
python-gitlab Documentation

Release 1.11.0

Gauvain Pocentek, Mika Mäenpää

Aug 31, 2019

Contents

1	Installation	3
2	gitlab CLI usage	5
2.1	Configuration	5
2.2	CLI	6
2.3	Examples	7
3	Getting started with the API	9
3.1	gitlab.Gitlab class	9
3.2	Managers	10
3.3	Gitlab Objects	11
3.4	Base types	11
3.5	Lazy objects	12
3.6	Pagination	12
3.7	Sudo	13
3.8	Advanced HTTP configuration	13
4	FAQ	15
5	Switching to GitLab API v4	17
5.1	Using the v4 API	17
5.2	Changes between v3 and v4 API	17
6	API examples	19
6.1	Access requests	19
6.2	Award Emojis	20
6.3	Badges	21
6.4	Branches	22
6.5	Broadcast messages	23
6.6	Commits	24
6.7	Deploy keys	26
6.8	Deployments	27
6.9	Discussions	28
6.10	Environments	30
6.11	Events	31
6.12	Epics	31
6.13	Features flags	33

6.14	Geo nodes	33
6.15	Groups	34
6.16	Issues	38
6.17	Issue boards	42
6.18	Labels	44
6.19	Notification settings	45
6.20	Merge requests	47
6.21	Merge request approvals settings	50
6.22	Milestones	51
6.23	Namespaces	52
6.24	Notes	52
6.25	Pages domains	53
6.26	Pipelines and Jobs	55
6.27	Projects	61
6.28	Protected branches	74
6.29	Runners	75
6.30	Registry Repositories	77
6.31	Registry Repository Tags	77
6.32	Search API	78
6.33	Settings	79
6.34	Snippets	80
6.35	System hooks	81
6.36	Templates	81
6.37	Todos	83
6.38	Users and current user	84
6.39	Sidekiq metrics	90
6.40	Wiki pages	90
7	gitlab package	93
7.1	Subpackages	93
7.2	Submodules	168
7.3	gitlab.base module	168
7.4	gitlab.cli module	169
7.5	gitlab.config module	169
7.6	gitlab.const module	170
7.7	gitlab.exceptions module	170
7.8	gitlab.mixins module	174
7.9	gitlab.utils module	180
7.10	Module contents	180
8	Release notes	187
8.1	Changes from 1.8 to 1.9	187
8.2	Changes from 1.7 to 1.8	187
8.3	Changes from 1.5 to 1.6	188
8.4	Changes from 1.4 to 1.5	188
8.5	Changes from 1.3 to 1.4	188
8.6	Changes from 1.2 to 1.3	189
8.7	Changes from 1.1 to 1.2	189
8.8	Changes from 1.0.2 to 1.1	189
8.9	Changes from 0.21 to 1.0.0	190
8.10	Changes from 0.20 to 0.21	190
8.11	Changes from 0.19 to 0.20	190
9	ChangeLog	191

9.1	Version 1.9.0 - 2019-06-19	191
9.2	Version 1.8.0 - 2019-02-22	192
9.3	Version 1.7.0 - 2018-12-09	192
9.4	Version 1.6.0 - 2018-08-25	193
9.5	Version 1.5.1 - 2018-06-23	193
9.6	Version 1.5.0 - 2018-06-22	193
9.7	Version 1.4.0 - 2018-05-19	195
9.8	Version 1.3.0 - 2018-02-18	195
9.9	Version 1.2.0 - 2018-01-01	196
9.10	Version 1.1.0 - 2017-11-03	197
9.11	Version 1.0.2 - 2017-09-29	198
9.12	Version 1.0.1 - 2017-09-21	198
9.13	Version 1.0.0 - 2017-09-08	198
9.14	Version 0.21.2 - 2017-06-11	199
9.15	Version 0.21.1 - 2017-05-25	199
9.16	Version 0.21 - 2017-05-24	199
9.17	Version 0.20 - 2017-03-25	200
9.18	Version 0.19 - 2017-02-21	200
9.19	Version 0.18 - 2016-12-27	200
9.20	Version 0.17 - 2016-12-02	201
9.21	Version 0.16 - 2016-10-16	202
9.22	Version 0.15.1 - 2016-10-16	202
9.23	Version 0.15 - 2016-08-28	202
9.24	Version 0.14 - 2016-08-07	203
9.25	Version 0.13 - 2016-05-16	204
9.26	Version 0.12.2 - 2016-03-19	205
9.27	Version 0.12.1 - 2016-02-03	206
9.28	Version 0.12 - 2016-02-03	206
9.29	Version 0.11.1 - 2016-01-17	206
9.30	Version 0.11 - 2016-01-09	207
9.31	Version 0.10 - 2015-12-29	207
9.32	Version 0.9.2 - 2015-07-11	207
9.33	Version 0.9.1 - 2015-05-15	207
9.34	Version 0.9 - 2015-05-15	208
9.35	Version 0.8 - 2014-10-26	208
9.36	Version 0.7 - 2014-08-21	208
9.37	Version 0.6 - 2014-01-16	209
9.38	Version 0.5 - 2013-12-26	209
9.39	Version 0.4 - 2013-09-26	209
9.40	Version 0.3 - 2013-08-27	209
9.41	Version 0.2 - 2013-08-08	210
9.42	Version 0.1 - 2013-07-08	210
10	Indices and tables	211
	Python Module Index	213
	Index	215

Contents:

CHAPTER 1

Installation

python-gitlab is compatible with Python 2.7 and 3.4+.

Use **pip** to install the latest stable version of python-gitlab:

```
$ sudo pip install --upgrade python-gitlab
```

The current development version is available on [github](https://github.com). Use **git** and **python setup.py** to install it:

```
$ git clone https://github.com/python-gitlab/python-gitlab
$ cd python-gitlab
$ sudo python setup.py install
```


`python-gitlab` provides a **gitlab** command-line tool to interact with GitLab servers. It uses a configuration file to define how to connect to the servers.

2.1 Configuration

2.1.1 Files

`gitlab` looks up 2 configuration files by default:

`/etc/python-gitlab.cfg` System-wide configuration file

`~/.python-gitlab.cfg` User configuration file

You can use a different configuration file with the `--config-file` option.

2.1.2 Content

The configuration file uses the INI format. It contains at least a `[global]` section, and a specific section for each GitLab server. For example:

```
[global]
default = somewhere
ssl_verify = true
timeout = 5

[somewhere]
url = https://some.whe.re
private_token = vTbFeqJYCY3sibBP7BZM
api_version = 3

[elsewhere]
```

(continues on next page)

(continued from previous page)

```
url = http://else.whe.re:8080
private_token = CkqsjqcQSFH5FQKDccu4
timeout = 1
```

The default option of the [global] section defines the GitLab server to use if no server is explicitly specified with the `--gitlab` CLI option.

The [global] section also defines the values for the default connection parameters. You can override the values in each GitLab server section.

Table 1: Global options

Option	Possible values	Description
<code>ssl_verify</code>	<code>True</code> , <code>False</code> , or a <code>str</code>	Verify the SSL certificate. Set to <code>False</code> to disable verification, though this will create warnings. Any other value is interpreted as path to a <code>CA_BUNDLE</code> file or directory with certificates of trusted CAs.
<code>timeout</code>	Integer	Number of seconds to wait for an answer before failing.
<code>api_version</code>	3 or 4	The API version to use to make queries. Requires <code>python-gitlab >= 1.3.0</code> .
<code>per_page</code>	Integer between 1 and 100	The number of items to return in listing queries. GitLab limits the value at 100.

You must define the `url` in each GitLab server section.

Warning: If the GitLab server you are using redirects requests from `http` to `https`, make sure to use the `https://` protocol in the `url` definition.

Only one of `private_token` or `oauth_token` should be defined. If neither are defined an anonymous request will be sent to the Gitlab server, with very limited permissions.

Table 2: GitLab server options

Option	Description
<code>url</code>	URL for the GitLab server
<code>private_token</code>	Your user token. Login/password is not supported. Refer to the official documentation to learn how to obtain a token.
<code>oauth_token</code>	An OAuth token for authentication. The Gitlab server must be configured to support this authentication method.
<code>api_version</code>	GitLab API version to use (3 or 4). Defaults to 4 since version 1.3.0.
<code>http_username</code>	Username for optional HTTP authentication
<code>http_password</code>	Password for optional HTTP authentication

2.2 CLI

2.2.1 Objects and actions

The `gitlab` command expects two mandatory arguments. The first one is the type of object that you want to manipulate. The second is the action that you want to perform. For example:

```
$ gitlab project list
```

Use the `--help` option to list the available object types and actions:

```
$ gitlab --help
$ gitlab project --help
```

Some actions require additional parameters. Use the `--help` option to list mandatory and optional arguments for an action:

```
$ gitlab project create --help
```

2.2.2 Optional arguments

Use the following optional arguments to change the behavior of `gitlab`. These options must be defined before the mandatory arguments.

--verbose, -v Outputs detail about retrieved objects. Available for legacy (default) output only.

--config-file, -c Path to a configuration file.

--gitlab, -g ID of a GitLab server defined in the configuration file.

--output, -o Output format. Defaults to a custom format. Can also be `yaml` or `json`.

Notice:

The [PyYAML package](#) is required to use the `yaml` output option. You need to install it separately using `pip install PyYAML`

--fields, -f Comma-separated list of fields to display (`yaml` and `json` output formats only). If not used, all the object fields are displayed.

Example:

```
$ gitlab -o yaml -f id,permissions -g elsewhere -c /tmp/gl.cfg project list
```

2.3 Examples

List the projects (paginated):

```
$ gitlab project list
```

List all the projects:

```
$ gitlab project list --all
```

Limit to 5 items per request, display the 1st page only

```
$ gitlab project list --page 1 --per-page 5
```

Get a specific project (id 2):

```
$ gitlab project get --id 2
```

Get a specific user by id:

```
$ gitlab user get --id 3
```

Get a list of snippets for this project:

```
$ gitlab project-issue list --project-id 2
```

Delete a snippet (id 3):

```
$ gitlab project-snippet delete --id 3 --project-id 2
```

Update a snippet:

```
$ gitlab project-snippet update --id 4 --project-id 2 \  
  --code "My New Code"
```

Create a snippet:

```
$ gitlab project-snippet create --project-id 2  
Impossible to create object (Missing attribute(s): title, file-name, code)  
$ # oops, let's add the attributes:  
$ gitlab project-snippet create --project-id 2 --title "the title" \  
  --file-name "the name" --code "the code"
```

Define the status of a commit (as would be done from a CI tool for example):

```
$ gitlab project-commit-status create --project-id 2 \  
  --commit-id a43290c --state success --name ci/jenkins \  
  --target-url http://server/build/123 \  
  --description "Jenkins build succeeded"
```

Use sudo to act as another user (admin only):

```
$ gitlab project create --name user_project1 --sudo username
```

List values are comma-separated:

```
$ gitlab issue list --labels foo,bar
```

2.3.1 Reading values from files

You can make `gitlab` read values from files instead of providing them on the command line. This is handy for values containing new lines for instance:

```
$ cat > /tmp/description << EOF  
This is the description of my project.  
  
It is obviously the best project around  
EOF  
$ gitlab project create --name SuperProject --description @/tmp/description
```

Getting started with the API

python-gitlab only supports GitLab APIs v4.

3.1 gitlab.Gitlab class

To connect to a GitLab server, create a `gitlab.Gitlab` object:

```
import gitlab

# private token or personal token authentication
gl = gitlab.Gitlab('http://10.0.0.1', private_token='JVNSEs8EwWRx5yDxM5q')

# oauth token authentication
gl = gitlab.Gitlab('http://10.0.0.1', oauth_token='my_long_token_here')

# username/password authentication (for GitLab << 10.2)
gl = gitlab.Gitlab('http://10.0.0.1', email='jdoe', password='s3cr3t')

# anonymous gitlab instance, read-only for public resources
gl = gitlab.Gitlab('http://10.0.0.1')

# make an API request to create the gl.user object. This is mandatory if you
# use the username/password authentication.
gl.auth()
```

You can also use configuration files to create `gitlab.Gitlab` objects:

```
gl = gitlab.Gitlab.from_config('somewhere', ['/tmp/gl.cfg'])
```

See the *Configuration* section for more information about configuration files.

Warning: If the GitLab server you are using redirects requests from http to https, make sure to use the `https://` protocol in the URL definition.

3.1.1 Note on password authentication

The `/session` API endpoint used for username/password authentication has been removed from GitLab in version 10.2, and is not available on gitlab.com anymore. Personal token authentication is the preferred authentication method.

If you need username/password authentication, you can use cookie-based authentication. You can use the web UI form to authenticate, retrieve cookies, and then use a custom `requests.Session` object to connect to the GitLab API. The following code snippet demonstrates how to automate this: <https://gist.github.com/gpocentek/bd4c3fbf8a6ce226ebddc4aad6b46c0a>.

See [issue 380](#) for a detailed discussion.

3.2 Managers

The `gitlab.Gitlab` class provides managers to access the GitLab resources. Each manager provides a set of methods to act on the resources. The available methods depend on the resource type.

Examples:

```
# list all the projects
projects = gl.projects.list()
for project in projects:
    print(project)

# get the group with id == 2
group = gl.groups.get(2)
for project in group.projects.list():
    print(project)

# create a new user
user_data = {'email': 'jen@foo.com', 'username': 'jen', 'name': 'Jen'}
user = gl.users.create(user_data)
print(user)
```

You can list the mandatory and optional attributes for object creation and update with the manager's `get_create_attrs()` and `get_update_attrs()` methods. They return 2 tuples, the first one is the list of mandatory attributes, the second one is the list of optional attribute:

```
# v4 only
print(gl.projects.get_create_attrs())
(('name',), ('path', 'namespace_id', ...))
```

The attributes of objects are defined upon object creation, and depend on the GitLab API itself. To list the available information associated with an object use the `attributes` attribute:

```
project = gl.projects.get(1)
print(project.attributes)
```

Some objects also provide managers to access related GitLab resources:


```
# list the issues for a project
project = gl.projects.get(1)
issues = project.issues.list()
```

python-gitlab allows to send any data to the GitLab server when making queries. In case of invalid or missing arguments python-gitlab will raise an exception with the GitLab server error message:

```
>>> gl.projects.list(sort='invalid value')
...
GitlabListError: 400: sort does not have a valid value
```

You can use the `query_parameters` argument to send arguments that would conflict with python or python-gitlab when using them as kwargs:

```
gl.user_activities.list(from='2019-01-01') ## invalid
gl.user_activities.list(query_parameters={'from': '2019-01-01'}) # OK
```

3.3 Gitlab Objects

You can update or delete a remote object when it exists locally:

```
# update the attributes of a resource
project = gl.projects.get(1)
project.wall_enabled = False
# don't forget to apply your changes on the server:
project.save()

# delete the resource
project.delete()
```

Some classes provide additional methods, allowing more actions on the GitLab resources. For example:

```
# star a git repository
project = gl.projects.get(1)
project.star()
```

3.4 Base types

The `gitlab` package provides some base types.

- `gitlab.Gitlab` is the primary class, handling the HTTP requests. It holds the GitLab URL and authentication information.
- `gitlab.base.RESTObject` is the base class for all the GitLab v4 objects. These objects provide an abstraction for GitLab resources (projects, groups, and so on).
- `gitlab.base.RESTManager` is the base class for v4 objects managers, providing the API to manipulate the resources and their attributes.

3.5 Lazy objects

To avoid useless API calls to the server you can create lazy objects. These objects are created locally using a known ID, and give access to other managers and methods.

The following example will only make one API call to the GitLab server to star a project (the previous example used 2 API calls):

```
# star a git repository
project = gl.projects.get(1, lazy=True) # no API call
project.star() # API call
```

3.6 Pagination

You can use pagination to iterate over long lists. All the Gitlab objects listing methods support the `page` and `per_page` parameters:

```
ten_first_groups = gl.groups.list(page=1, per_page=10)
```

Warning: The first page is page 1, not page 0.

By default GitLab does not return the complete list of items. Use the `all` parameter to get all the items when using listing methods:

```
all_groups = gl.groups.list(all=True)
all_owned_projects = gl.projects.list(owned=True, all=True)
```

You can define the `per_page` value globally to avoid passing it to every `list()` method call:

```
gl = gitlab.Gitlab(url, token, per_page=50)
```

`list()` methods can also return a generator object which will handle the next calls to the API when required. This is the recommended way to iterate through a large number of items:

```
items = gl.groups.list(as_list=False)
for item in items:
    print(item.attributes)
```

The generator exposes extra listing information as received from the server:

- `current_page`: current page number (first page is 1)
- `prev_page`: if `None` the current page is the first one
- `next_page`: if `None` the current page is the last one
- `per_page`: number of items per page
- `total_pages`: total number of pages available
- `total`: total number of items in the list

3.7 Sudo

If you have the administrator status, you can use `sudo` to act as another user. For example:

```
p = gl.projects.create({'name': 'awesome_project'}, sudo='user1')
```

3.8 Advanced HTTP configuration

`python-gitlab` relies on `requests` `Session` objects to perform all the HTTP requests to the Gitlab servers.

You can provide your own `Session` object with custom configuration when you create a `Gitlab` object.

3.8.1 Context manager

You can use `Gitlab` objects as context managers. This makes sure that the `requests.Session` object associated with a `Gitlab` instance is always properly closed when you exit a `with` block:

```
with gitlab.Gitlab(host, token) as gl:
    gl.projects.list()
```

Warning: The context manager will also close the custom `Session` object you might have used to build the `Gitlab` instance.

3.8.2 Proxy configuration

The following sample illustrates how to define a proxy configuration when using `python-gitlab`:

```
import gitlab
import requests

session = requests.Session()
session.proxies = {
    'https': os.environ.get('https_proxy'),
    'http': os.environ.get('http_proxy'),
}
gl = gitlab.gitlab(url, token, api_version=4, session=session)
```

Reference: <http://docs.python-requests.org/en/master/user/advanced/#proxies>

3.8.3 Client side certificate

The following sample illustrates how to use a client-side certificate:

```
import gitlab
import requests

session = requests.Session()
session.cert = ('/path/to/client.cert', '/path/to/client.key')
gl = gitlab.gitlab(url, token, api_version=4, session=session)
```

Reference: <http://docs.python-requests.org/en/master/user/advanced/#client-side-certificates>

3.8.4 Rate limits

python-gitlab obeys the rate limit of the GitLab server by default. On receiving a 429 response (Too Many Requests), python-gitlab sleeps for the amount of time in the Retry-After header that GitLab sends back. If GitLab does not return a response with the Retry-After header, python-gitlab will perform an exponential backoff.

If you don't want to wait, you can disable the rate-limiting feature, by supplying the `obey_rate_limit` argument.

```
import gitlab
import requests

gl = gitlab.gitlab(url, token, api_version=4)
gl.projects.list(all=True, obey_rate_limit=False)
```

If you do not disable the rate-limiting feature, you can supply a custom value for `max_retries`; by default, this is set to 10. To retry without bound when throttled, you can set this parameter to -1. This parameter is ignored if `obey_rate_limit` is set to `False`.

```
import gitlab
import requests

gl = gitlab.gitlab(url, token, api_version=4)
gl.projects.list(all=True, max_retries=12)
```

Warning: You will get an Exception, if you then go over the rate limit of your GitLab instance.

I cannot edit the merge request / issue I've just retrieved It is likely that you used a `MergeRequest`, `GroupMergeRequest`, `Issue` or `GroupIssue` object. These objects cannot be edited. But you can create a new `ProjectMergeRequest` or `ProjectIssue` object to apply changes. For example:

```
issue = gl.issues.list()[0]
project = gl.projects.get(issue.project_id, lazy=True)
editable_issue = project.issues.get(issue.iid, lazy=True)
# you can now edit the object
```

See the *merge requests example* and the *issues examples*.

How can I clone the repository of a project? `python-gitlab` doesn't provide an API to clone a project. You have to use a `git` library or call the `git` command.

The `git` URI is exposed in the `ssh_url_to_repo` attribute of `Project` objects.

Example:

```
import subprocess

project = gl.projects.create(data) # or gl.projects.get(project_id)
print(project.attributes) # displays all the attributes
git_url = project.ssh_url_to_repo
subprocess.call(['git', 'clone', git_url])
```

Switching to GitLab API v4

GitLab provides a new API version (v4) since its 9.0 release. `python-gitlab` provides support for this new version, but the python API has been modified to solve some problems with the existing one.

GitLab does not support the v3 API anymore, and you should consider switching to v4 if you use a recent version of GitLab (≥ 9.0), or if you use <https://gitlab.com>.

