service unavailable

godweb.views.**api_user_project_list**(*request*)

GET /api/1.0/user/{str:user}/project

Get the list of projects for user

>json list List of projects

Statuscode 200 no error

Statuscode 401 need login/authentication

Statuscode 403 not authorized

godweb.views.**api_user_task_delete**(*request*)

DELETE /api/1.0/user/(**str:** *id*)**/task**
Kill all user tasks (self or admin), optionally tasks with input query parameter *tag*=MYTAG

Response JSON Object

- **list** – list of task ids

Status Codes

- 200 OK – no error
- 401 Unauthorized – need login/authentication
- 403 Forbidden – not authorized

godweb.views.**guest_status**(*request*)
Update status of a user, user is in body

godweb.views.**prometheus**(*request*)

GET /metrics
Prometheus metrics
Status Codes
- 200 OK – no error

godweb.views.**prometheus_api_inc**(*request*, *method*)
Increment API calls for prometheus

godweb.views.**redis_call**(*request*, *f*, **args*, ***kwargs*)
Executes a redis call, and retries connection and command in case of failure.

Example: redis_call(self.r.hset, field, keys, 1)

godweb.views.**user_delete**(*request*)

PUT /api/1.0/user/(**str:** *id*)
Delete user and all jobs info/data executed by this user (admin) WARNING: this operation cannot be reverted, all user info and jobs will be deleted
Status Codes
- 200 OK – no error

godweb.views.**user_info**(*request*)

GET /api/1.0/user/(**str:** *id*)
Get user information (self or admin)
Response JSON Object
- **dict** – user data
Status Codes
- 200 OK – no error

godweb.views.**user_info_edit**(*request*)

PUT /api/1.0/user/(**str:** *id*)
Update logged in user information (self or admin)
Request JSON Object
- **dict** – user data to update
Response JSON Object
- **dict** – user data
Status Codes
- 200 OK – no error

godweb.views.**user_list**(*request*)

GET /api/1.0/user (Admin only)
List users
Request JSON Object
- **dict** – user data to update
Response JSON Object
- **dict** – user data
Status Codes

- 200 OK – no error

godweb.views.**user_usage**(*request*)

GET /api/1.0/user/(**str:** *id*)**/usage**

Get user usage (self or admin)

Response JSON Object

- **list** – user usage {'cpu','ram','time'}

Status Codes

- 200 OK – no error

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

HTTP Routing Table

/admin

/api

10
DELETE /api/1.0/user/(str:id)/task, 15
DELETE /api/1.0/user/{str:user}/apikey,
14

/metrics

GET /metrics, 16

Python Module Index

g

Index