5.1 Using the v4 API

`python-gitlab` uses the v4 API by default since the 1.3.0 release. If you are migrating from an older release, make sure that you remove the `api_version` definition in you constructors and configuration file:

The following examples are **not valid** anymore:

```
gl = gitlab.Gitlab(..., api_version=3)
```

```
[my_gitlab]
...
api_version = 3
```

5.2 Changes between v3 and v4 API

For a list of GitLab (upstream) API changes, see https://docs.gitlab.com/ce/api/v3_to_v4.html.

The `python-gitlab` API reflects these changes. But also consider the following important changes in the python API:

- managers and objects don't inherit from `GitlabObject` and `BaseManager` anymore. They inherit from `RESTManager` and `RESTObject`.
- You should only use the managers to perform CRUD operations.

The following v3 code:

```
gl = gitlab.Gitlab(...)
p = Project(gl, project_id)
```

Should be replaced with:

```
gl = gitlab.Gitlab(...)
p = gl.projects.get(project_id)
```

- Listing methods (`manager.list()` for instance) can now return generators (*RESTObjectList*). They handle the calls to the API when needed to fetch new items.

By default you will still get lists. To get generators use `as_list=False`:

```
all_projects_g = gl.projects.list(as_list=False)
```

- The “nested” managers (for instance `gl.project_issues` or `gl.group_members`) are not available anymore. Their goal was to provide a direct way to manage nested objects, and to limit the number of needed API calls.

To limit the number of API calls, you can now use `get()` methods with the `lazy=True` parameter. This creates shallow objects that provide usual managers.

The following v3 code:

```
issues = gl.project_issues.list(project_id=project_id)
```

Should be replaced with:

```
issues = gl.projects.get(project_id, lazy=True).issues.list()
```

This will make only one API call, instead of two if `lazy` is not used.

- The following *Gitlab* methods should not be used anymore for v4:
 - `list()`
 - `get()`
 - `create()`
 - `update()`
 - `delete()`
- If you need to perform HTTP requests to the GitLab server (which you shouldn't), you can use the following *Gitlab* methods:
 - `http_request`
 - `http_get`
 - `http_list`
 - `http_post`
 - `http_put`
 - `http_delete`

6.1 Access requests

Users can request access to groups and projects.

When access is granted the user should be given a numerical access level. The following constants are provided to represent the access levels:

- `gitlab.GUEST_ACCESS: 10`
- `gitlab.REPORTER_ACCESS: 20`
- `gitlab.DEVELOPER_ACCESS: 30`
- `gitlab.MAINTAINER_ACCESS: 40`
- `gitlab.OWNER_ACCESS: 50`

6.1.1 References

- v4 API:
 - `gitlab.v4.objects.ProjectAccessRequest`
 - `gitlab.v4.objects.ProjectAccessRequestManager`
 - `gitlab.v4.objects.Project.accessrequests`
 - `gitlab.v4.objects.GroupAccessRequest`
 - `gitlab.v4.objects.GroupAccessRequestManager`
 - `gitlab.v4.objects.Group.accessrequests`
- GitLab API: https://docs.gitlab.com/ce/api/access_requests.html

6.1.2 Examples

List access requests from projects and groups:

```
p_ars = project.accessrequests.list()
g_ars = group.accessrequests.list()
```

Create an access request:

```
p_ar = project.accessrequests.create({})
g_ar = group.accessrequests.create({})
```

Approve an access request:

```
ar.approve() # defaults to DEVELOPER level
ar.approve(access_level=gitlab.MAINTAINER_ACCESS) # explicitly set access level
```

Deny (delete) an access request:

```
project.accessrequests.delete(user_id)
group.accessrequests.delete(user_id)
# or
ar.delete()
```

6.2 Award Emojis

6.2.1 Reference

- v4 API:
 - `gitlab.v4.objects.ProjectIssueAwardEmoji`
 - `gitlab.v4.objects.ProjectIssueNoteAwardEmoji`
 - `gitlab.v4.objects.ProjectMergeRequestAwardEmoji`
 - `gitlab.v4.objects.ProjectMergeRequestNoteAwardEmoji`
 - `gitlab.v4.objects.ProjectSnippetAwardEmoji`
 - `gitlab.v4.objects.ProjectSnippetNoteAwardEmoji`
 - `gitlab.v4.objects.ProjectIssueAwardEmojiManager`
 - `gitlab.v4.objects.ProjectIssueNoteAwardEmojiManager`
 - `gitlab.v4.objects.ProjectMergeRequestAwardEmojiManager`
 - `gitlab.v4.objects.ProjectMergeRequestNoteAwardEmojiManager`
 - `gitlab.v4.objects.ProjectSnippetAwardEmojiManager`
 - `gitlab.v4.objects.ProjectSnippetNoteAwardEmojiManager`
- GitLab API: https://docs.gitlab.com/ce/api/award_emoji.html

6.2.2 Examples

List emojis for a resource:

```
emojis = obj.awardemojis.list()
```

Get a single emoji:

```
emoji = obj.awardemojis.get(emoji_id)
```

Add (create) an emoji:

```
emoji = obj.awardemojis.create({'name': 'tractor'})
```

Delete an emoji:

```
emoji.delete
# or
obj.awardemojis.delete(emoji_id)
```

6.3 Badges

Badges can be associated with groups and projects.

6.3.1 Reference

- v4 API:
 - `gitlab.v4.objects.GroupBadge`
 - `gitlab.v4.objects.GroupBadgeManager`
 - `gitlab.v4.objects.Group.badges`
 - `gitlab.v4.objects.ProjectBadge`
 - `gitlab.v4.objects.ProjectBadgeManager`
 - `gitlab.v4.objects.Project.badges`
- GitLab API:
 - https://docs.gitlab.com/ce/api/group_badges.html
 - https://docs.gitlab.com/ce/api/project_badges.html

6.3.2 Examples

List badges:

```
badges = group_or_project.badges.list()
```

Get ad badge:

```
badge = group_or_project.badges.get(badge_id)
```

Create a badge:

```
badge = group_or_project.badges.create({'link_url': link, 'image_url': image_link})
```

Update a badge:

```
badge.image_link = new_link
badge.save()
```

Delete a badge:

```
badge.delete()
```

Render a badge (preview the generate URLs):

```
output = group_or_project.badges.render(link, image_link)
print(output['rendered_link_url'])
print(output['rendered_image_url'])
```

6.4 Branches

6.4.1 References

- v4 API:
 - `gitlab.v4.objects.ProjectBranch`
 - `gitlab.v4.objects.ProjectBranchManager`
 - `gitlab.v4.objects.Project.branches`
- GitLab API: <https://docs.gitlab.com/ce/api/branches.html>

6.4.2 Examples

Get the list of branches for a repository:

```
branches = project.branches.list()
```

Get a single repository branch:

```
branch = project.branches.get('master')
```

Create a repository branch:

```
branch = project.branches.create({'branch': 'feature1',
                                  'ref': 'master'})
```

Delete a repository branch:

```
project.branches.delete('feature1')
# or
branch.delete()
```

Protect/unprotect a repository branch:

```
branch.protect ()
branch.unprotect ()
```

Note: By default, developers are not authorized to push or merge into protected branches. This can be changed by passing `developers_can_push` or `developers_can_merge`:

```
branch.protect (developers_can_push=True, developers_can_merge=True)
```

Delete the merged branches for a project:

```
project.delete_merged_branches ()
```

6.5 Broadcast messages

You can use broadcast messages to display information on all pages of the gitlab web UI. You must have administration permissions to manipulate broadcast messages.

6.5.1 References

- v4 API:
 - `gitlab.v4.objects.BroadcastMessage`
 - `gitlab.v4.objects.BroadcastMessageManager`
 - `gitlab.Gitlab.broadcastmessages`
- GitLab API: https://docs.gitlab.com/ce/api/broadcast_messages.html

6.5.2 Examples

List the messages:

```
msgs = gl.broadcastmessages.list ()
```

Get a single message:

```
msg = gl.broadcastmessages.get (msg_id)
```

Create a message:

```
msg = gl.broadcastmessages.create ({'message': 'Important information'})
```

The date format for the `starts_at` and `ends_at` parameters is `YYYY-MM-ddThh:mm:ssZ`.

Update a message:

```
msg.font = '#444444'
msg.color = '#999999'
msg.save ()
```

Delete a message:

```
gl.broadcastmessages.delete(msg_id)
# or
msg.delete()
```

6.6 Commits

6.6.1 Commits

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectCommit`
 - `gitlab.v4.objects.ProjectCommitManager`
 - `gitlab.v4.objects.Project.commits`

Examples

List the commits for a project:

```
commits = project.commits.list()
```

You can use the `ref_name`, `since` and `until` filters to limit the results:

```
commits = project.commits.list(ref_name='my_branch')
commits = project.commits.list(since='2016-01-01T00:00:00Z')
```

Note: The available `all` listing argument conflicts with the `python-gitlab` argument. Use `query_parameters` to avoid the conflict:

```
commits = project.commits.list(all=True,
                               query_parameters={'ref_name': 'my_branch'})
```

Create a commit:

```
# See https://docs.gitlab.com/ce/api/commits.html#create-a-commit-with-multiple-files-
→and-actions
# for actions detail
data = {
    'branch': 'master',
    'commit_message': 'blah blah blah',
    'actions': [
        {
            'action': 'create',
            'file_path': 'README.rst',
            'content': open('path/to/file.rst').read(),
        },
        {
            # Binary files need to be base64 encoded
```

(continues on next page)

(continued from previous page)

```

        'action': 'create',
        'file_path': 'logo.png',
        'content': base64.b64encode(open('logo.png').read()),
        'encoding': 'base64',
    }
]
}

commit = project.commits.create(data)

```

Get a commit detail:

```
commit = project.commits.get('e3d5a71b')
```

Get the diff for a commit:

```
diff = commit.diff()
```

Cherry-pick a commit into another branch:

```
commit.cherry_pick(branch='target_branch')
```

Get the references the commit has been pushed to (branches and tags):

```

commit.refs() # all references
commit.refs('tag') # only tags
commit.refs('branch') # only branches

```

List the merge requests related to a commit:

```
commit.merge_requests()
```

6.6.2 Commit comments

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectCommitComment`
 - `gitlab.v4.objects.ProjectCommitCommentManager`
 - `gitlab.v4.objects.ProjectCommit.comments`
- GitLab API: <https://docs.gitlab.com/ce/api/commits.html>

Examples

Get the comments for a commit:

```
comments = commit.comments.list()
```

Add a comment on a commit:

```
# Global comment
commit = commit.comments.create({'note': 'This is a nice comment'})
# Comment on a line in a file (on the new version of the file)
commit = commit.comments.create({'note': 'This is another comment',
                                'line': 12,
                                'line_type': 'new',
                                'path': 'README.rst'})
```

6.6.3 Commit status

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectCommitStatus`
 - `gitlab.v4.objects.ProjectCommitStatusManager`
 - `gitlab.v4.objects.ProjectCommit.statuses`
- GitLab API: <https://docs.gitlab.com/ce/api/commits.html>

Examples

List the statuses for a commit:

```
statuses = commit.statuses.list()
```

Change the status of a commit:

```
commit.statuses.create({'state': 'success'})
```

6.7 Deploy keys

6.7.1 Deploy keys

Reference

- v4 API:
 - `gitlab.v4.objects.DeployKey`
 - `gitlab.v4.objects.DeployKeyManager`
 - `gitlab.Gitlab.deploykeys`
- GitLab API: https://docs.gitlab.com/ce/api/deploy_keys.html

Examples

List the deploy keys:


```
keys = gl.deploykeys.list()
```

6.7.2 Deploy keys for projects

Deploy keys can be managed on a per-project basis.

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectKey`
 - `gitlab.v4.objects.ProjectKeyManager`
 - `gitlab.v4.objects.Project.keys`
- GitLab API: https://docs.gitlab.com/ce/api/deploy_keys.html

Examples

List keys for a project:

```
keys = project.keys.list()
```

Get a single deploy key:

```
key = project.keys.get(key_id)
```

Create a deploy key for a project:

```
key = project.keys.create({'title': 'jenkins key',
                          'key': open('/home/me/.ssh/id_rsa.pub').read()})
```

Delete a deploy key for a project:

```
key = project.keys.list(key_id)
# or
key.delete()
```

Enable a deploy key for a project:

```
project.keys.enable(key_id)
```

Disable a deploy key for a project:

```
project_key.delete()
```

6.8 Deployments

6.8.1 Reference

- v4 API:

- `gitlab.v4.objects.ProjectDeployment`
- `gitlab.v4.objects.ProjectDeploymentManager`
- `gitlab.v4.objects.Project.deployments`

- GitLab API: <https://docs.gitlab.com/ce/api/deployments.html>

6.8.2 Examples

List deployments for a project:

```
deployments = project.deployments.list()
```

Get a single deployment:

```
deployment = project.deployments.get(deployment_id)
```

6.9 Discussions

Discussions organize the notes in threads. See the *Notes* chapter for more information about notes.

Discussions are available for project issues, merge requests, snippets and commits.

6.9.1 Reference

- v4 API:

Issues:

- `gitlab.v4.objects.ProjectIssueDiscussion`
- `gitlab.v4.objects.ProjectIssueDiscussionManager`
- `gitlab.v4.objects.ProjectIssueDiscussionNote`
- `gitlab.v4.objects.ProjectIssueDiscussionNoteManager`
- `gitlab.v4.objects.ProjectIssue.notes`

MergeRequests:

- `gitlab.v4.objects.ProjectMergeRequestDiscussion`
- `gitlab.v4.objects.ProjectMergeRequestDiscussionManager`
- `gitlab.v4.objects.ProjectMergeRequestDiscussionNote`
- `gitlab.v4.objects.ProjectMergeRequestDiscussionNoteManager`
- `gitlab.v4.objects.ProjectMergeRequest.notes`

Snippets:

- `gitlab.v4.objects.ProjectSnippetDiscussion`
- `gitlab.v4.objects.ProjectSnippetDiscussionManager`
- `gitlab.v4.objects.ProjectSnippetDiscussionNote`
- `gitlab.v4.objects.ProjectSnippetDiscussionNoteManager`

- `gitlab.v4.objects.ProjectSnippet.notes`

- GitLab API: <https://docs.gitlab.com/ce/api/discussions.html>

6.9.2 Examples

List the discussions for a resource (issue, merge request, snippet or commit):

```
discussions = resource.discussions.list()
```

Get a single discussion:

```
discussion = resource.discussions.get(discussion_id)
```

You can access the individual notes in the discussion through the `notes` attribute. It holds a list of notes in chronological order:

```
# ``resource.notes`` is a DiscussionNoteManager, so we need to get the
# object notes using ``attributes``
for note in discussion.attributes['notes']:
    print(note['body'])
```

Note: The notes are dicts, not objects.

You can add notes to existing discussions:

```
new_note = discussion.notes.create({'body': 'Episode IV: A new note'})
```

You can get and update a single note using the `*DiscussionNote` resources:

```
discussion = resource.discussions.get(discussion_id)
# Get the latest note's id
note_id = discussion.attributes['note'][-1]['id']
last_note = discussion.notes.get(note_id)
last_note.body = 'Updated comment'
last_note.save()
```

Create a new discussion:

```
discussion = resource.discussions.create({'body': 'First comment of discussion'})
```

You can comment on merge requests and commit diffs. Provide the `position` dict to define where the comment should appear in the diff:

```
mr_diff = mr.diffs.get(diff_id)
mr.dussions.create({'body': 'Note content',
                   'position': {
                       'base_sha': mr_diff.base_commit_sha,
                       'start_sha': mr_diff.start_commit_sha,
                       'head_sha': mr_diff.head_commit_sha,
                       'position_type': 'text',
                       'new_line': 1,
                       'old_path': 'README.rst',
                       'new_path': 'README.rst'
                   }})
```

Resolve / unresolve a merge request discussion:

```
mr_d = mr.discussions.get(d_id)
mr_d.resolved = True # True to resolve, False to unresolve
mr_d.save()
```

Delete a comment:

```
discussions.notes.delete(note_id)
# or
note.delete()
```

6.10 Environments

6.10.1 Reference

- v4 API:
 - `gitlab.v4.objects.ProjectEnvironment`
 - `gitlab.v4.objects.ProjectEnvironmentManager`
 - `gitlab.v4.objects.Project.environments`
- GitLab API: <https://docs.gitlab.com/ce/api/environments.html>

6.10.2 Examples

List environments for a project:

```
environments = project.environments.list()
```

Create an environment for a project:

```
environment = project.environments.create({'name': 'production'})
```

Retrieve a specific environment for a project:

```
environment = project.environments.get(112)
```

Update an environment for a project:

```
environment.external_url = 'http://foo.bar.com'
environment.save()
```

Delete an environment for a project:

```
environment = project.environments.delete(environment_id)
# or
environment.delete()
```

Stop an environments:

```
environment.stop()
```

6.11 Events

6.11.1 Reference

- v4 API:
 - `gitlab.v4.objects.Event`
 - `gitlab.v4.objects.EventManager`
 - `gitlab.Gitlab.events`
 - `gitlab.v4.objects.ProjectEvent`
 - `gitlab.v4.objects.ProjectEventManager`
 - `gitlab.v4.objects.Project.events`
 - `gitlab.v4.objects.UserEvent`
 - `gitlab.v4.objects.UserEventManager`
 - `gitlab.v4.objects.User.events`
- GitLab API: <https://docs.gitlab.com/ce/api/events.html>

6.11.2 Examples

You can list events for an entire Gitlab instance (admin), users and projects. You can filter you events you want to retrieve using the `action` and `target_type` attributes. The possible values for these attributes are available on the [gitlab documentation](#).

List all the events (paginated):

```
events = gl.events.list()
```

List the issue events on a project:

```
events = project.events.list(target_type='issue')
```

List the user events:

```
events = project.events.list()
```

6.12 Epics

6.12.1 Epics

Reference

- v4 API:
 - `gitlab.v4.objects.GroupEpic`
 - `gitlab.v4.objects.GroupEpicManager`
 - `gitlab.Gitlab.Group.epics`

- GitLab API: <https://docs.gitlab.com/ee/api/epics.html> (EE feature)

Examples

List the epics for a group:

```
epics = groups.epics.list()
```

Get a single epic for a group:

```
epic = group.epics.get(epic_iid)
```

Create an epic for a group:

```
epic = group.epics.create({'title': 'My Epic'})
```

Edit an epic:

```
epic.title = 'New title'  
epic.labels = ['label1', 'label2']  
epic.save()
```

Delete an epic:

```
epic.delete()
```

6.12.2 Epics issues

Reference

- v4 API:
 - `gitlab.v4.objects.GroupEpicIssue`
 - `gitlab.v4.objects.GroupEpicIssueManager`
 - `gitlab.Gitlab.GroupEpic.issues`
- GitLab API: https://docs.gitlab.com/ee/api/epic_issues.html (EE feature)

Examples

List the issues associated with an issue:

```
ei = epic.issues.list()
```

Associate an issue with an epic:

```
# use the issue id, not its iid  
ei = epic.issues.create({'issue_id': 4})
```

Move an issue in the list:

```
ei.move_before_id = epic_issue_id_1
# or
ei.move_after_id = epic_issue_id_2
ei.save()
```

Delete an issue association:

```
ei.delete()
```

6.13 Features flags

6.13.1 Reference

- v4 API:
 - `gitlab.v4.objects.Feature`
 - `gitlab.v4.objects.FeatureManager`
 - `gitlab.Gitlab.features`
- GitLab API: <https://docs.gitlab.com/ce/api/features.html>

6.13.2 Examples

List features:

```
features = gl.features.list()
```

Create or set a feature:

```
feature = gl.features.set(feature_name, True)
feature = gl.features.set(feature_name, 30)
```

Delete a feature:

```
feature.delete()
```

6.14 Geo nodes

6.14.1 Reference

- v4 API:
 - `gitlab.v4.objects.GeoNode`
 - `gitlab.v4.objects.GeoNodeManager`
 - `gitlab.Gitlab.geonodes`
- GitLab API: https://docs.gitlab.com/ee/api/geo_nodes.html (EE feature)

6.14.2 Examples

List the geo nodes:

```
nodes = gl.geonodes.list()
```

Get the status of all the nodes:

```
status = gl.geonodes.status()
```

Get a specific node and its status:

```
node = gl.geonodes.get(node_id)
node.status()
```

Edit a node configuration:

```
node.url = 'https://secondary.mygitlab.domain'
node.save()
```

Delete a node:

```
node.delete()
```

List the sync failure on the current node:

```
failures = gl.geonodes.current_failures()
```

6.15 Groups

6.15.1 Groups

Reference

- v4 API:
 - *gitlab.v4.objects.Group*
 - *gitlab.v4.objects.GroupManager*
 - `gitlab.Gitlab.groups`
- GitLab API: <https://docs.gitlab.com/ce/api/groups.html>

Examples

List the groups:

```
groups = gl.groups.list()
```

Get a group's detail:

```
group = gl.groups.get(group_id)
```

List a group's projects:


```
projects = group.projects.list()
```

Note: `GroupProject` objects returned by this API call are very limited, and do not provide all the features of `Project` objects. If you need to manipulate projects, create a new `Project` object:

```
first_group_project = group.projects.list()[0]
manageable_project = gl.projects.get(first_group_project.id, lazy=True)
```

You can filter and sort the result using the following parameters:

- `archived`: limit by archived status
- `visibility`: limit by visibility. Allowed values are `public`, `internal` and `private`
- `search`: limit to groups matching the given value
- `order_by`: sort by criteria. Allowed values are `id`, `name`, `path`, `created_at`, `updated_at` and `last_activity_at`
- `sort`: sort order: `asc` or `desc`
- `ci_enabled_first`: return CI enabled groups first

Create a group:

```
group = gl.groups.create({'name': 'group1', 'path': 'group1'})
```

Update a group:

```
group.description = 'My awesome group'
group.save()
```

Remove a group:

```
gl.groups.delete(group_id)
# or
group.delete()
```

6.15.2 Subgroups

Reference

- v4 API:
 - `gitlab.v4.objects.GroupSubgroup`
 - `gitlab.v4.objects.GroupSubgroupManager`
 - `gitlab.v4.objects.Group.subgroups`

Examples

List the subgroups for a group:

```
subgroups = group.subgroups.list()
```

Note: The `GroupSubgroup` objects don't expose the same API as the `Group` objects. If you need to manipulate a subgroup as a group, create a new `Group` object:

```
real_group = gl.groups.get(subgroup_id, lazy=True)
real_group.issues.list()
```

6.15.3 Group custom attributes

Reference

- v4 API:
 - `gitlab.v4.objects.GroupCustomAttribute`
 - `gitlab.v4.objects.GroupCustomAttributeManager`
 - `gitlab.v4.objects.Group.customattributes`
- GitLab API: https://docs.gitlab.com/ce/api/custom_attributes.html

Examples

List custom attributes for a group:

```
attrs = group.customattributes.list()
```

Get a custom attribute for a group:

```
attr = group.customattributes.get(attr_key)
```

Set (create or update) a custom attribute for a group:

```
attr = group.customattributes.set(attr_key, attr_value)
```

Delete a custom attribute for a group:

```
attr.delete()
# or
group.customattributes.delete(attr_key)
```

Search groups by custom attribute:

```
group.customattributes.set('role': 'admin')
gl.groups.list(custom_attributes={'role': 'admin'})
```

6.15.4 Group members

The following constants define the supported access levels:

- `gitlab.GUEST_ACCESS = 10`
- `gitlab.REPORTER_ACCESS = 20`
- `gitlab.DEVELOPER_ACCESS = 30`

- `gitlab.MAINTAINER_ACCESS = 40`
- `gitlab.OWNER_ACCESS = 50`

Reference

- v4 API:
 - `gitlab.v4.objects.GroupMember`
 - `gitlab.v4.objects.GroupMemberManager`
 - `gitlab.v4.objects.Group.members`
- GitLab API: <https://docs.gitlab.com/ce/api/groups.html>

Examples

List group members:

```
members = group.members.list()
```

List the group members recursively (including inherited members through ancestor groups):

```
members = group.members.all(all=True)
```

Get a group member:

```
members = group.members.get(member_id)
```

Add a member to the group:

```
member = group.members.create({'user_id': user_id,
                               'access_level': gitlab.GUEST_ACCESS})
```

Update a member (change the access level):

```
member.access_level = gitlab.DEVELOPER_ACCESS
member.save()
```

Remove a member from the group:

```
group.members.delete(member_id)
# or
member.delete()
```

6.15.5 LDAP group links

Add an LDAP group link to an existing GitLab group:

```
group.add_ldap_group_link(ldap_group_cn, gitlab.DEVELOPER_ACCESS, 'ldapmain')
```

Remove a link:

```
group.delete_ldap_group_link(ldap_group_cn, 'ldapmain')
```

Sync the LDAP groups:

```
group.ldap_sync()
```

You can use the `ldapgroups` manager to list available LDAP groups:

```
# listing (supports pagination)
ldap_groups = gl.ldapgroups.list()

# filter using a group name
ldap_groups = gl.ldapgroups.list(search='foo')

# list the groups for a specific LDAP provider
ldap_groups = gl.ldapgroups.list(search='foo', provider='ldapmain')
```

6.16 Issues

6.16.1 Reported issues

Reference

- v4 API:
 - `gitlab.v4.objects.Issue`
 - `gitlab.v4.objects.IssueManager`
 - `gitlab.Gitlab.issues`
- GitLab API: <https://docs.gitlab.com/ce/api/issues.html>

Examples

List the issues:

```
issues = gl.issues.list()
```

Use the `state` and `label` parameters to filter the results. Use the `order_by` and `sort` attributes to sort the results:

```
open_issues = gl.issues.list(state='opened')
closed_issues = gl.issues.list(state='closed')
tagged_issues = gl.issues.list(labels=['foo', 'bar'])
```

Note: It is not possible to edit or delete Issue objects. You need to create a `ProjectIssue` object to perform changes:

```
issue = gl.issues.list()[0]
project = gl.projects.get(issue.project_id, lazy=True)
editable_issue = project.issues.get(issue.iid, lazy=True)
editable_issue.title = updated_title
editable_issue.save()
```

6.16.2 Group issues

Reference

- v4 API:
 - `gitlab.v4.objects.GroupIssue`
 - `gitlab.v4.objects.GroupIssueManager`
 - `gitlab.v4.objects.Group.issues`
- GitLab API: <https://docs.gitlab.com/ce/api/issues.html>

Examples

List the group issues:

```
issues = group.issues.list()
# Filter using the state, labels and milestone parameters
issues = group.issues.list(milestone='1.0', state='opened')
# Order using the order_by and sort parameters
issues = group.issues.list(order_by='created_at', sort='desc')
```

Note: It is not possible to edit or delete `GroupIssue` objects. You need to create a `ProjectIssue` object to perform changes:

```
issue = group.issues.list()[0]
project = gl.projects.get(issue.project_id, lazy=True)
editable_issue = project.issues.get(issue.iid, lazy=True)
editable_issue.title = updated_title
editable_issue.save()
```

6.16.3 Project issues

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectIssue`
 - `gitlab.v4.objects.ProjectIssueManager`
 - `gitlab.v4.objects.Project.issues`
- GitLab API: <https://docs.gitlab.com/ce/api/issues.html>

Examples

List the project issues:

```
issues = project.issues.list()
# Filter using the state, labels and milestone parameters
issues = project.issues.list(milestone='1.0', state='opened')
# Order using the order_by and sort parameters
issues = project.issues.list(order_by='created_at', sort='desc')
```

Get a project issue:

```
issue = project.issues.get(issue_iid)
```

Create a new issue:

```
issue = project.issues.create({'title': 'I have a bug',
                              'description': 'Something useful here.'})
```

Update an issue:

```
issue.labels = ['foo', 'bar']
issue.save()
```

Close / reopen an issue:

```
# close an issue
issue.state_event = 'close'
issue.save()
# reopen it
issue.state_event = 'reopen'
issue.save()
```

Delete an issue:

```
project.issues.delete(issue_id)
# pr
issue.delete()
```

Subscribe / unsubscribe from an issue:

```
issue.subscribe()
issue.unsubscribe()
```

Move an issue to another project:

```
issue.move(other_project_id)
```

Make an issue as todo:

```
issue.todo()
```

Get time tracking stats:

```
issue.time_stats()
```

On recent versions of Gitlab the time stats are also returned as an issue object attribute:

```
issue = project.issue.get(iid)
print(issue.attributes['time_stats'])
```

Set a time estimate for an issue:

```
issue.time_estimate('3h30m')
```

Reset a time estimate for an issue:

```
issue.reset_time_estimate()
```

Add spent time for an issue:

```
issue.add_spent_time('3h30m')
```

Reset spent time for an issue:

```
issue.reset_spent_time()
```

Get user agent detail for the issue (admin only):

```
detail = issue.user_agent_detail()
```

Get the list of merge requests that will close an issue when merged:

```
mrs = issue.closed_by()
```

Get the merge requests related to an issue:

```
mrs = issue.related_merge_requests()
```

Get the list of participants:

```
users = issue.participants()
```

6.16.4 Issue links

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectIssueLink`
 - `gitlab.v4.objects.ProjectIssueLinkManager`
 - `gitlab.v4.objects.ProjectIssue.links`
- GitLab API: https://docs.gitlab.com/ee/api/issue_links.html (EE feature)

Examples

List the issues linked to `i1`:

```
links = i1.links.list()
```

Link issue `i1` to issue `i2`:

```
data = {
    'target_project_id': i2.project_id,
    'target_issue_iid': i2.iid
}
src_issue, dest_issue = i1.links.create(data)
```

Note: The `create()` method returns the source and destination `ProjectIssue` objects, not a `ProjectIssueLink` object.

Delete a link:

```
i1.links.delete(issue_link_id)
```

6.17 Issue boards

6.17.1 Boards

Boards are a visual representation of existing issues for a project or a group. Issues can be moved from one list to the other to track progress and help with priorities.

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectBoard`
 - `gitlab.v4.objects.ProjectBoardManager`
 - `gitlab.v4.objects.Project.boards`
 - `gitlab.v4.objects.GroupBoard`
 - `gitlab.v4.objects.GroupBoardManager`
 - `gitlab.v4.objects.Group.boards`
- GitLab API:
 - <https://docs.gitlab.com/ce/api/boards.html>
 - https://docs.gitlab.com/ce/api/group_boards.html

Examples

Get the list of existing boards for a project or a group:

```
# item is a Project or a Group
boards = project_or_group.boards.list()
```

Get a single board for a project or a group:

```
board = project_or_group.boards.get(board_id)
```

Create a board:


```
board = project_or_group.boards.create({'name': 'new-board'})
```

Note: Board creation is not supported in the GitLab CE edition.

Delete a board:

```
board.delete()  
# or  
project_or_group.boards.delete(board_id)
```

Note: Board deletion is not supported in the GitLab CE edition.

6.17.2 Board lists

Boards are made of lists of issues. Each list is associated to a label, and issues tagged with this label automatically belong to the list.

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectBoardList`
 - `gitlab.v4.objects.ProjectBoardListManager`
 - `gitlab.v4.objects.ProjectBoard.lists`
 - `gitlab.v4.objects.GroupBoardList`
 - `gitlab.v4.objects.GroupBoardListManager`
 - `gitlab.v4.objects.GroupBoard.lists`
- GitLab API:
 - <https://docs.gitlab.com/ce/api/boards.html>
 - https://docs.gitlab.com/ce/api/group_boards.html

Examples

List the issue lists for a board:

```
b_lists = board.lists.list()
```

Get a single list:

```
b_list = board.lists.get(list_id)
```

Create a new list:

```
# First get a ProjectLabel
label = get_or_create_label()
# Then use its ID to create the new board list
b_list = board.lists.create({'label_id': label.id})
```

Change a list position. The first list is at position 0. Moving a list will set it at the given position and move the following lists up a position:

```
b_list.position = 2
b_list.save()
```

Delete a list:

```
b_list.delete()
```

6.18 Labels

6.18.1 Project labels

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectLabel`
 - `gitlab.v4.objects.ProjectLabelManager`
 - `gitlab.v4.objects.Project.labels`
- GitLab API: <https://docs.gitlab.com/ce/api/labels.html>

Examples

List labels for a project:

```
labels = project.labels.list()
```

Create a label for a project:

```
label = project.labels.create({'name': 'foo', 'color': '#8899aa'})
```

Update a label for a project:

```
# change the name of the label:
label.new_name = 'bar'
label.save()
# change its color:
label.color = '#112233'
label.save()
```

Delete a label for a project:

```
project.labels.delete(label_id)
# or
label.delete()
```

Manage labels in issues and merge requests:

```
# Labels are defined as lists in issues and merge requests. The labels must
# exist.
issue = p.issues.create({'title': 'issue title',
                        'description': 'issue description',
                        'labels': ['foo']})
issue.labels.append('bar')
issue.save()
```

6.18.2 Label events

Resource label events keep track about who, when, and which label was added or removed to an issuable.

Group epic label events are only available in the EE edition.

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectIssueResourceLabelEvent`
 - `gitlab.v4.objects.ProjectIssueResourceLabelEventManager`
 - `gitlab.v4.objects.ProjectIssue.resourcelabelevents`
 - `gitlab.v4.objects.ProjectMergeRequestResourceLabelEvent`
 - `gitlab.v4.objects.ProjectMergeRequestResourceLabelEventManager`
 - `gitlab.v4.objects.ProjectMergeRequest.resourcelabelevents`
 - `gitlab.v4.objects.GroupEpicResourceLabelEvent`
 - `gitlab.v4.objects.GroupEpicResourceLabelEventManager`
 - `gitlab.v4.objects.GroupEpic.resourcelabelevents`
- GitLab API: https://docs.gitlab.com/ee/api/resource_label_events.html

Examples

Get the events for a resource (issue, merge request or epic):

```
events = resource.resourcelabelevents.list()
```

Get a specific event for a resource:

```
event = resource.resourcelabelevents.get(event_id)
```

6.19 Notification settings

You can define notification settings globally, for groups and for projects. Valid levels are defined as constants:

- `gitlab.NOTIFICATION_LEVEL_DISABLED`
- `gitlab.NOTIFICATION_LEVEL_PARTICIPATING`

- `gitlab.NOTIFICATION_LEVEL_WATCH`
- `gitlab.NOTIFICATION_LEVEL_GLOBAL`
- `gitlab.NOTIFICATION_LEVEL_MENTION`
- `gitlab.NOTIFICATION_LEVEL_CUSTOM`

You get access to fine-grained settings if you use the `NOTIFICATION_LEVEL_CUSTOM` level.

6.19.1 Reference

- v4 API:
 - `gitlab.v4.objects.NotificationSettings`
 - `gitlab.v4.objects.NotificationSettingsManager`
 - `gitlab.Gitlab.notificationsettings`
 - `gitlab.v4.objects.GroupNotificationSettings`
 - `gitlab.v4.objects.GroupNotificationSettingsManager`
 - `gitlab.v4.objects.Group.notificationsettings`
 - `gitlab.v4.objects.ProjectNotificationSettings`
 - `gitlab.v4.objects.ProjectNotificationSettingsManager`
 - `gitlab.v4.objects.Project.notificationsettings`
- GitLab API: https://docs.gitlab.com/ce/api/notification_settings.html

6.19.2 Examples

Get the notifications settings:

```
# global settings
settings = gl.notificationsettings.get ()
# for a group
settings = gl.groups.get (group_id) .notificationsettings.get ()
# for a project
settings = gl.projects.get (project_id) .notificationsettings.get ()
```

Update the notifications settings:

```
# use a predefined level
settings.level = gitlab.NOTIFICATION_LEVEL_WATCH

# create a custom setup
settings.level = gitlab.NOTIFICATION_LEVEL_CUSTOM
settings.save () # will create additional attributes, but not mandatory

settings.new_merge_request = True
settings.new_issue = True
settings.new_note = True
settings.save ()
```

6.20 Merge requests

You can use merge requests to notify a project that a branch is ready for merging. The owner of the target project can accept the merge request.

Merge requests are linked to projects, but they can be listed globally or for groups.

6.20.1 Group and global listing

Reference

- v4 API:
 - `gitlab.v4.objects.GroupMergeRequest`
 - `gitlab.v4.objects.GroupMergeRequestManager`
 - `gitlab.v4.objects.Group.mergerequests`
 - `gitlab.v4.objects.MergeRequest`
 - `gitlab.v4.objects.MergeRequestManager`
 - `gitlab.Gtilab.mergerequests`
- GitLab API: https://docs.gitlab.com/ce/api/merge_requests.html

Examples

List the merge requests available on the GitLab server:

```
mrs = gl.mergerequests.list()
```

List the merge requests for a group:

```
group = gl.groups.get('mygroup')
mrs = group.mergerequests.list()
```

Note: It is not possible to edit or delete `MergeRequest` and `GroupMergeRequest` objects. You need to create a `ProjectMergeRequest` object to apply changes:

```
mr = group.mergerequests.list()[0]
project = gl.projects.get(mr.project_id, lazy=True)
editable_mr = project.mergerequests.get(mr.iid, lazy=True)
editable_mr.title = updated_title
editable_mr.save()
```

6.20.2 Project merge requests

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectMergeRequest`

- `gitlab.v4.objects.ProjectMergeRequestManager`
- `gitlab.v4.objects.Project.mergerequests`
- GitLab API: https://docs.gitlab.com/ce/api/merge_requests.html

Examples

List MRs for a project:

```
mrs = project.mergerequests.list()
```

You can filter and sort the returned list with the following parameters:

- `state`: state of the MR. It can be one of all, merged, opened or closed
- `order_by`: sort by `created_at` or `updated_at`
- `sort`: sort order (asc or desc)

For example:

```
mrs = project.mergerequests.list(state='merged', order_by='updated_at')
```

Get a single MR:

```
mr = project.mergerequests.get(mr_id)
```

Create a MR:

```
mr = project.mergerequests.create({'source_branch': 'cool_feature',  
                                  'target_branch': 'master',  
                                  'title': 'merge cool feature',  
                                  'labels': ['label1', 'label2']})
```

Update a MR:

```
mr.description = 'New description'  
mr.labels = ['foo', 'bar']  
mr.save()
```

Change the state of a MR (close or reopen):

```
mr.state_event = 'close' # or 'reopen'  
mr.save()
```

Delete a MR:

```
project.mergerequests.delete(mr_id)  
# or  
mr.delete()
```

Accept a MR:

```
mr.merge()
```

Cancel a MR when the build succeeds:

```
mr.cancel_merge_when_pipeline_succeeds()
```

List commits of a MR:

```
commits = mr.commits()
```

List the changes of a MR:

```
changes = mr.changes()
```

List the pipelines for a MR:

```
pipelines = mr.pipelines()
```

List issues that will close on merge:

```
mr.closes_issues()
```

Subscribe to / unsubscribe from a MR:

```
mr.subscribe()  
mr.unsubscribe()
```

Mark a MR as todo:

```
mr.todo()
```

List the diffs for a merge request:

```
diffs = mr.diffs.list()
```

Get a diff for a merge request:

```
diff = mr.diffs.get(diff_id)
```

Get time tracking stats:

```
merge_request.time_stats()
```

On recent versions of Gitlab the time stats are also returned as a merge request object attribute:

```
mr = project.mergerequests.get(id)  
print(mr.attributes['time_stats'])
```

Set a time estimate for a merge request:

```
mr.time_estimate('3h30m')
```

Reset a time estimate for a merge request:

```
mr.reset_time_estimate()
```

Add spent time for a merge request:

```
mr.add_spent_time('3h30m')
```

Reset spent time for a merge request:

```
mr.reset_spent_time()
```

Get user agent detail for the issue (admin only):

```
detail = issue.user_agent_detail()
```

Attempt to rebase an MR:

```
mr.rebase()
```

6.21 Merge request approvals settings

Merge request approvals can be defined at the project level or at the merge request level.

6.21.1 References

- v4 API:
 - `gitlab.v4.objects.ProjectApproval`
 - `gitlab.v4.objects.ProjectApprovalManager`
 - `gitlab.v4.objects.Project.approvals`
 - `gitlab.v4.objects.ProjectMergeRequestApproval`
 - `gitlab.v4.objects.ProjectMergeRequestApprovalManager`
 - `gitlab.v4.objects.ProjectMergeRequest.approvals`
- GitLab API: https://docs.gitlab.com/ee/api/merge_request_approvals.html

6.21.2 Examples

Get project-level or MR-level MR approvals settings:

```
p_mras = project.approvals.get()
mr_mras = mr.approvals.get()
```

Change project-level or MR-level MR approvals settings:

```
p_mras.approvals_before_merge = 2
p_mras.save()

mr_mras.approvals_before_merge = 2
mr_mras.save()
```

Change project-level or MR-level MR allowed approvers:

```
project.approvals.set_approvers(approver_ids=[105],
                                approver_group_ids=[653, 654])

mr.approvals.set_approvers(approver_ids=[105],
                            approver_group_ids=[653, 654])
```


6.22 Milestones

6.22.1 Reference

- v4 API:
 - `gitlab.v4.objects.ProjectMilestone`
 - `gitlab.v4.objects.ProjectMilestoneManager`
 - `gitlab.v4.objects.Project.milestones`
 - `gitlab.v4.objects.GroupMilestone`
 - `gitlab.v4.objects.GroupMilestoneManager`
 - `gitlab.v4.objects.Group.milestones`
- GitLab API:
 - <https://docs.gitlab.com/ce/api/milestones.html>
 - https://docs.gitlab.com/ce/api/group_milestones.html

6.22.2 Examples

List the milestones for a project or a group:

```
p_milestones = project.milestones.list()
g_milestones = group.milestones.list()
```

You can filter the list using the following parameters:

- `iids`: unique IDs of milestones for the project
- `state`: either `active` or `closed`
- `search`: to search using a string

```
p_milestones = project.milestones.list(state='closed')
g_milestones = group.milestones.list(state='active')
```

Get a single milestone:

```
p_milestone = project.milestones.get(milestone_id)
g_milestone = group.milestones.get(milestone_id)
```

Create a milestone:

```
milestone = project.milestones.create({'title': '1.0'})
```

Edit a milestone:

```
milestone.description = 'v 1.0 release'
milestone.save()
```

Change the state of a milestone (activate / close):

```
# close a milestone
milestone.state_event = 'close'
milestone.save()

# activate a milestone
milestone.state_event = 'activate'
milestone.save()
```

List the issues related to a milestone:

```
issues = milestone.issues()
```

List the merge requests related to a milestone:

```
merge_requests = milestone.merge_requests()
```

6.23 Namespaces

6.23.1 Reference

- v4 API:
 - *gitlab.v4.objects.Namespace*
 - *gitlab.v4.objects.NamespaceManager*
 - `gitlab.Gitlab.namespaces`
- GitLab API: <https://docs.gitlab.com/ce/api/namespaces.html>

6.23.2 Examples

List namespaces:

```
namespaces = gl.namespaces.list()
```

Search namespaces:

```
namespaces = gl.namespaces.list(search='foo')
```

6.24 Notes

You can manipulate notes (comments) on project issues, merge requests and snippets.

6.24.1 Reference

- v4 API:
 - Issues:
 - *gitlab.v4.objects.ProjectIssueNote*

- `gitlab.v4.objects.ProjectIssueNoteManager`
- `gitlab.v4.objects.ProjectIssue.notes`

MergeRequests:

- `gitlab.v4.objects.ProjectMergeRequestNote`
- `gitlab.v4.objects.ProjectMergeRequestNoteManager`
- `gitlab.v4.objects.ProjectMergeRequest.notes`

Snippets:

- `gitlab.v4.objects.ProjectSnippetNote`
- `gitlab.v4.objects.ProjectSnippetNoteManager`
- `gitlab.v4.objects.ProjectSnippet.notes`

- GitLab API: <https://docs.gitlab.com/ce/api/notes.html>

6.24.2 Examples

List the notes for a resource:

```
i_notes = issue.notes.list()
mr_notes = mr.notes.list()
s_notes = snippet.notes.list()
```

Get a note for a resource:

```
i_note = issue.notes.get(note_id)
mr_note = mr.notes.get(note_id)
s_note = snippet.notes.get(note_id)
```

Create a note for a resource:

```
i_note = issue.notes.create({'body': 'note content'})
mr_note = mr.notes.create({'body': 'note content'})
s_note = snippet.notes.create({'body': 'note content'})
```

Update a note for a resource:

```
note.body = 'updated note content'
note.save()
```

Delete a note for a resource:

```
note.delete()
```

6.25 Pages domains

6.25.1 Admin

References

- v4 API:

- `gitlab.v4.objects.PagesDomain`
- `gitlab.v4.objects.PagesDomainManager`
- `gitlab.Gitlab.pagesdomains`

- GitLab API: https://docs.gitlab.com/ce/api/pages_domains.html#list-all-pages-domains

Examples

List all the existing domains (admin only):

```
domains = gl.pagesdomains.list()
```

6.25.2 Project pages domain

References

- v4 API:
 - `gitlab.v4.objects.ProjectPagesDomain`
 - `gitlab.v4.objects.ProjectPagesDomainManager`
 - `gitlab.v4.objects.Project.pagesdomains`
- GitLab API: https://docs.gitlab.com/ce/api/pages_domains.html#list-pages-domains

Examples

List domains for a project:

```
domains = project.pagesdomains.list()
```

Get a single domain:

```
domain = project.pagesdomains.get('d1.example.com')
```

Create a new domain:

```
domain = project.pagesdomains.create({'domain': 'd2.example.com'})
```

Update an existing domain:

```
domain.certificate = open('d2.crt').read()
domain.key = open('d2.key').read()
domain.save()
```

Delete an existing domain:

```
domain.delete
# or
project.pagesdomains.delete('d2.example.com')
```

6.26 Pipelines and Jobs

6.26.1 Project pipelines

A pipeline is a group of jobs executed by GitLab CI.

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectPipeline`
 - `gitlab.v4.objects.ProjectPipelineManager`
 - `gitlab.v4.objects.Project.pipelines`
- GitLab API: <https://docs.gitlab.com/ce/api/pipelines.html>

Examples

List pipelines for a project:

```
pipelines = project.pipelines.list()
```

Get a pipeline for a project:

```
pipeline = project.pipelines.get(pipeline_id)
```

Get variables of a pipeline:

```
variables = pipeline.variables.list()
```

Create a pipeline for a particular reference:

```
pipeline = project.pipelines.create({'ref': 'master'})
```

Retry the failed builds for a pipeline:

```
pipeline.retry()
```

Cancel builds in a pipeline:

```
pipeline.cancel()
```

Delete a pipeline:

```
pipeline.delete()
```

6.26.2 Triggers

Triggers provide a way to interact with the GitLab CI. Using a trigger a user or an application can run a new build/job for a specific commit.

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectTrigger`
 - `gitlab.v4.objects.ProjectTriggerManager`
 - `gitlab.v4.objects.Project.triggers`
- GitLab API: https://docs.gitlab.com/ce/api/pipeline_triggers.html

Examples

List triggers:

```
triggers = project.triggers.list()
```

Get a trigger:

```
trigger = project.triggers.get(trigger_token)
```

Create a trigger:

```
trigger = project.triggers.create({'description': 'mytrigger'})
```

Remove a trigger:

```
project.triggers.delete(trigger_token)
# or
trigger.delete()
```

Full example with wait for finish:

```
def get_or_create_trigger(project):
    trigger_decription = 'my_trigger_id'
    for t in project.triggers.list():
        if t.description == trigger_decription:
            return t
    return project.triggers.create({'description': trigger_decription})

trigger = get_or_create_trigger(project)
pipeline = project.trigger_pipeline('master', trigger.token, variables={"DEPLOY_ZONE": "us-west1"})
while pipeline.finished_at is None:
    pipeline.refresh()
    time.sleep(1)
```

You can trigger a pipeline using token authentication instead of user authentication. To do so create an anonymous Gitlab instance and use lazy objects to get the associated project:

```
gl = gitlab.Gitlab(URL) # no authentication
project = gl.projects.get(project_id, lazy=True) # no API call
project.trigger_pipeline('master', trigger_token)
```

Reference: <https://docs.gitlab.com/ee/ci/triggers/#trigger-token>

6.26.3 Pipeline schedule

You can schedule pipeline runs using a cron-like syntax. Variables can be associated with the scheduled pipelines.

Reference

- v4 API
 - `gitlab.v4.objects.ProjectPipelineSchedule`
 - `gitlab.v4.objects.ProjectPipelineScheduleManager`
 - `gitlab.v4.objects.Project.pipelineschedules`
 - `gitlab.v4.objects.ProjectPipelineScheduleVariable`
 - `gitlab.v4.objects.ProjectPipelineScheduleVariableManager`
 - `gitlab.v4.objects.Project.pipelineschedules`
- GitLab API: https://docs.gitlab.com/ce/api/pipeline_schedules.html

Examples

List pipeline schedules:

```
scheds = project.pipelineschedules.list()
```

Get a single schedule:

```
sched = projects.pipelineschedules.get(schedule_id)
```

Create a new schedule:

```
sched = project.pipelineschedules.create({
    'ref': 'master',
    'description': 'Daily test',
    'cron': '0 1 * * *'})
```

Update a schedule:

```
sched.cron = '1 2 * * *'
sched.save()
```

Delete a schedule:

```
sched.delete()
```

List schedule variables:

```
# note: you need to use get() to retrieve the schedule variables. The
# attribute is not present in the response of a list() call
sched = projects.pipelineschedules.get(schedule_id)
vars = sched.attributes['variables']
```

Create a schedule variable:

```
var = sched.variables.create({'key': 'foo', 'value': 'bar'})
```

Edit a schedule variable:

```
var.value = 'new_value'  
var.save()
```

Delete a schedule variable:

```
var.delete()
```

6.26.4 Projects and groups variables

You can associate variables to projects and groups to modify the build/job scripts behavior.

Reference

- v4 API
 - *gitlab.v4.objects.ProjectVariable*
 - *gitlab.v4.objects.ProjectVariableManager*
 - `gitlab.v4.objects.Project.variables`
 - *gitlab.v4.objects.GroupVariable*
 - *gitlab.v4.objects.GroupVariableManager*
 - `gitlab.v4.objects.Group.variables`
- GitLab API
 - https://docs.gitlab.com/ce/api/project_level_variables.html
 - https://docs.gitlab.com/ce/api/group_level_variables.html

Examples

List variables:

```
p_variables = project.variables.list()  
g_variables = group.variables.list()
```

Get a variable:

```
p_var = project.variables.get('key_name')  
g_var = group.variables.get('key_name')
```

Create a variable:

```
var = project.variables.create({'key': 'key1', 'value': 'value1'})  
var = group.variables.create({'key': 'key1', 'value': 'value1'})
```

Update a variable value:


```
var.value = 'new_value'
var.save()
```

Remove a variable:

```
project.variables.delete('key_name')
group.variables.delete('key_name')
# or
var.delete()
```

6.26.5 Jobs

Jobs are associated to projects, pipelines and commits. They provide information on the jobs that have been run, and methods to manipulate them.

Reference

- v4 API
 - `gitlab.v4.objects.ProjectJob`
 - `gitlab.v4.objects.ProjectJobManager`
 - `gitlab.v4.objects.Project.jobs`
- GitLab API: <https://docs.gitlab.com/ce/api/jobs.html>

Examples

Jobs are usually automatically triggered, but you can explicitly trigger a new job:

```
project.trigger_build('master', trigger_token,
                     {'extra_var1': 'foo', 'extra_var2': 'bar'})
```

List jobs for the project:

```
jobs = project.jobs.list()
```

Get a single job:

```
project.jobs.get(job_id)
```

List the jobs of a pipeline:

```
project = gl.projects.get(project_id)
pipeline = project.pipelines.get(pipeline_id)
jobs = pipeline.jobs.list()
```

Note: Job methods (play, cancel, and so on) are not available on `ProjectPipelineJob` objects. To use these methods create a `ProjectJob` object:

```
pipeline_job = pipeline.jobs.list()[0]
job = project.jobs.get(pipeline_job.id, lazy=True)
job.retry()
```

Get the artifacts of a job:

```
build_or_job.artifacts()
```

Warning: Artifacts are entirely stored in memory in this example.

You can download artifacts as a stream. Provide a callable to handle the stream:

```
class Foo(object):
    def __init__(self):
        self._fd = open('artifacts.zip', 'wb')

    def __call__(self, chunk):
        self._fd.write(chunk)

target = Foo()
build_or_job.artifacts(streamed=True, action=target)
del(target) # flushes data on disk
```

You can also directly stream the output into a file, and unzip it afterwards:

```
zipfn = "__artifacts.zip"
with open(zipfn, "wb") as f:
    build_or_job.artifacts(streamed=True, action=f.write)
subprocess.run(["unzip", "-bo", zipfn])
os.unlink(zipfn)
```

Get a single artifact file:

```
build_or_job.artifact('path/to/file')
```

Get a single artifact file by branch and job:

```
project.artifact('branch', 'path/to/file', 'job')
```

Mark a job artifact as kept when expiration is set:

```
build_or_job.keep_artifacts()
```

Delete the artifacts of a job:

```
build_or_job.delete_artifacts()
```

Get a job trace:

```
build_or_job.trace()
```

Warning: Traces are entirely stored in memory unless you use the streaming feature. See *the artifacts example*.

Cancel/retry a job:

```
build_or_job.cancel()
build_or_job.retry()
```

Play (trigger) a job:

```
build_or_job.play()
```

Erase a job (artifacts and trace):

```
build_or_job.erase()
```

6.27 Projects

6.27.1 Projects

Reference

- v4 API:
 - `gitlab.v4.objects.Project`
 - `gitlab.v4.objects.ProjectManager`
 - `gitlab.Gitlab.projects`
- GitLab API: <https://docs.gitlab.com/ce/api/projects.html>

Examples

List projects:

```
projects = gl.projects.list()
```

The API provides several filtering parameters for the listing methods:

- `archived`: if `True` only archived projects will be returned
- `visibility`: returns only projects with the specified visibility (can be `public`, `internal` or `private`)
- `search`: returns project matching the given pattern

Results can also be sorted using the following parameters:

- `order_by`: sort using the given argument. Valid values are `id`, `name`, `path`, `created_at`, `updated_at` and `last_activity_at`. The default is to sort by `created_at`
- `sort`: sort order (`asc` or `desc`)

```
# List all projects (default 20)
projects = gl.projects.list(all=True)
# Archived projects
projects = gl.projects.list(archived=1)
# Limit to projects with a defined visibility
projects = gl.projects.list(visibility='public')

# List owned projects
```

(continues on next page)

(continued from previous page)

```
projects = gl.projects.list(owned=True)

# List starred projects
projects = gl.projects.list(starred=True)

# Search projects
projects = gl.projects.list(search='keyword')
```

Get a single project:

```
# Get a project by ID
project = gl.projects.get(project_id)
# Get a project by userspace/name
project = gl.projects.get('myteam/myproject')
```

Create a project:

```
project = gl.projects.create({'name': 'project1'})
```

Create a project for a user (admin only):

```
alice = gl.users.list(username='alice')[0]
user_project = alice.projects.create({'name': 'project'})
user_projects = alice.projects.list()
```

Create a project in a group:

```
# You need to get the id of the group, then use the namespace_id attribute
# to create the group
group_id = gl.groups.search('my-group')[0].id
project = gl.projects.create({'name': 'myrepo', 'namespace_id': group_id})
```

Update a project:

```
project.snippets_enabled = 1
project.save()
```

Set the avatar image for a project:

```
# the avatar image can be passed as data (content of the file) or as a file
# object opened in binary mode
project.avatar = open('path/to/file.png', 'rb')
project.save()
```

Delete a project:

```
gl.projects.delete(project_id)
# or
project.delete()
```

Fork a project:

```
fork = project.forks.create({})

# fork to a specific namespace
fork = project.forks.create({'namespace': 'myteam'})
```

Get a list of forks for the project:

```
forks = project.forks.list()
```

Create/delete a fork relation between projects (requires admin permissions):

```
project.create_fork_relation(source_project.id)
project.delete_fork_relation()
```

Get languages used in the project with percentage value:

```
languages = project.languages()
```

Star/unstar a project:

```
project.star()
project.unstar()
```

Archive/unarchive a project:

```
project.archive()
project.unarchive()
```

Start the housekeeping job:

```
project.housekeeping()
```

List the repository tree:

```
# list the content of the root directory for the default branch
items = project.repository_tree()

# list the content of a subdirectory on a specific branch
items = project.repository_tree(path='docs', ref='branch1')
```

Get the content and metadata of a file for a commit, using a blob sha:

```
items = project.repository_tree(path='docs', ref='branch1')
file_info = p.repository_blob(items[0]['id'])
content = base64.b64decode(file_info['content'])
size = file_info['size']
```

Get the repository archive:

```
tgz = project.repository_archive()

# get the archive for a branch/tag/commit
tgz = project.repository_archive(sha='4567abc')
```

Warning: Archives are entirely stored in memory unless you use the streaming feature. See *the artifacts example*.

Get the content of a file using the blob id:

```
# find the id for the blob (simple search)
id = [d['id'] for d in p.repository_tree() if d['name'] == 'README.rst'][0]
```

(continues on next page)

(continued from previous page)

```
# get the content
file_content = p.repository_raw_blob(id)
```

Warning: Blobs are entirely stored in memory unless you use the streaming feature. See *the artifacts example*.

Get a snapshot of the repository:

```
tar_file = project.snapshot()
```

Warning: Snapshots are entirely stored in memory unless you use the streaming feature. See *the artifacts example*.

Compare two branches, tags or commits:

```
result = project.repository_compare('master', 'branch1')

# get the commits
for commit in result['commits']:
    print(commit)

# get the diffs
for file_diff in result['diffs']:
    print(file_diff)
```

Get a list of contributors for the repository:

```
contributors = project.repository_contributors()
```

Get a list of users for the repository:

```
users = p.users.list()

# search for users
users = p.users.list(search='pattern')
```

Start the pull mirroring process (EE edition):

```
project.mirror_pull()
```

6.27.2 Import / Export

You can export projects from gitlab, and re-import them to create new projects or overwrite existing ones.

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectExport`
 - `gitlab.v4.objects.ProjectExportManager`

- `gitlab.v4.objects.Project.exports`
- `gitlab.v4.objects.ProjectImport`
- `gitlab.v4.objects.ProjectImportManager`
- `gitlab.v4.objects.Project.imports`
- `gitlab.v4.objects.ProjectManager.import_project`

- GitLab API: https://docs.gitlab.com/ce/api/project_import_export.html

Examples

A project export is an asynchronous operation. To retrieve the archive generated by GitLab you need to:

1. Create an export using the API
2. Wait for the export to be done
3. Download the result

```
# Create the export
p = gl.projects.get(my_project)
export = p.exports.create({})

# Wait for the 'finished' status
export.refresh()
while export.export_status != 'finished':
    time.sleep(1)
    export.refresh()

# Download the result
with open('/tmp/export.tgz', 'wb') as f:
    export.download(streamed=True, action=f.write)
```

Import the project:

```
output = gl.projects.import_project(open('/tmp/export.tgz', 'rb'), 'my_new_project')
# Get a ProjectImport object to track the import status
project_import = gl.projects.get(output['id'], lazy=True).imports.get()
while project_import.import_status != 'finished':
    time.sleep(1)
    project_import.refresh()
```

6.27.3 Project custom attributes

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectCustomAttribute`
 - `gitlab.v4.objects.ProjectCustomAttributeManager`
 - `gitlab.v4.objects.Project.customattributes`
- GitLab API: https://docs.gitlab.com/ce/api/custom_attributes.html

Examples

List custom attributes for a project:

```
attrs = project.customattributes.list()
```

Get a custom attribute for a project:

```
attr = project.customattributes.get(attr_key)
```

Set (create or update) a custom attribute for a project:

```
attr = project.customattributes.set(attr_key, attr_value)
```

Delete a custom attribute for a project:

```
attr.delete()  
# or  
project.customattributes.delete(attr_key)
```

Search projects by custom attribute:

```
project.customattributes.set('type', 'internal')  
gl.projects.list(custom_attributes={'type': 'internal'})
```

6.27.4 Project files

Reference

- v4 API:
 - *gitlab.v4.objects.ProjectFile*
 - *gitlab.v4.objects.ProjectFileManager*
 - *gitlab.v4.objects.Project.files*
- GitLab API: https://docs.gitlab.com/ce/api/repository_files.html

Examples

Get a file:

```
f = project.files.get(file_path='README.rst', ref='master')  
  
# get the base64 encoded content  
print(f.content)  
  
# get the decoded content  
print(f.decode())
```

Create a new file:


```
f = project.files.create({'file_path': 'testfile.txt',
                          'branch': 'master',
                          'content': file_content,
                          'author_email': 'test@example.com',
                          'author_name': 'yourname',
                          'encoding': 'text',
                          'commit_message': 'Create testfile'})
```

Update a file. The entire content must be uploaded, as plain text or as base64 encoded text:

```
f.content = 'new content'
f.save(branch='master', commit_message='Update testfile')

# or for binary data
# Note: decode() is required with python 3 for data serialization. You can omit
# it with python 2
f.content = base64.b64encode(open('image.png').read()).decode()
f.save(branch='master', commit_message='Update testfile', encoding='base64')
```

Delete a file:

```
f.delete(commit_message='Delete testfile')
```

6.27.5 Project tags

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectTag`
 - `gitlab.v4.objects.ProjectTagManager`
 - `gitlab.v4.objects.Project.tags`
- GitLab API: <https://docs.gitlab.com/ce/api/tags.html>

Examples

List the project tags:

```
tags = project.tags.list()
```

Get a tag:

```
tag = project.tags.get('1.0')
```

Create a tag:

```
tag = project.tags.create({'tag_name': '1.0', 'ref': 'master'})
```

Set or update the release note for a tag:

```
tag.set_release_description('awesome v1.0 release')
```

Delete a tag:

```
project.tags.delete('1.0')
# or
tag.delete()
```

6.27.6 Project snippets

The snippet visibility can be defined using the following constants:

- `gitlab.VISIBILITY_PRIVATE`
- `gitlab.VISIBILITY_INTERNAL`
- `gitlab.VISIBILITY_PUBLIC`

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectSnippet`
 - `gitlab.v4.objects.ProjectSnippetManager`
 - `gitlab.v4.objects.Project.files`
- GitLab API: https://docs.gitlab.com/ce/api/project_snippets.html

Examples

List the project snippets:

```
snippets = project.snippets.list()
```

Get a snippet:

```
snippets = project.snippets.list(snippet_id)
```

Get the content of a snippet:

```
print(snippet.content())
```

Warning: The snippet content is entirely stored in memory unless you use the streaming feature. See *the artifacts example*.

Create a snippet:

```
snippet = project.snippets.create({'title': 'sample 1',
                                  'file_name': 'foo.py',
                                  'code': 'import gitlab',
                                  'visibility_level':
gitlab.VISIBILITY_PRIVATE})
```

Update a snippet:

```
snippet.code = 'import gitlab\nimport whatever'
snippet.save
```

Delete a snippet:

```
project.snippets.delete(snippet_id)
# or
snippet.delete()
```

Get user agent detail (admin only):

```
detail = snippet.user_agent_detail()
```

6.27.7 Notes

See *Notes*.

6.27.8 Project members

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectMember`
 - `gitlab.v4.objects.ProjectMemberManager`
 - `gitlab.v4.objects.Project.members`
- GitLab API: <https://docs.gitlab.com/ce/api/members.html>

Examples

List the project members:

```
members = project.members.list()
```

List the project members recursively (including inherited members through ancestor groups):

```
members = project.members.all(all=True)
```

Search project members matching a query string:

```
members = project.members.list(query='bar')
```

Get a single project member:

```
member = project.members.get(user_id)
```

Add a project member:

```
member = project.members.create({'user_id': user.id, 'access_level':
                                gitlab.DEVELOPER_ACCESS})
```

Modify a project member (change the access level):

```
member.access_level = gitlab.MAINTAINER_ACCESS
member.save()
```

Remove a member from the project team:

```
project.members.delete(user.id)
# or
member.delete()
```

Share/unshare the project with a group:

```
project.share(group.id, gitlab.DEVELOPER_ACCESS)
project.unshare(group.id)
```

6.27.9 Project hooks

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectHook`
 - `gitlab.v4.objects.ProjectHookManager`
 - `gitlab.v4.objects.Project.hooks`
- GitLab API: <https://docs.gitlab.com/ce/api/projects.html#hooks>

Examples

List the project hooks:

```
hooks = project.hooks.list()
```

Get a project hook:

```
hook = project.hooks.get(hook_id)
```

Create a project hook:

```
hook = project.hooks.create({'url': 'http://my/action/url', 'push_events': 1})
```

Update a project hook:

```
hook.push_events = 0
hook.save()
```

Delete a project hook:

```
project.hooks.delete(hook_id)
# or
hook.delete()
```

6.27.10 Project Services

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectService`
 - `gitlab.v4.objects.ProjectServiceManager`
 - `gitlab.v4.objects.Project.services`
- GitLab API: <https://docs.gitlab.com/ce/api/services.html>

Examples

Get a service:

```
service = project.services.get('asana')
# display its status (enabled/disabled)
print(service.active)
```

List the code names of available services (doesn't return objects):

```
services = project.services.available()
```

Configure and enable a service:

```
service.api_key = 'randomkey'
service.save()
```

Disable a service:

```
service.delete()
```

6.27.11 File uploads

Reference

- v4 API:
 - `gitlab.v4.objects.Project.upload`
- Gitlab API: <https://docs.gitlab.com/ce/api/projects.html#upload-a-file>

Examples

Upload a file into a project using a filesystem path:

```
project.upload("filename.txt", filepath="/some/path/filename.txt")
```

Upload a file into a project without a filesystem path:

```
project.upload("filename.txt", filedata="Raw data")
```

Upload a file and comment on an issue using the uploaded file's markdown:

```
uploaded_file = project.upload("filename.txt", filedata="data")
issue = project.issues.get(issue_id)
issue.notes.create({
    "body": "See the attached file: {}".format(uploaded_file["markdown"])
})
```

Upload a file and comment on an issue while using custom markdown to reference the uploaded file:

```
uploaded_file = project.upload("filename.txt", filedata="data")
issue = project.issues.get(issue_id)
issue.notes.create({
    "body": "See the [attached file]({})".format(uploaded_file["url"])
})
```

6.27.12 Project push rules

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectPushRules`
 - `gitlab.v4.objects.ProjectPushRulesManager`
 - `gitlab.v4.objects.Project.pushrules`
- GitLab API: <https://docs.gitlab.com/ee/api/projects.html#push-rules>

Examples

Create project push rules (at least one rule is necessary):

```
project.pushrules.create({'deny_delete_tag': True})
```

Get project push rules (returns None if there are no push rules):

```
pr = project.pushrules.get()
```

Edit project push rules:

```
pr.branch_name_regex = '^^(master|develop|support-\d+|release-\d+\.\.+|hotfix-\d+|feature-\d+)$'
pr.save()
```

Delete project push rules:

```
pr.delete()
```

6.27.13 Project releases

Reference

- v4 API:

- `gitlab.v4.objects.ProjectRelease`
- `gitlab.v4.objects.ProjectReleaseManager`
- `gitlab.v4.objects.Project.releases`
- Gitlab API: <https://docs.gitlab.com/ee/api/releases/index.html>

Examples

Get a list of releases from a project:

```
release = project.releases.list()
```

Get a single release:

```
release = project.releases.get('v1.2.3')
```

Create a release for a project tag:

```
release = project.releases.create({'name': 'Demo Release', 'tag_name': 'v1.2.3',
↪ 'description': 'release notes go here'})
```

Delete a release:

```
release = p.releases.delete('v1.2.3')
```

6.27.14 Project protected tags

Reference

- v4 API:
 - `gitlab.v4.objects.ProjectProtectedTag`
 - `gitlab.v4.objects.ProjectProtectedTagManager`
 - `gitlab.v4.objects.Project.protectedtags`
- GitLab API: https://docs.gitlab.com/ce/api/protected_tags.html

Examples

Get a list of protected tags from a project:

```
protected_tags = project.protectedtags.list()
```

Get a single protected tag or wildcard protected tag:

```
protected_tag = project.protectedtags.get('v*')
```

Protect a single repository tag or several project repository tags using a wildcard protected tag:

```
project.protectedtags.create({'name': 'v*', 'create_access_level': '40'})
```

Unprotect the given protected tag or wildcard protected tag.:

```
protected_tag.delete()
```

6.28 Protected branches

You can define a list of protected branch names on a repository. Names can use wildcards (*).

6.28.1 References

- v4 API:
 - `gitlab.v4.objects.ProjectProtectedBranch`
 - `gitlab.v4.objects.ProjectProtectedBranchManager`
 - `gitlab.v4.objects.Project.protectedbranches`
- GitLab API: https://docs.gitlab.com/ce/api/protected_branches.html#protected-branches-api

6.28.2 Examples

Get the list of protected branches for a project:

```
p_branches = project.protectedbranches.list()
```

Get a single protected branch:

```
p_branch = project.protectedbranches.get('master')
```

Create a protected branch:

```
p_branch = project.protectedbranches.create({
    'name': '*-stable',
    'merge_access_level': gitlab.DEVELOPER_ACCESS,
    'push_access_level': gitlab.MAINTAINER_ACCESS
})
```

Create a protected branch with more granular access control:

```
p_branch = project.protectedbranches.create({
    'name': '*-stable',
    'allowed_to_push': [{"user_id": 99}, {"user_id": 98}],
    'allowed_to_merge': [{"group_id": 653}],
    'allowed_to_unprotect': [{"access_level": gitlab.MAINTAINER_ACCESS}]
})
```

Delete a protected branch:

```
project.protectedbranches.delete('*-stable')
# or
p_branch.delete()
```


6.29 Runners

Runners are external processes used to run CI jobs. They are deployed by the administrator and registered to the GitLab instance.

Shared runners are available for all projects. Specific runners are enabled for a list of projects.

6.29.1 Global runners (admin)

Reference

- v4 API:
 - `gitlab.v4.objects.Runner`
 - `gitlab.v4.objects.RunnerManager`
 - `gitlab.Gitlab.runners`
- GitLab API: <https://docs.gitlab.com/ce/api/runners.html>

Examples

Use the `list()` and `all()` methods to list runners.

Both methods accept a `scope` parameter to filter the list. Allowed values for this parameter are:

- `active`
- `paused`
- `online`
- `specific(all() only)`
- `shared(all() only)`

Note: The returned objects hold minimal information about the runners. Use the `get()` method to retrieve detail about a runner.

```
# List owned runners
runners = gl.runners.list()
# With a filter
runners = gl.runners.list(scope='active')
# List all runners, using a filter
runners = gl.runners.all(scope='paused')
```

Get a runner's detail:

```
runner = gl.runners.get(runner_id)
```

Register a new runner:

```
runner = gl.runners.create({'token': secret_token})
```

Update a runner:

```
runner = gl.runners.get(runner_id)
runner.tag_list.append('new_tag')
runner.save()
```

Remove a runner:

```
gl.runners.delete(runner_id)
# or
runner.delete()
```

Verify a registered runner token:

```
try:
    gl.runners.verify(runner_token)
    print("Valid token")
except GitlabVerifyError:
    print("Invalid token")
```

6.29.2 Project runners

Reference

- v4 API:
 - *gitlab.v4.objects.ProjectRunner*
 - *gitlab.v4.objects.ProjectRunnerManager*
 - *gitlab.v4.objects.Project.runners*
- GitLab API: <https://docs.gitlab.com/ce/api/runners.html>

Examples

List the runners for a project:

```
runners = project.runners.list()
```

Enable a specific runner for a project:

```
p_runner = project.runners.create({'runner_id': runner.id})
```

Disable a specific runner for a project:

```
project.runners.delete(runner.id)
```

6.29.3 Runner jobs

Reference

- v4 API:
 - *gitlab.v4.objects.RunnerJob*
 - *gitlab.v4.objects.RunnerJobManager*

- `gitlab.v4.objects.Runner.jobs`
- GitLab API: <https://docs.gitlab.com/ce/api/runners.html>

Examples

List for jobs for a runner:

```
jobs = runner.jobs.list()
```

Filter the list using the jobs status:

```
# status can be 'running', 'success', 'failed' or 'canceled'
active_jobs = runner.jobs.list(status='running')
```

6.30 Registry Repositories

6.30.1 References

- v4 API:
 - `gitlab.v4.objects.ProjectRegistryRepository`
 - `gitlab.v4.objects.ProjectRegistryRepositoryManager`
 - `gitlab.v4.objects.Project.repositories`
- Gitlab API: https://docs.gitlab.com/ce/api/container_registry.html

6.30.2 Examples

Get the list of container registry repositories associated with the project:

```
repositories = project.repositories.list()
```

Delete repository:

```
project.repositories.delete(id=x)
# or
repository = repositories.pop()
repository.delete()
```

6.31 Registry Repository Tags

6.31.1 References

- v4 API:
 - `gitlab.v4.objects.ProjectRegistryTag`
 - `gitlab.v4.objects.ProjectRegistryTagManager`
 - `gitlab.v4.objects.Repository.tags`

- Gitlab API: https://docs.gitlab.com/ce/api/container_registry.html

6.31.2 Examples

Get the list of repository tags in given registry:

```
repositories = project.repositories.list()
repository = repositories.pop()
tags = repository.tags.list()
```

Get specific tag:

```
repository.tags.get(id=tag_name)
```

Delete tag:

```
repository.tags.delete(id=tag_name)
# or
tag = repository.tags.get(id=tag_name)
tag.delete()
```

Delete tag in bulk:

```
repository.tags.delete_in_bulk(keep_n=1)
# or
repository.tags.delete_in_bulk(older_than="1m")
# or
repository.tags.delete_in_bulk(name_regex="v.+ ", keep_n=2)
```

Note: Delete in bulk is asynchronous operation and may take a while. Refer to: https://docs.gitlab.com/ce/api/container_registry.html#delete-repository-tags-in-bulk

6.32 Search API

You can search for resources at the top level, in a project or in a group. Searches are based on a scope (issues, merge requests, and so on) and a search string.

6.32.1 Reference

- v4 API:
 - `gitlab.Gitlab.search`
 - `gitlab.v4.objects.Group.search`
 - `gitlab.v4.objects.Project.search`
- GitLab API: <https://docs.gitlab.com/ce/api/search.html>

6.32.2 Examples

Search for issues matching a specific string:

```
# global search
gl.search('issues', 'regression')

# group search
group = gl.groups.get('mygroup')
group.search('issues', 'regression')

# project search
project = gl.projects.get('myproject')
project.search('issues', 'regression')
```

The search() methods implement the pagination support:

```
# get lists of 10 items, and start at page 2
gl.search('issues', search_str, page=2, per_page=10)

# get a generator that will automatically make required API calls for
# pagination
for item in gl.search('issues', search_str, as_list=False):
    do_something(item)
```

The search API doesn't return objects, but dicts. If you need to act on objects, you need to create them explicitly:

```
for item in gl.search('issues', search_str, as_list=False):
    issue_project = gl.projects.get(item['project_id'], lazy=True)
    issue = issue_project.issues.get(item['iid'])
    issue.state = 'closed'
    issue.save()
```

6.33 Settings

6.33.1 Reference

- v4 API:
 - `gitlab.v4.objects.ApplicationSettings`
 - `gitlab.v4.objects.ApplicationSettingsManager`
 - `gitlab.Gitlab.settings`
- GitLab API: <https://docs.gitlab.com/ce/api/settings.html>

6.33.2 Examples

Get the settings:

```
settings = gl.settings.get()
```

Update the settings:

```
settings.signin_enabled = False
settings.save()
```

6.34 Snippets

6.34.1 Reference

- v4 API:
 - `gitlab.v4.objects.Snippet`
 - `gitlab.v4.objects.SnippetManager`
 - `gitlab.Gitlab.snippets`
- GitLab API: <https://docs.gitlab.com/ce/api/snippets.html>

6.34.2 Examples

List snippets owned by the current user:

```
snippets = gl.snippets.list()
```

List the public snippets:

```
public_snippets = gl.snippets.public()
```

Get a snippet:

```
snippet = gl.snippets.get(snippet_id)
# get the content
content = snippet.content()
```

Warning: Blobs are entirely stored in memory unless you use the streaming feature. See *the artifacts example*.

Create a snippet:

```
snippet = gl.snippets.create({'title': 'snippet1',
                             'file_name': 'snippet1.py',
                             'content': open('snippet1.py').read()})
```

Update the snippet attributes:

```
snippet.visibility_level = gitlab.Project.VISIBILITY_PUBLIC
snippet.save()
```

To update a snippet code you need to create a `ProjectSnippet` object:

```
snippet = gl.snippets.get(snippet_id)
project = gl.projects.get(snippet.project_id, lazy=True)
editable_snippet = project.snippets.get(snippet.id)
editable_snippet.code = new_snippet_content
editable_snippet.save()
```

Delete a snippet:

```
gl.snippets.delete(snippet_id)
# or
snippet.delete()
```

Get user agent detail (admin only):

```
detail = snippet.user_agent_detail()
```

6.35 System hooks

6.35.1 Reference

- v4 API:
 - `gitlab.v4.objects.Hook`
 - `gitlab.v4.objects.HookManager`
 - `gitlab.Gitlab.hooks`
- GitLab API: https://docs.gitlab.com/ce/api/system_hooks.html

6.35.2 Examples

List the system hooks:

```
hooks = gl.hooks.list()
```

Create a system hook:

```
gl.hooks.get(hook_id)
```

Test a system hook. The returned object is not usable (it misses the hook ID):

```
hook = gl.hooks.create({'url': 'http://your.target.url'})
```

Delete a system hook:

```
gl.hooks.delete(hook_id)
# or
hook.delete()
```

6.36 Templates

You can request templates for different type of files:

- License files
- .gitignore files
- GitLab CI configuration files
- Dockerfiles

6.36.1 License templates

Reference

- v4 API:
 - `gitlab.v4.objects.License`
 - `gitlab.v4.objects.LicenseManager`
 - `gitlab.Gitlab.licenses`
- GitLab API: <https://docs.gitlab.com/ce/api/templates/licenses.html>

Examples

List known license templates:

```
licenses = gl.licenses.list()
```

Generate a license content for a project:

```
license = gl.licenses.get('apache-2.0', project='foobar', fullname='John Doe')
print(license.content)
```

6.36.2 .gitignore templates

Reference

- v4 API:
 - `gitlab.v4.objects.Gitignore`
 - `gitlab.v4.objects.GitignoreManager`
 - `gitlab.Gitlab.gitignores`
- GitLab API: <https://docs.gitlab.com/ce/api/templates/gitignores.html>

Examples

List known gitignore templates:

```
gitignores = gl.gitignores.list()
```

Get a gitignore template:

```
gitignore = gl.gitignores.get('Python')
print(gitignore.content)
```

6.36.3 GitLab CI templates

Reference

- v4 API:

- `gitlab.v4.objects.Gitlabciyaml`
- `gitlab.v4.objects.GitlabciyamlManager`
- `gitlab.Gitlab.gitlabciyaml`

- GitLab API: https://docs.gitlab.com/ce/api/templates/gitlab_ci_ymls.html

Examples

List known GitLab CI templates:

```
gitlabciyaml = gl.gitlabciyaml.list()
```

Get a GitLab CI template:

```
gitlabciyaml = gl.gitlabciyaml.get('Pelican')
print(gitlabciyaml.content)
```

6.36.4 Dockerfile templates

Reference

- v4 API:
 - `gitlab.v4.objects.Dockerfile`
 - `gitlab.v4.objects.DockerfileManager`
 - `gitlab.Gitlab.gitlabciyaml`
- GitLab API: Not documented.

Examples

List known Dockerfile templates:

```
dockerfiles = gl.dockerfiles.list()
```

Get a Dockerfile template:

```
dockerfile = gl.dockerfiles.get('Python')
print(dockerfile.content)
```

6.37 Todos

6.37.1 Reference

- v4 API:
 - `Todo`
 - `TodoManager`
 - `gitlab.Gitlab.todos`

- GitLab API: <https://docs.gitlab.com/ce/api/todos.html>

6.37.2 Examples

List active todos:

```
todos = gl.todos.list()
```

You can filter the list using the following parameters:

- `action`: can be assigned, mentioned, build_failed, marked, or approval_required
- `author_id`
- `project_id`
- `state`: can be pending or done
- `type`: can be Issue or MergeRequest

For example:

```
todos = gl.todos.list(project_id=1)
todos = gl.todos.list(state='done', type='Issue')
```

Mark a todo as done:

```
gl.todos.delete(todo_id)
# or
todo.delete()
```

Mark all the todos as done:

```
nb_of_closed_todos = gl.todos.delete_all()
```

6.38 Users and current user

The Gitlab API exposes user-related method that can be manipulated by admins only.

The currently logged-in user is also exposed.

6.38.1 Users

References

- v4 API:
 - `gitlab.v4.objects.User`
 - `gitlab.v4.objects.UserManager`
 - `gitlab.Gitlab.users`
- GitLab API: <https://docs.gitlab.com/ce/api/users.html>

Examples

Get the list of users:

```
users = gl.users.list()
```

Search users whose username match a given string:

```
users = gl.users.list(search='foo')
```

Get a single user:

```
# by ID
user = gl.users.get(user_id)
# by username
user = gl.users.list(username='root')[0]
```

Create a user:

```
user = gl.users.create({'email': 'john@doe.com',
                        'password': 's3cur3s3cr3T',
                        'username': 'jdoe',
                        'name': 'John Doe'})
```

Update a user:

```
user.name = 'Real Name'
user.save()
```

Delete a user:

```
gl.users.delete(user_id)
# or
user.delete()
```

Block/Unblock a user:

```
user.block()
user.unblock()
```

Set the avatar image for a user:

```
# the avatar image can be passed as data (content of the file) or as a file
# object opened in binary mode
user.avatar = open('path/to/file.png', 'rb')
user.save()
```

Set an external identity for a user:

```
user.provider = 'oauth2_generic'
user.extern_uid = '3'
user.save()
```

6.38.2 User custom attributes

References

- v4 API:
 - `gitlab.v4.objects.UserCustomAttribute`
 - `gitlab.v4.objects.UserCustomAttributeManager`
 - `gitlab.v4.objects.User.customattributes`
- GitLab API: https://docs.gitlab.com/ce/api/custom_attributes.html

Examples

List custom attributes for a user:

```
attrs = user.customattributes.list()
```

Get a custom attribute for a user:

```
attr = user.customattributes.get(attr_key)
```

Set (create or update) a custom attribute for a user:

```
attr = user.customattributes.set(attr_key, attr_value)
```

Delete a custom attribute for a user:

```
attr.delete()  
# or  
user.customattributes.delete(attr_key)
```

Search users by custom attribute:

```
user.customattributes.set('role', 'QA')  
gl.users.list(custom_attributes={'role': 'QA'})
```

6.38.3 User impersonation tokens

References

- v4 API:
 - `gitlab.v4.objects.UserImpersonationToken`
 - `gitlab.v4.objects.UserImpersonationTokenManager`
 - `gitlab.v4.objects.User.impersonationtokens`
- GitLab API: <https://docs.gitlab.com/ce/api/users.html#get-all-impersonation-tokens-of-a-user>

List impersonation tokens for a user:

```
i_t = user.impersonationtokens.list(state='active')  
i_t = user.impersonationtokens.list(state='inactive')
```

Get an impersonation token for a user:

```
i_t = user.impersonationtokens.get(i_t_id)
```

Create and use an impersonation token for a user:

```
i_t = user.impersonationtokens.create({'name': 'token1', 'scopes': ['api']})
# use the token to create a new gitlab connection
user_gl = gitlab.Gitlab(gitlab_url, private_token=i_t.token)
```

Revoke (delete) an impersonation token for a user:

```
i_t.delete()
```

6.38.4 Current User

References

- v4 API:
 - *gitlab.v4.objects.CurrentUser*
 - *gitlab.v4.objects.CurrentUserManager*
 - `gitlab.Gitlab.user`
- GitLab API: <https://docs.gitlab.com/ce/api/users.html>

Examples

Get the current user:

```
gl.auth()
current_user = gl.user
```

6.38.5 GPG keys

References

You can manipulate GPG keys for the current user and for the other users if you are admin.

- v4 API:
 - *gitlab.v4.objects.CurrentUserGPGKey*
 - *gitlab.v4.objects.CurrentUserGPGKeyManager*
 - `gitlab.v4.objects.CurrentUser.gpgkeys`
 - *gitlab.v4.objects.UserGPGKey*
 - *gitlab.v4.objects.UserGPGKeyManager*
 - `gitlab.v4.objects.User.gpgkeys`
- GitLab API: <https://docs.gitlab.com/ce/api/users.html#list-all-gpg-keys>

Examples

List GPG keys for a user:

```
gpgkeys = user.gpgkeys.list()
```

Get a GPG gpgkey for a user:

```
gpgkey = user.gpgkeys.get(key_id)
```

Create a GPG gpgkey for a user:

```
# get the key with `gpg --export -a GPG_KEY_ID`  
k = user.gpgkeys.create({'key': public_key_content})
```

Delete a GPG gpgkey for a user:

```
user.gpgkeys.delete(key_id)  
# or  
gpgkey.delete()
```

6.38.6 SSH keys

References

You can manipulate SSH keys for the current user and for the other users if you are admin.

- v4 API:
 - `gitlab.v4.objects.CurrentUserKey`
 - `gitlab.v4.objects.CurrentUserKeyManager`
 - `gitlab.v4.objects.CurrentUser.keys`
 - `gitlab.v4.objects.UserKey`
 - `gitlab.v4.objects.UserKeyManager`
 - `gitlab.v4.objects.User.keys`
- GitLab API: <https://docs.gitlab.com/ce/api/users.html#list-ssh-keys>

Examples

List SSH keys for a user:

```
keys = user.keys.list()
```

Create an SSH key for a user:

```
k = user.keys.create({'title': 'my_key',  
                    'key': open('/home/me/.ssh/id_rsa.pub').read()})
```

Delete an SSH key for a user:

```
user.keys.delete(key_id)
# or
key.delete()
```

6.38.7 Emails

References

You can manipulate emails for the current user and for the other users if you are admin.

- v4 API:
 - *gitlab.v4.objects.CurrentUserEmail*
 - *gitlab.v4.objects.CurrentUserEmailManager*
 - *gitlab.v4.objects.CurrentUser.emails*
 - *gitlab.v4.objects.UserEmail*
 - *gitlab.v4.objects.UserEmailManager*
 - *gitlab.v4.objects.User.emails*
- GitLab API: <https://docs.gitlab.com/ce/api/users.html#list-emails>

Examples

List emails for a user:

```
emails = user.emails.list()
```

Get an email for a user:

```
email = user.emails.get(email_id)
```

Create an email for a user:

```
k = user.emails.create({'email': 'foo@bar.com'})
```

Delete an email for a user:

```
user.emails.delete(email_id)
# or
email.delete()
```

6.38.8 Users activities

References

- admin only
- v4 API:
 - *gitlab.v4.objects.UserActivities*
 - *gitlab.v4.objects.UserActivitiesManager*

- `gitlab.Gitlab.user_activities`

- GitLab API: <https://docs.gitlab.com/ce/api/users.html#get-user-activities-admin-only>

Examples

Get the users activities:

```
activities = gl.user_activities.list(  
    query_parameters={'from': '2018-07-01'},  
    all=True, as_list=False)
```

6.39 Sidekiq metrics

6.39.1 Reference

- v4 API:
 - `gitlab.v4.objects.SidekiqManager`
 - `gitlab.Gitlab.sidekiq`
- GitLab API: https://docs.gitlab.com/ce/api/sidekiq_metrics.html

6.39.2 Examples

```
gl.sidekiq.queue_metrics()  
gl.sidekiq.process_metrics()  
gl.sidekiq.job_stats()  
gl.sidekiq.compound_metrics()
```

6.40 Wiki pages

6.40.1 References

- v4 API:
 - `gitlab.v4.objects.ProjectWiki`
 - `gitlab.v4.objects.ProjectWikiManager`
 - `gitlab.v4.objects.Project.wikis`
- GitLab API: <https://docs.gitlab.com/ce/api/wikis.html>

Examples

Get the list of wiki pages for a project:

```
pages = project.wikis.list()
```

Get a single wiki page:


```
page = project.wikis.get(page_slug)
```

Create a wiki page:

```
page = project.wikis.create({'title': 'Wiki Page 1',  
                             'content': open(a_file).read()})
```

Update a wiki page:

```
page.content = 'My new content'  
page.save()
```

Delete a wiki page:

```
page.delete()
```


7.1 Subpackages

7.1.1 gitlab.v4 package

Submodules

gitlab.v4.objects module

class `gitlab.v4.objects.ApplicationSettings` (*manager, attrs*)
Bases: `gitlab.mixins.SaveMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.ApplicationSettingsManager` (*gl, parent=None*)
Bases: `gitlab.mixins.GetWithoutIdMixin`, `gitlab.mixins.UpdateMixin`, `gitlab.base.RESTManager`

Object update

Optional attributes for object update:

- `admin_notification_email`
- `after_sign_out_path`
- `after_sign_up_text`
- `akismet_api_key`
- `akismet_enabled`
- `circuitbreaker_access_retries`
- `circuitbreaker_check_interval`
- `circuitbreaker_failure_count_threshold`
- `circuitbreaker_failure_reset_time`

- `circuitbreaker_storage_timeout`
- `clientside_sentry_dsn`
- `clientside_sentry_enabled`
- `container_registry_token_expire_delay`
- `default_artifacts_expire_in`
- `default_branch_protection`
- `default_group_visibility`
- `default_project_visibility`
- `default_projects_limit`
- `default_snippet_visibility`
- `disabled_oauth_sign_in_sources`
- `domain_blacklist_enabled`
- `domain_blacklist`
- `domain_whitelist`
- `dsa_key_restriction`
- `ecdsa_key_restriction`
- `ed25519_key_restriction`
- `email_author_in_body`
- `enabled_git_access_protocol`
- `gravatar_enabled`
- `help_page_hide_commercial_content`
- `help_page_support_url`
- `home_page_url`
- `housekeeping_bitmaps_enabled`
- `housekeeping_enabled`
- `housekeeping_full_repack_period`
- `housekeeping_gc_period`
- `housekeeping_incremental_repack_period`
- `html_emails_enabled`
- `import_sources`
- `koding_enabled`
- `koding_url`
- `max_artifacts_size`
- `max_attachment_size`
- `max_pages_size`
- `metrics_enabled`

- `metrics_host`
- `metrics_method_call_threshold`
- `metrics_packet_size`
- `metrics_pool_size`
- `metrics_port`
- `metrics_sample_interval`
- `metrics_timeout`
- `password_authentication_enabled_for_web`
- `password_authentication_enabled_for_git`
- `performance_bar_allowed_group_id`
- `performance_bar_enabled`
- `plantuml_enabled`
- `plantuml_url`
- `polling_interval_multiplier`
- `project_export_enabled`
- `prometheus_metrics_enabled`
- `recaptcha_enabled`
- `recaptcha_private_key`
- `recaptcha_site_key`
- `repository_checks_enabled`
- `repository_storages`
- `require_two_factor_authentication`
- `restricted_visibility_levels`
- `rsa_key_restriction`
- `send_user_confirmation_email`
- `sentry_dsn`
- `sentry_enabled`
- `session_expire_delay`
- `shared_runners_enabled`
- `shared_runners_text`
- `sidekiq_throttling_enabled`
- `sidekiq_throttling_factor`
- `sidekiq_throttling_queues`
- `sign_in_text`
- `signup_enabled`
- `terminal_max_session_time`

- `two_factor_grace_period`
- `unique_ips_limit_enabled`
- `unique_ips_limit_per_user`
- `unique_ips_limit_time_window`
- `usage_ping_enabled`
- `user_default_external`
- `user_oauth_applications`
- `version_check_enabled`
- `enforce_terms`
- `terms`

update (***kwargs*)

Update an object on the server.

Parameters

- **id** – ID of the object to update (can be None if not required)
- **new_data** – the update data for the object
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns The new object data (*not* a RESTObject)

Return type dict

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabUpdateError` – If the server cannot perform the request

class `gitlab.v4.objects.BroadcastMessage` (*manager, attrs*)

Bases: `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.BroadcastMessageManager` (*gl, parent=None*)

Bases: `gitlab.mixins.CRUDMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- `message`

Optional attributes:

- `starts_at`
- `ends_at`
- `color`
- `font`

Object update

Optional attributes for object update:

- `message`

- starts_at
- ends_at
- color
- font

class gitlab.v4.objects.**CurrentUser** (*manager, attrs*)

Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**CurrentUserEmail** (*manager, attrs*)

Bases: *gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**CurrentUserEmailManager** (*gl, parent=None*)

Bases: *gitlab.mixins.RetrieveMixin, gitlab.mixins.CreateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- email

class gitlab.v4.objects.**CurrentUserGPGKey** (*manager, attrs*)

Bases: *gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**CurrentUserGPGKeyManager** (*gl, parent=None*)

Bases: *gitlab.mixins.RetrieveMixin, gitlab.mixins.CreateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- key

class gitlab.v4.objects.**CurrentUserKey** (*manager, attrs*)

Bases: *gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**CurrentUserKeyManager** (*gl, parent=None*)

Bases: *gitlab.mixins.RetrieveMixin, gitlab.mixins.CreateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- title
- key

class gitlab.v4.objects.**CurrentUserManager** (*gl, parent=None*)

Bases: *gitlab.mixins.GetWithoutIdMixin, gitlab.base.RESTManager*

class gitlab.v4.objects.**DeployKey** (*manager, attrs*)

Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**DeployKeyManager** (*gl, parent=None*)

Bases: *gitlab.mixins.ListMixin, gitlab.base.RESTManager*

class gitlab.v4.objects.**Dockerfile** (*manager, attrs*)

Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**DockerfileManager** (*gl, parent=None*)

Bases: *gitlab.mixins.RetrieveMixin, gitlab.base.RESTManager*

class gitlab.v4.objects.**Event** (*manager, attrs*)

Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**EventManager** (*gl, parent=None*)

Bases: *gitlab.mixins.ListMixin, gitlab.base.RESTManager*

Object listing filters

- `action`
- `target_type`
- `before`
- `after`
- `sort`

class gitlab.v4.objects.**Feature** (*manager, attrs*)

Bases: *gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**FeatureManager** (*gl, parent=None*)

Bases: *gitlab.mixins.ListMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager*

set (***kwargs*)

Create or update the object.

Parameters

- **name** (*str*) – The value to set for the object
- **value** (*bool/int*) – The value to set for the object
- **feature_group** (*str*) – A feature group name
- **user** (*str*) – A GitLab username
- ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabSetError` – If an error occurred

Returns The created/updated attribute

Return type `obj`

class gitlab.v4.objects.**GeoNode** (*manager, attrs*)

Bases: *gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

repair (***kwargs*)

Repair the OAuth authentication of the geo node.

Parameters ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabRepairError` – If the server failed to perform the request

status (***kwargs*)

Get the status of the geo node.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server failed to perform the request

Returns The status of the geo node

Return type dict

class `gitlab.v4.objects.GeoNodeManager` (*gl, parent=None*)

Bases: `gitlab.mixins.RetrieveMixin`, `gitlab.mixins.UpdateMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

Object update

Optional attributes for object update:

- `enabled`
- `url`
- `files_max_capacity`
- `repos_max_capacity`

current_failures (***kwargs*)

Get the list of failures on the current geo node.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server failed to perform the request

Returns The list of failures

Return type list

status (***kwargs*)

Get the status of all the geo nodes.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server failed to perform the request

Returns The status of all the geo nodes

Return type list

class `gitlab.v4.objects.Gitignore` (*manager, attrs*)

Bases: `gitlab.base.RESTObject`

class `gitlab.v4.objects.GitignoreManager` (*gl, parent=None*)

Bases: `gitlab.mixins.RetrieveMixin`, `gitlab.base.RESTManager`

class `gitlab.v4.objects.Gitlabciyaml` (*manager, attrs*)

Bases: `gitlab.base.RESTObject`

class `gitlab.v4.objects.GitlabciyamlManager` (*gl, parent=None*)

Bases: `gitlab.mixins.RetrieveMixin`, `gitlab.base.RESTManager`

class gitlab.v4.objects.**Group** (*manager, attrs*)

Bases: *gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

add_ldap_group_link (***kwargs*)

Add an LDAP group link.

Parameters

- **cn** (*str*) – CN of the LDAP group
- **group_access** (*int*) – Minimum access level for members of the LDAP group
- **provider** (*str*) – LDAP provider for the LDAP group
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the server cannot perform the request

delete_ldap_group_link (***kwargs*)

Delete an LDAP group link.

Parameters

- **cn** (*str*) – CN of the LDAP group
- **provider** (*str*) – LDAP provider for the LDAP group
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabDeleteError` – If the server cannot perform the request

ldap_sync (***kwargs*)

Sync LDAP groups.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the server cannot perform the request

search (***kwargs*)

Search the group resources matching the provided string.

Parameters

- **scope** (*str*) – Scope of the search
- **search** (*str*) – Search string
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabSearchError` – If the server failed to perform the request

Returns A list of dicts describing the resources found.

Return type *GitlabList*

transfer_project (***kwargs*)

Transfer a project to this group.

Parameters

- **to_project_id** (*int*) – ID of the project to transfer
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabTransferProjectError` – If the project could not be transferred

class `gitlab.v4.objects.GroupAccessRequest` (*manager, attrs*)

Bases: `gitlab.mixins.AccessRequestMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.GroupAccessRequestManager` (*gl, parent=None*)

Bases: `gitlab.mixins.ListMixin`, `gitlab.mixins.CreateMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

class `gitlab.v4.objects.GroupBadge` (*manager, attrs*)

Bases: `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.GroupBadgeManager` (*gl, parent=None*)

Bases: `gitlab.mixins.BadgeRenderMixin`, `gitlab.mixins.CRUDMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- `link_url`
- `image_url`

Object update

Optional attributes for object update:

- `link_url`
- `image_url`

class `gitlab.v4.objects.GroupBoard` (*manager, attrs*)

Bases: `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.GroupBoardList` (*manager, attrs*)

Bases: `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.GroupBoardListManager` (*gl, parent=None*)

Bases: `gitlab.mixins.CRUDMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- `label_id`

Object update

Mandatory attributes for object update:

- `position`

```
class gitlab.v4.objects.GroupBoardManager (gl, parent=None)  
    Bases: gitlab.mixins.CRUDMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- `name`

```
class gitlab.v4.objects.GroupCustomAttribute (manager, attrs)  
    Bases: gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.GroupCustomAttributeManager (gl, parent=None)  
    Bases: gitlab.mixins.RetrieveMixin, gitlab.mixins.SetMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager
```

```
class gitlab.v4.objects.GroupEpic (manager, attrs)  
    Bases: gitlab.mixins.ObjectDeleteMixin, gitlab.mixins.SaveMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.GroupEpicIssue (manager, attrs)  
    Bases: gitlab.mixins.ObjectDeleteMixin, gitlab.mixins.SaveMixin, gitlab.base.RESTObject
```

save (***kwargs*)

Save the changes made to the object to the server.

The object is updated to match what the server returns.

Parameters ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raise: `GitlabAuthenticationError`: If authentication is not correct `GitlabUpdateError`: If the server cannot perform the request

```
class gitlab.v4.objects.GroupEpicIssueManager (gl, parent=None)  
    Bases: gitlab.mixins.ListMixin, gitlab.mixins.CreateMixin, gitlab.mixins.UpdateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- `issue_id`

Object update

Optional attributes for object update:

- `move_before_id`
- `move_after_id`

create (***kwargs*)

Create a new object.

Parameters

- **data** (*dict*) – Parameters to send to the server to create the resource
- ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the server cannot perform the request

Returns

A new instance of the manage object class build with the data sent by the server

Return type *RESTObject*

```
class gitlab.v4.objects.GroupEpicManager (gl, parent=None)
    Bases: gitlab.mixins.CRUDMixin, gitlab.base.RESTManager
```

Object listing filters

- `author_id`
- `labels`
- `order_by`
- `sort`
- `search`

Object Creation

Mandatory attributes:

- `title`

Optional attributes:

- `labels`
- `description`
- `start_date`
- `end_date`

Object update

Optional attributes for object update:

- `title`
- `labels`
- `description`
- `start_date`
- `end_date`

```
class gitlab.v4.objects.GroupEpicResourceLabelEvent (manager, attrs)
    Bases: gitlab.base.RESTObject
```

```
class gitlab.v4.objects.GroupEpicResourceLabelEventManager (gl, parent=None)
    Bases: gitlab.mixins.RetrieveMixin, gitlab.base.RESTManager
```

```
class gitlab.v4.objects.GroupIssue (manager, attrs)
    Bases: gitlab.base.RESTObject
```

```
class gitlab.v4.objects.GroupIssueManager (gl, parent=None)
    Bases: gitlab.mixins.ListMixin, gitlab.base.RESTManager
```

Object listing filters

- state
- labels
- milestone
- order_by
- sort
- iids
- author_id
- assignee_id
- my_reaction_emoji
- search
- created_after
- created_before
- updated_after
- updated_before

class gitlab.v4.objects.**GroupLabel** (*manager, attrs*)

Bases: `gitlab.mixins.SubscribableMixin`, `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

save (***kwargs*)

Saves the changes made to the object to the server.

The object is updated to match what the server returns.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct.
- `GitlabUpdateError` – If the server cannot perform the request.

class gitlab.v4.objects.**GroupLabelManager** (*gl, parent=None*)

Bases: `gitlab.mixins.ListMixin`, `gitlab.mixins.CreateMixin`, `gitlab.mixins.UpdateMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- name
- color

Optional attributes:

- description
- priority

Object update

Mandatory attributes for object update:

- name

Optional attributes for object update:

- `new_name`
- `color`
- `description`
- `priority`

delete (***kwargs*)

Delete a Label on the server.

Parameters

- **name** – The name of the label
- ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabDeleteError` – If the server cannot perform the request

update (*name, new_data=None, **kwargs*)

Update a Label on the server.

Parameters

- **name** – The name of the label
- ****kwargs** – Extra options to send to the server (e.g. `sudo`)

class `gitlab.v4.objects.GroupManager` (*gl, parent=None*)

Bases: `gitlab.mixins.CRUDMixin`, `gitlab.base.RESTManager`

Object listing filters

- `skip_groups`
- `all_available`
- `search`
- `order_by`
- `sort`
- `statistics`
- `owned`
- `with_custom_attributes`

Object Creation

Mandatory attributes:

- `name`
- `path`

Optional attributes:

- `description`
- `visibility`
- `parent_id`
- `lfs_enabled`

- `request_access_enabled`

Object update

Optional attributes for object update:

- `name`
- `path`
- `description`
- `visibility`
- `lfs_enabled`
- `request_access_enabled`

class `gitlab.v4.objects.GroupMember` (*manager, attrs*)

Bases: `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.GroupMemberManager` (*gl, parent=None*)

Bases: `gitlab.mixins.CRUDMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- `access_level`
- `user_id`

Optional attributes:

- `expires_at`

Object update

Mandatory attributes for object update:

- `access_level`

Optional attributes for object update:

- `expires_at`

all (***kwargs*)

List all the members, included inherited ones.

Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the list could not be retrieved

Returns The list of members

Return type *RESTObjectList*

class gitlab.v4.objects.**GroupMergeRequest** (*manager, attrs*)
 Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**GroupMergeRequestManager** (*gl, parent=None*)
 Bases: *gitlab.mixins.ListMixin, gitlab.base.RESTManager*

Object listing filters

- state
- order_by
- sort
- milestone
- view
- labels
- created_after
- created_before
- updated_after
- updated_before
- scope
- author_id
- assignee_id
- my_reaction_emoji
- source_branch
- target_branch
- search

class gitlab.v4.objects.**GroupMilestone** (*manager, attrs*)
 Bases: *gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

issues (***kwargs*)
 List issues related to this milestone.

Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the list could not be retrieved

Returns The list of issues

Return type *RESTObjectList*

merge_requests (***kwargs*)

List the merge requests related to this milestone.

Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the list could not be retrieved

Returns The list of merge requests

Return type *RESTObjectList*

class `gitlab.v4.objects.GroupMilestoneManager` (*gl, parent=None*)

Bases: `gitlab.mixins.CRUDMixin`, `gitlab.base.RESTManager`

Object listing filters

- `iids`
- `state`
- `search`

Object Creation

Mandatory attributes:

- `title`

Optional attributes:

- `description`
- `due_date`
- `start_date`

Object update

Optional attributes for object update:

- `title`
- `description`
- `due_date`
- `start_date`
- `state_event`

class gitlab.v4.objects.**GroupNotificationSettings** (*manager, attrs*)
Bases: *gitlab.v4.objects.NotificationSettings*

class gitlab.v4.objects.**GroupNotificationSettingsManager** (*gl, parent=None*)
Bases: *gitlab.v4.objects.NotificationSettingsManager*

Object update

Optional attributes for object update:

- level
- notification_email
- new_note
- new_issue
- reopen_issue
- close_issue
- reassign_issue
- new_merge_request
- reopen_merge_request
- close_merge_request
- reassign_merge_request
- merge_merge_request

class gitlab.v4.objects.**GroupProject** (*manager, attrs*)
Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**GroupProjectManager** (*gl, parent=None*)
Bases: *gitlab.mixins.ListMixin, gitlab.base.RESTManager*

Object listing filters

- archived
- visibility
- order_by
- sort
- search
- ci_enabled_first
- simple
- owned
- starred
- with_custom_attributes

class gitlab.v4.objects.**GroupSubgroup** (*manager, attrs*)
Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**GroupSubgroupManager** (*gl, parent=None*)
Bases: *gitlab.mixins.ListMixin, gitlab.base.RESTManager*

Object listing filters

- skip_groups
- all_available
- search
- order_by
- sort
- statistics
- owned
- with_custom_attributes

class gitlab.v4.objects.**GroupVariable** (*manager, attrs*)

Bases: *gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**GroupVariableManager** (*gl, parent=None*)

Bases: *gitlab.mixins.CRUDMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- key
- value

Optional attributes:

- protected

Object update

Mandatory attributes for object update:

- key
- value

Optional attributes for object update:

- protected

class gitlab.v4.objects.**Hook** (*manager, attrs*)

Bases: *gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**HookManager** (*gl, parent=None*)

Bases: *gitlab.mixins.NoUpdateMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- url

class gitlab.v4.objects.**Issue** (*manager, attrs*)

Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**IssueManager** (*gl, parent=None*)

Bases: *gitlab.mixins.ListMixin, gitlab.base.RESTManager*

Object listing filters

- state

- labels
- milestone
- scope
- author_id
- assignee_id
- my_reaction_emoji
- iids
- order_by
- sort
- search
- created_after
- created_before
- updated_after
- updated_before

class gitlab.v4.objects.**LDAPGroup** (*manager, attrs*)

Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**LDAPGroupManager** (*gl, parent=None*)

Bases: *gitlab.base.RESTManager*

Object listing filters

- search
- provider

list (***kwargs*)

Retrieve a list of objects.

Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns The list of objects, or a generator if *as_list* is False

Return type list

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the server cannot perform the request

class gitlab.v4.objects.**License** (*manager, attrs*)

Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**LicenseManager** (*gl, parent=None*)
Bases: *gitlab.mixins.RetrieveMixin, gitlab.base.RESTManager*

Object listing filters

- popular

class gitlab.v4.objects.**MergeRequest** (*manager, attrs*)
Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**MergeRequestManager** (*gl, parent=None*)
Bases: *gitlab.mixins.ListMixin, gitlab.base.RESTManager*

Object listing filters

- state
- order_by
- sort
- milestone
- view
- labels
- created_after
- created_before
- updated_after
- updated_before
- scope
- author_id
- assignee_id
- my_reaction_emoji
- source_branch
- target_branch
- search

class gitlab.v4.objects.**Namespace** (*manager, attrs*)
Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**NamespaceManager** (*gl, parent=None*)
Bases: *gitlab.mixins.RetrieveMixin, gitlab.base.RESTManager*

Object listing filters

- search

class gitlab.v4.objects.**NotificationSettings** (*manager, attrs*)
Bases: *gitlab.mixins.SaveMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**NotificationSettingsManager** (*gl, parent=None*)
Bases: *gitlab.mixins.GetWithoutIdMixin, gitlab.mixins.UpdateMixin, gitlab.base.RESTManager*

Object update

Optional attributes for object update:

- level
- notification_email
- new_note
- new_issue
- reopen_issue
- close_issue
- reassign_issue
- new_merge_request
- reopen_merge_request
- close_merge_request
- reassign_merge_request
- merge_merge_request

class gitlab.v4.objects.**PagesDomain** (*manager, attrs*)

Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**PagesDomainManager** (*gl, parent=None*)

Bases: *gitlab.mixins.ListMixin, gitlab.base.RESTManager*

class gitlab.v4.objects.**Project** (*manager, attrs*)

Bases: *gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

archive (***kwargs*)

Archive a project.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the server failed to perform the request

artifact (***kwargs*)

Download a single artifact file from a specific tag or branch from within the job's artifacts archive.

Parameters

- **ref_name** (*str*) – Branch or tag name in repository. HEAD or SHA references are not supported.
- **artifact_path** (*str*) – Path to a file inside the artifacts archive.
- **job** (*str*) – The name of the job.
- **streamed** (*bool*) – If True the data will be processed by chunks of *chunk_size* and each chunk is passed to *action* for treatment
- **action** (*callable*) – Callable responsible of dealing with chunk of data
- **chunk_size** (*int*) – Size of each chunk
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct

- `GitlabGetError` – If the artifacts could not be retrieved

Returns The artifacts if *streamed* is False, None otherwise.

Return type str

create_fork_relation (**kwargs)

Create a forked from/to relation between existing projects.

Parameters

- **forked_from_id** (*int*) – The ID of the project that was forked from
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the relation could not be created

delete_fork_relation (**kwargs)

Delete a forked relation between existing projects.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabDeleteError` – If the server failed to perform the request

delete_merged_branches (**kwargs)

Delete merged branches.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabDeleteError` – If the server failed to perform the request

housekeeping (**kwargs)

Start the housekeeping task.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabHousekeepingError` – If the server failed to perform the request

languages (**kwargs)

Get languages used in the project with percentage value.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server failed to perform the request

mirror_pull (**kwargs)

Start the pull mirroring process for the project.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the server failed to perform the request

repository_archive (**kwargs)

Return a tarball of the repository.

Parameters

- **sha** (*str*) – ID of the commit (default branch by default)
- **streamed** (*bool*) – If True the data will be processed by chunks of *chunk_size* and each chunk is passed to *action* for treatment
- **action** (*callable*) – Callable responsible of dealing with chunk of data
- **chunk_size** (*int*) – Size of each chunk
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the server failed to perform the request

Returns The binary data of the archive

Return type str

repository_blob (**kwargs)

Return a file by blob SHA.

Parameters

- **sha** (*str*) – ID of the blob
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server failed to perform the request

Returns The blob content and metadata

Return type dict

repository_compare (**kwargs)

Return a diff between two branches/commits.

Parameters

- **from** (*str*) – Source branch/SHA
- **to** (*str*) – Destination branch/SHA
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server failed to perform the request

Returns The diff

Return type str

repository_contributors (**kwargs)

Return a list of contributors for the project.

Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server failed to perform the request

Returns The contributors

Return type list

repository_raw_blob (**kwargs)

Return the raw file contents for a blob.

Parameters

- **sha** (*str*) – ID of the blob
- **streamed** (*bool*) – If True the data will be processed by chunks of *chunk_size* and each chunk is passed to *action* for treatment
- **action** (*callable*) – Callable responsible of dealing with chunk of data
- **chunk_size** (*int*) – Size of each chunk
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server failed to perform the request

Returns The blob content if streamed is False, None otherwise

Return type str

repository_tree (**kwargs)

Return a list of files in the repository.

Parameters

- **path** (*str*) – Path of the top folder (/ by default)
- **ref** (*str*) – Reference to a commit or branch
- **recursive** (*bool*) – Whether to get the tree recursively
- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)

- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server failed to perform the request

Returns The representation of the tree

Return type list

search (***kwargs*)

Search the project resources matching the provided string.’

Parameters

- **scope** (*str*) – Scope of the search
- **search** (*str*) – Search string
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabSearchError` – If the server failed to perform the request

Returns A list of dicts describing the resources found.

Return type *GitlabList*

share (***kwargs*)

Share the project with a group.

Parameters

- **group_id** (*int*) – ID of the group.
- **group_access** (*int*) – Access level for the group.
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the server failed to perform the request

snapshot (***kwargs*)

Return a snapshot of the repository.

Parameters

- **wiki** (*bool*) – If True return the wiki repository
- **streamed** (*bool*) – If True the data will be processed by chunks of *chunk_size* and each chunk is passed to *action* for treatment.
- **action** (*callable*) – Callable responsible of dealing with chunk of data
- **chunk_size** (*int*) – Size of each chunk
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the content could not be retrieved

Returns The uncompressed tar archive of the repository

Return type `str`

star (***kwargs*)

Star a project.

Parameters ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the server failed to perform the request

transfer_project (***kwargs*)

Transfer a project to the given namespace ID

Parameters

- **to_namespace** (*str*) – ID or path of the namespace to transfer the
- **to** (*project*) –
- ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabTransferProjectError` – If the project could not be transferred

trigger_pipeline (***kwargs*)

Trigger a CI build.

See <https://gitlab.com/help/ci/triggers/README.md#trigger-a-build>

Parameters

- **ref** (*str*) – Commit to build; can be a branch name or a tag
- **token** (*str*) – The trigger token
- **variables** (*dict*) – Variables passed to the build script
- ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the server failed to perform the request

unarchive (***kwargs*)

Unarchive a project.

Parameters ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabDeleteError` – If the server failed to perform the request

unshare (***kwargs*)

Delete a shared project link within a group.

Parameters

- **group_id** (*int*) – ID of the group.
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabDeleteError` – If the server failed to perform the request

unstar (***kwargs*)

Unstar a project.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabDeleteError` – If the server failed to perform the request

upload (***kwargs*)

Upload the specified file into the project.

Note: Either `filedata` or `filepath` *MUST* be specified.

Parameters

- **filename** (*str*) – The name of the file being uploaded
- **filedata** (*bytes*) – The raw data of the file being uploaded
- **filepath** (*str*) – The path to a local file to upload (optional)

Raises

- `GitlabConnectionError` – If the server cannot be reached
- `GitlabUploadError` – If the file upload fails
- `GitlabUploadError` – If `filedata` and `filepath` are not specified
- `GitlabUploadError` – If both `filedata` and `filepath` are specified

Returns

A dict with the keys:

- `alt` - The alternate text for the upload
- `url` - The direct url to the uploaded file
- `markdown` - Markdown for the uploaded file

Return type dict

class `gitlab.v4.objects.ProjectAccessRequest` (*manager, attrs*)

Bases: `gitlab.mixins.AccessRequestMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.ProjectAccessRequestManager` (*gl, parent=None*)

Bases: `gitlab.mixins.ListMixin`, `gitlab.mixins.CreateMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

```
class gitlab.v4.objects.ProjectApproval (manager, attrs)
    Bases: gitlab.mixins.SaveMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectApprovalManager (gl, parent=None)
    Bases: gitlab.mixins.GetWithoutIdMixin, gitlab.mixins.UpdateMixin, gitlab.base.RESTManager
```

Object update

Optional attributes for object update:

- `approvals_before_merge`
- `reset_approvals_on_push`
- `disable_overriding_approvers_per_merge_request`

```
set_approvers (**kwargs)
    Change project-level allowed approvers and approver groups.
```

Parameters

- **`approver_ids`** (*list*) – User IDs that can approve MRs
- **`approver_group_ids`** (*list*) – Group IDs whose members can approve MRs

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabUpdateError` – If the server failed to perform the request

```
class gitlab.v4.objects.ProjectBadge (manager, attrs)
    Bases: gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectBadgeManager (gl, parent=None)
    Bases: gitlab.mixins.BadgeRenderMixin, gitlab.mixins.CRUDMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- `link_url`
- `image_url`

Object update

Optional attributes for object update:

- `link_url`
- `image_url`

```
class gitlab.v4.objects.ProjectBoard (manager, attrs)
    Bases: gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectBoardList (manager, attrs)
    Bases: gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectBoardListManager (gl, parent=None)
    Bases: gitlab.mixins.CRUDMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- `label_id`

Object update

Mandatory attributes for object update:

- `position`

```
class gitlab.v4.objects.ProjectBoardManager (gl, parent=None)
    Bases: gitlab.mixins.CRUDMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- `name`

```
class gitlab.v4.objects.ProjectBranch (manager, attrs)
    Bases: gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
protect (**kwargs)
    Protect the branch.
```

Parameters

- **`developers_can_push`** (*bool*) – Set to True if developers are allowed to push to the branch
- **`developers_can_merge`** (*bool*) – Set to True if developers are allowed to merge to the branch
- **`**kwargs`** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabProtectError` – If the branch could not be protected

```
unprotect (**kwargs)
    Unprotect the branch.
```

Parameters **`**kwargs`** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabProtectError` – If the branch could not be unprotected

```
class gitlab.v4.objects.ProjectBranchManager (gl, parent=None)
    Bases: gitlab.mixins.NoUpdateMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- `branch`
- `ref`

```
class gitlab.v4.objects.ProjectCommit (manager, attrs)
    Bases: gitlab.base.RESTObject
```

cherry_pick (**kwargs)

Cherry-pick a commit into a branch.

Parameters

- **branch** (*str*) – Name of target branch
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCherryPickError` – If the cherry-pick could not be performed

diff (**kwargs)

Generate the commit diff.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the diff could not be retrieved

Returns The changes done in this commit

Return type list

merge_requests (**kwargs)

List the merge requests related to the commit.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the references could not be retrieved

Returns The merge requests related to the commit.

Return type list

refs (**kwargs)

List the references the commit is pushed to.

Parameters

- **type** (*str*) – The scope of references ('branch', 'tag' or 'all')
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the references could not be retrieved

Returns The references the commit is pushed to.

Return type list

class `gitlab.v4.objects.ProjectCommitComment` (*manager, attrs*)

Bases: `gitlab.base.RESTObject`


```
class gitlab.v4.objects.ProjectCommitCommentManager (gl, parent=None)
  Bases: gitlab.mixins.ListMixin, gitlab.mixins.CreateMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- note

Optional attributes:

- path
- line
- line_type

```
class gitlab.v4.objects.ProjectCommitDiscussion (manager, attrs)
  Bases: gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectCommitDiscussionManager (gl, parent=None)
  Bases: gitlab.mixins.RetrieveMixin, gitlab.mixins.CreateMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- body

Optional attributes:

- created_at

```
class gitlab.v4.objects.ProjectCommitDiscussionNote (manager, attrs)
  Bases: gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectCommitDiscussionNoteManager (gl, parent=None)
  Bases: gitlab.mixins.GetMixin, gitlab.mixins.CreateMixin, gitlab.mixins.UpdateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- body

Optional attributes:

- created_at
- position

Object update

Mandatory attributes for object update:

- body

```
class gitlab.v4.objects.ProjectCommitManager (gl, parent=None)
  Bases: gitlab.mixins.RetrieveMixin, gitlab.mixins.CreateMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- `branch`
- `commit_message`
- `actions`

Optional attributes:

- `author_email`
- `author_name`

class `gitlab.v4.objects.ProjectCommitStatus` (*manager, attrs*)

Bases: `gitlab.base.RESTObject`, `gitlab.mixins.RefreshMixin`

class `gitlab.v4.objects.ProjectCommitStatusManager` (*gl, parent=None*)

Bases: `gitlab.mixins.ListMixin`, `gitlab.mixins.CreateMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- `state`

Optional attributes:

- `description`
- `name`
- `context`
- `ref`
- `target_url`
- `coverage`

create (***kwargs*)

Create a new object.

Parameters

- **data** (*dict*) – Parameters to send to the server to create the resource
- ****kwargs** – Extra options to send to the server (e.g. `sudo` or `'ref_name'`, `'stage'`, `'name'`, `'all'`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the server cannot perform the request

Returns

A new instance of the manage object class build with the data sent by the server

Return type `RESTObject`

class `gitlab.v4.objects.ProjectCustomAttribute` (*manager, attrs*)

Bases: `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.ProjectCustomAttributeManager` (*gl, parent=None*)

Bases: `gitlab.mixins.RetrieveMixin`, `gitlab.mixins.SetMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

class `gitlab.v4.objects.ProjectDeployment` (*manager, attrs*)
 Bases: `gitlab.base.RESTObject`

class `gitlab.v4.objects.ProjectDeploymentManager` (*gl, parent=None*)
 Bases: `gitlab.mixins.RetrieveMixin`, `gitlab.base.RESTManager`

Object listing filters

- `order_by`
- `sort`

class `gitlab.v4.objects.ProjectEnvironment` (*manager, attrs*)
 Bases: `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

stop (***kwargs*)

Stop the environment.

Parameters ***kwargs* – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabStopError` – If the operation failed

class `gitlab.v4.objects.ProjectEnvironmentManager` (*gl, parent=None*)
 Bases: `gitlab.mixins.RetrieveMixin`, `gitlab.mixins.CreateMixin`, `gitlab.mixins.UpdateMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- `name`

Optional attributes:

- `external_url`

Object update

Optional attributes for object update:

- `name`
- `external_url`

class `gitlab.v4.objects.ProjectEvent` (*manager, attrs*)
 Bases: `gitlab.v4.objects.Event`

class `gitlab.v4.objects.ProjectEventManager` (*gl, parent=None*)
 Bases: `gitlab.v4.objects.EventManager`

Object listing filters

- `action`
- `target_type`
- `before`
- `after`
- `sort`

class gitlab.v4.objects.**ProjectExport** (*manager, attrs*)
Bases: *gitlab.mixins.RefreshMixin, gitlab.base.RESTObject*

download (***kwargs*)
Download the archive of a project export.

Parameters

- **streamed** (*bool*) – If True the data will be processed by chunks of *chunk_size* and each chunk is passed to *action* for treatment
- **action** (*callable*) – Callable responsible of dealing with chunk of data
- **chunk_size** (*int*) – Size of each chunk
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server failed to perform the request

Returns The blob content if streamed is False, None otherwise

Return type str

class gitlab.v4.objects.**ProjectExportManager** (*gl, parent=None*)
Bases: *gitlab.mixins.GetWithoutIdMixin, gitlab.mixins.CreateMixin, gitlab.base.RESTManager*

Object Creation

Optional attributes:

- *description*

class gitlab.v4.objects.**ProjectFile** (*manager, attrs*)
Bases: *gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

decode ()
Returns the decoded content of the file.

Returns the decoded content.

Return type (str)

delete (*branch, commit_message, **kwargs*)
Delete the file from the server.

Parameters

- **branch** (*str*) – Branch from which the file will be removed
- **commit_message** (*str*) – Commit message for the deletion
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabDeleteError` – If the server cannot perform the request

save (*branch, commit_message, **kwargs*)
Save the changes made to the file to the server.

The object is updated to match what the server returns.

Parameters

- **branch** (*str*) – Branch in which the file will be updated
- **commit_message** (*str*) – Message to send with the commit
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabUpdateError` – If the server cannot perform the request

class `gitlab.v4.objects.ProjectFileManager` (*gl*, *parent=None*)

Bases: `gitlab.mixins.GetMixin`, `gitlab.mixins.CreateMixin`, `gitlab.mixins.UpdateMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- `file_path`
- `branch`
- `content`
- `commit_message`

Optional attributes:

- `encoding`
- `author_email`
- `author_name`

Object update

Mandatory attributes for object update:

- `file_path`
- `branch`
- `content`
- `commit_message`

Optional attributes for object update:

- `encoding`
- `author_email`
- `author_name`

create (****kwargs**)

Create a new object.

Parameters

- **data** (*dict*) – parameters to send to the server to create the resource
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns

a new instance of the managed object class built with the data sent by the server

Return type *RESTObject*

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the server cannot perform the request

delete (***kwargs*)

Delete a file on the server.

Parameters

- **file_path** (*str*) – Path of the file to remove
- **branch** (*str*) – Branch from which the file will be removed
- **commit_message** (*str*) – Commit message for the deletion
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabDeleteError` – If the server cannot perform the request

get (***kwargs*)

Retrieve a single file.

Parameters

- **file_path** (*str*) – Path of the file to retrieve
- **ref** (*str*) – Name of the branch, tag or commit
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the file could not be retrieved

Returns The generated `RESTObject`

Return type `object`

raw (***kwargs*)

Return the content of a file for a commit.

Parameters

- **ref** (*str*) – ID of the commit
- **filepath** (*str*) – Path of the file to return
- **streamed** (*bool*) – If True the data will be processed by chunks of *chunk_size* and each chunk is passed to *action* for treatment
- **action** (*callable*) – Callable responsible of dealing with chunk of data
- **chunk_size** (*int*) – Size of each chunk
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the file could not be retrieved

Returns The file content

Return type `str`

update (***kwargs*)

Update an object on the server.

Parameters

- `id` – ID of the object to update (can be `None` if not required)
- `new_data` – the update data for the object
- ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Returns The new object data (*not* a `RESTObject`)

Return type `dict`

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabUpdateError` – If the server cannot perform the request

class `gitlab.v4.objects.ProjectFork` (*manager, attrs*)

Bases: `gitlab.base.RESTObject`

class `gitlab.v4.objects.ProjectForkManager` (*gl, parent=None*)

Bases: `gitlab.mixins.CreateMixin`, `gitlab.mixins.ListMixin`, `gitlab.base.RESTManager`

Object listing filters

- `archived`
- `visibility`
- `order_by`
- `sort`
- `search`
- `simple`
- `owned`
- `membership`
- `starred`
- `statistics`
- `with_custom_attributes`
- `with_issues_enabled`
- `with_merge_requests_enabled`

Object Creation

Optional attributes:

- `namespace`

list (**kwargs)

Retrieve a list of objects.

Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns The list of objects, or a generator if *as_list* is False

Return type list

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the server cannot perform the request

class `gitlab.v4.objects.ProjectHook` (*manager, attrs*)

Bases: `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.ProjectHookManager` (*gl, parent=None*)

Bases: `gitlab.mixins.CRUDMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- `url`

Optional attributes:

- `push_events`
- `issues_events`
- `confidential_issues_events`
- `merge_requests_events`
- `tag_push_events`
- `note_events`
- `job_events`
- `pipeline_events`
- `wiki_page_events`
- `enable_ssl_verification`
- `token`

Object update

Mandatory attributes for object update:

- `url`

Optional attributes for object update:

- `push_events`
- `issues_events`
- `confidential_issues_events`
- `merge_requests_events`
- `tag_push_events`
- `note_events`
- `job_events`
- `pipeline_events`
- `wiki_events`
- `enable_ssl_verification`
- `token`

class `gitlab.v4.objects.ProjectImport` (*manager, attrs*)
 Bases: `gitlab.mixins.RefreshMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.ProjectImportManager` (*gl, parent=None*)
 Bases: `gitlab.mixins.GetWithoutIdMixin`, `gitlab.base.RESTManager`

class `gitlab.v4.objects.ProjectIssue` (*manager, attrs*)
 Bases: `gitlab.mixins.UserAgentDetailMixin`, `gitlab.mixins.SubscribableMixin`,
`gitlab.mixins.TODOMixin`, `gitlab.mixins.TimeTrackingMixin`, `gitlab.mixins.ParticipantsMixin`,
`gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`,
`gitlab.base.RESTObject`

closed_by (***kwargs*)

List merge requests that will close the issue when merged.

Parameters ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the merge requests could not be retrieved

Returns The list of merge requests.

Return type list

move (***kwargs*)

Move the issue to another project.

Parameters

- **to_project_id** (*int*) – ID of the target project
- ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabUpdateError` – If the issue could not be moved

related_merge_requests (***kwargs*)

List merge requests related to the issue.

Parameters ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the merge requests could not be retrieved

Returns The list of merge requests.

Return type list

```
class gitlab.v4.objects.ProjectIssueAwardEmoji (manager, attrs)
    Bases: gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectIssueAwardEmojiManager (gl, parent=None)
    Bases: gitlab.mixins.NoUpdateMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- name

```
class gitlab.v4.objects.ProjectIssueDiscussion (manager, attrs)
    Bases: gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectIssueDiscussionManager (gl, parent=None)
    Bases: gitlab.mixins.RetrieveMixin, gitlab.mixins.CreateMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- body

Optional attributes:

- created_at

```
class gitlab.v4.objects.ProjectIssueDiscussionNote (manager, attrs)
    Bases: gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectIssueDiscussionNoteManager (gl, parent=None)
    Bases: gitlab.mixins.GetMixin, gitlab.mixins.CreateMixin, gitlab.mixins.UpdateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- body

Optional attributes:

- created_at

Object update

Mandatory attributes for object update:

- body

```
class gitlab.v4.objects.ProjectIssueLink (manager, attrs)
    Bases: gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

class `gitlab.v4.objects.ProjectIssueLinkManager` (*gl*, *parent=None*)
 Bases: `gitlab.mixins.ListMixin`, `gitlab.mixins.CreateMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- `target_project_id`
- `target_issue_iid`

create (***kwargs*)
 Create a new object.

Parameters

- **data** (*dict*) – parameters to send to the server to create the resource
- ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Returns The source and target issues

Return type `RESTObject`, `RESTObject`

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the server cannot perform the request

class `gitlab.v4.objects.ProjectIssueManager` (*gl*, *parent=None*)
 Bases: `gitlab.mixins.CRUDMixin`, `gitlab.base.RESTManager`

Object listing filters

- `iids`
- `state`
- `labels`
- `milestone`
- `scope`
- `author_id`
- `assignee_id`
- `my_reaction_emoji`
- `order_by`
- `sort`
- `search`
- `created_after`
- `created_before`
- `updated_after`
- `updated_before`

Object Creation

Mandatory attributes:

- title

Optional attributes:

- description
- confidential
- assignee_ids
- assignee_id
- milestone_id
- labels
- created_at
- due_date
- merge_request_to_resolve_discussions_of
- discussion_to_resolve

Object update

Optional attributes for object update:

- title
- description
- confidential
- assignee_ids
- assignee_id
- milestone_id
- labels
- state_event
- updated_at
- due_date
- discussion_locked

class gitlab.v4.objects.**ProjectIssueNote** (*manager, attrs*)

Bases: *gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**ProjectIssueNoteAwardEmoji** (*manager, attrs*)

Bases: *gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**ProjectIssueNoteAwardEmojiManager** (*gl, parent=None*)

Bases: *gitlab.mixins.NoUpdateMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- name

class gitlab.v4.objects.**ProjectIssueNoteManager** (*gl, parent=None*)

Bases: *gitlab.mixins.CRUDMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- `body`

Optional attributes:

- `created_at`

Object update

Mandatory attributes for object update:

- `body`

```
class gitlab.v4.objects.ProjectIssueResourceLabelEvent (manager, attrs)
```

Bases: `gitlab.base.RESTObject`

```
class gitlab.v4.objects.ProjectIssueResourceLabelEventManager (gl, parent=None)
```

Bases: `gitlab.mixins.RetrieveMixin`, `gitlab.base.RESTManager`

```
class gitlab.v4.objects.ProjectJob (manager, attrs)
```

Bases: `gitlab.base.RESTObject`, `gitlab.mixins.RefreshMixin`

```
artifact (**kwargs)
```

Get a single artifact file from within the job's artifacts archive.

Parameters

- **path** (*str*) – Path of the artifact
- **streamed** (*bool*) – If True the data will be processed by chunks of *chunk_size* and each chunk is passed to *action* for treatment
- **action** (*callable*) – Callable responsible of dealing with chunk of data
- **chunk_size** (*int*) – Size of each chunk
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the artifacts could not be retrieved

Returns The artifacts if *streamed* is False, None otherwise.

Return type `str`

```
artifacts (**kwargs)
```

Get the job artifacts.

Parameters

- **streamed** (*bool*) – If True the data will be processed by chunks of *chunk_size* and each chunk is passed to *action* for treatment
- **action** (*callable*) – Callable responsible of dealing with chunk of data
- **chunk_size** (*int*) – Size of each chunk
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the artifacts could not be retrieved

Returns The artifacts if *streamed* is False, None otherwise.

Return type str

cancel (**kwargs)

Cancel the job.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabJobCancelError` – If the job could not be canceled

delete_artifacts (**kwargs)

Delete artifacts of a job.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabDeleteError` – If the request could not be performed

erase (**kwargs)

Erase the job (remove job artifacts and trace).

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabJobEraseError` – If the job could not be erased

keep_artifacts (**kwargs)

Prevent artifacts from being deleted when expiration is set.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the request could not be performed

play (**kwargs)

Trigger a job explicitly.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabJobPlayError` – If the job could not be triggered

retry (**kwargs)

Retry the job.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabJobRetryError` – If the job could not be retried

trace (**kwargs)
Get the job trace.

Parameters

- **streamed** (*bool*) – If True the data will be processed by chunks of *chunk_size* and each chunk is passed to *action* for treatment
- **action** (*callable*) – Callable responsible of dealing with chunk of data
- **chunk_size** (*int*) – Size of each chunk
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the artifacts could not be retrieved

Returns The trace

Return type str

```
class gitlab.v4.objects.ProjectJobManager (gl, parent=None)
```

Bases: `gitlab.mixins.RetrieveMixin`, `gitlab.base.RESTManager`

```
class gitlab.v4.objects.ProjectKey (manager, attrs)
```

Bases: `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

```
class gitlab.v4.objects.ProjectKeyManager (gl, parent=None)
```

Bases: `gitlab.mixins.CRUDMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- title
- key

Optional attributes:

- can_push

Object update

Optional attributes for object update:

- title
- can_push

enable (**kwargs)
Enable a deploy key for a project.

Parameters

- **key_id** (*int*) – The ID of the key to enable
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabProjectDeployKeyError` – If the key could not be enabled

```
class gitlab.v4.objects.ProjectLabel (manager, attrs)  
    Bases: gitlab.mixins.SubscribableMixin, gitlab.mixins.SaveMixin, gitlab.  
mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
save (**kwargs)
```

Saves the changes made to the object to the server.

The object is updated to match what the server returns.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct.
- `GitlabUpdateError` – If the server cannot perform the request.

```
class gitlab.v4.objects.ProjectLabelManager (gl, parent=None)
```

```
    Bases: gitlab.mixins.ListMixin, gitlab.mixins.CreateMixin, gitlab.mixins.  
UpdateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- `name`
- `color`

Optional attributes:

- `description`
- `priority`

Object update

Mandatory attributes for object update:

- `name`

Optional attributes for object update:

- `new_name`
- `color`
- `description`
- `priority`

```
delete (**kwargs)
```

Delete a Label on the server.

Parameters

- **name** – The name of the label
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabDeleteError` – If the server cannot perform the request

```
update (name, new_data=None, **kwargs)
```

Update a Label on the server.

Parameters

- **name** – The name of the label
- ****kwargs** – Extra options to send to the server (e.g. sudo)

class gitlab.v4.objects.**ProjectManager** (*gl, parent=None*)
Bases: *gitlab.mixins.CRUDMixin, gitlab.base.RESTManager*

Object listing filters

- search
- owned
- starred
- archived
- visibility
- order_by
- sort
- simple
- membership
- statistics
- with_issues_enabled
- with_merge_requests_enabled
- with_custom_attributes

Object Creation

Optional attributes:

- name
- path
- namespace_id
- description
- issues_enabled
- merge_requests_enabled
- jobs_enabled
- wiki_enabled
- snippets_enabled
- resolve_outdated_diff_discussions
- container_registry_enabled
- shared_runners_enabled
- visibility
- import_url
- public_jobs

- `only_allow_merge_if_pipeline_succeeds`
- `only_allow_merge_if_all_discussions_are_resolved`
- `merge_method`
- `lfs_enabled`
- `request_access_enabled`
- `tag_list`
- `avatar`
- `printing_merge_request_link_enabled`
- `ci_config_path`

Object update

Optional attributes for object update:

- `name`
- `path`
- `default_branch`
- `description`
- `issues_enabled`
- `merge_requests_enabled`
- `jobs_enabled`
- `wiki_enabled`
- `snippets_enabled`
- `resolve_outdated_diff_discussions`
- `container_registry_enabled`
- `shared_runners_enabled`
- `visibility`
- `import_url`
- `public_jobs`
- `only_allow_merge_if_pipeline_succeeds`
- `only_allow_merge_if_all_discussions_are_resolved`
- `merge_method`
- `lfs_enabled`
- `request_access_enabled`
- `tag_list`
- `avatar`
- `ci_config_path`

import_project (*file, path, namespace=None, overwrite=False, override_params=None, **kwargs*)
Import a project from an archive file.

Parameters

- **file** – Data or file object containing the project
- **path** (*str*) – Name and path for the new project
- **namespace** (*str*) – The ID or path of the namespace that the project will be imported to
- **overwrite** (*bool*) – If True overwrite an existing project with the same path
- **override_params** (*dict*) – Set the specific settings for the project
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the server failed to perform the request

Returns A representation of the import status.

Return type dict

```
class gitlab.v4.objects.ProjectMember (manager, attrs)
    Bases: gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectMemberManager (gl, parent=None)
    Bases: gitlab.mixins.CRUDMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- `access_level`
- `user_id`

Optional attributes:

- `expires_at`

Object update

Mandatory attributes for object update:

- `access_level`

Optional attributes for object update:

- `expires_at`

all (***kwargs*)

List all the members, included inherited ones.

Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the list could not be retrieved

Returns The list of members**Return type** *RESTObjectList***class** `gitlab.v4.objects.ProjectMergeRequest` (*manager, attrs*)**Bases:** `gitlab.mixins.SubscribableMixin`, `gitlab.mixins.TODOMixin`, `gitlab.mixins.TimeTrackingMixin`, `gitlab.mixins.ParticipantsMixin`, `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`**approve** (***kwargs*)

Approve the merge request.

Parameters

- **sha** (*str*) – Head SHA of MR
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabMRApprovalError` – If the approval failed

cancel_merge_when_pipeline_succeeds (***kwargs*)

Cancel merge when the pipeline succeeds.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)**Raises**

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabMROnBuildSuccessError` – If the server could not handle the request

changes (***kwargs*)

List the merge request changes.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)**Raises**

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the list could not be retrieved

Returns List of changes**Return type** *RESTObjectList***closes_issues** (***kwargs*)

List issues that will close on merge.”

Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list

- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the list could not be retrieved

Returns List of issues

Return type *RESTObjectList*

commits (***kwargs*)

List the merge request commits.

Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the list could not be retrieved

Returns The list of commits

Return type *RESTObjectList*

merge (***kwargs*)

Accept the merge request.

Parameters

- **merge_commit_message** (*bool*) – Commit message
- **should_remove_source_branch** (*bool*) – If True, removes the source branch
- **merge_when_pipeline_succeeds** (*bool*) – Wait for the build to succeed, then merge
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabMRClosedError` – If the merge failed

pipelines (***kwargs*)

List the merge request pipelines.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the list could not be retrieved

Returns List of changes

Return type *RESTObjectList*

rebase (***kwargs*)

Attempt to rebase the source branch onto the target branch

Parameters ***kwargs* – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabMRRebaseError` – If rebasing failed

unapprove (***kwargs*)

Unapprove the merge request.

Parameters ***kwargs* – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabMRApprovalError` – If the unapproval failed

class `gitlab.v4.objects.ProjectMergeRequestApproval` (*manager, attrs*)

Bases: `gitlab.mixins.SaveMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.ProjectMergeRequestApprovalManager` (*gl, parent=None*)

Bases: `gitlab.mixins.GetWithoutIdMixin`, `gitlab.mixins.UpdateMixin`, `gitlab.base.RESTManager`

Object update

Mandatory attributes for object update:

- `approvals_required`

set_approvers (***kwargs*)

Change MR-level allowed approvers and approver groups.

Parameters

- **approver_ids** (*list*) – User IDs that can approve MRs
- **approver_group_ids** (*list*) – Group IDs whose members can approve MRs

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabUpdateError` – If the server failed to perform the request

class `gitlab.v4.objects.ProjectMergeRequestAwardEmoji` (*manager, attrs*)

Bases: `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.ProjectMergeRequestAwardEmojiManager` (*gl, parent=None*)

Bases: `gitlab.mixins.NoUpdateMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- `name`

class `gitlab.v4.objects.ProjectMergeRequestDiff` (*manager, attrs*)

Bases: `gitlab.base.RESTObject`

```
class gitlab.v4.objects.ProjectMergeRequestDiffManager (gl, parent=None)
    Bases: gitlab.mixins.RetrieveMixin, gitlab.base.RESTManager
```

```
class gitlab.v4.objects.ProjectMergeRequestDiscussion (manager, attrs)
    Bases: gitlab.mixins.SaveMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectMergeRequestDiscussionManager (gl, parent=None)
    Bases: gitlab.mixins.RetrieveMixin, gitlab.mixins.CreateMixin, gitlab.mixins.UpdateMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- `body`

Optional attributes:

- `created_at`
- `position`

Object update

Mandatory attributes for object update:

- `resolved`

```
class gitlab.v4.objects.ProjectMergeRequestDiscussionNote (manager, attrs)
    Bases: gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectMergeRequestDiscussionNoteManager (gl, parent=None)
    Bases: gitlab.mixins.GetMixin, gitlab.mixins.CreateMixin, gitlab.mixins.UpdateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- `body`

Optional attributes:

- `created_at`

Object update

Mandatory attributes for object update:

- `body`

```
class gitlab.v4.objects.ProjectMergeRequestManager (gl, parent=None)
    Bases: gitlab.mixins.CRUDMixin, gitlab.base.RESTManager
```

Object listing filters

- `state`
- `order_by`
- `sort`
- `milestone`
- `view`
- `labels`

- created_after
- created_before
- updated_after
- updated_before
- scope
- author_id
- assignee_id
- my_reaction_emoji
- source_branch
- target_branch
- search

Object Creation

Mandatory attributes:

- source_branch
- target_branch
- title

Optional attributes:

- assignee_id
- description
- target_project_id
- labels
- milestone_id
- remove_source_branch
- allow_maintainer_to_push
- squash

Object update

Optional attributes for object update:

- target_branch
- assignee_id
- title
- description
- state_event
- labels
- milestone_id
- remove_source_branch
- discussion_locked

- `allow_maintainer_to_push`
- `squash`

class `gitlab.v4.objects.ProjectMergeRequestNote` (*manager, attrs*)
 Bases: `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.ProjectMergeRequestNoteAwardEmoji` (*manager, attrs*)
 Bases: `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.ProjectMergeRequestNoteAwardEmojiManager` (*gl, parent=None*)
 Bases: `gitlab.mixins.NoUpdateMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- `name`

class `gitlab.v4.objects.ProjectMergeRequestNoteManager` (*gl, parent=None*)
 Bases: `gitlab.mixins.CRUDMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- `body`

Object update

Mandatory attributes for object update:

- `body`

class `gitlab.v4.objects.ProjectMergeRequestResourceLabelEvent` (*manager, attrs*)
 Bases: `gitlab.base.RESTObject`

class `gitlab.v4.objects.ProjectMergeRequestResourceLabelEventManager` (*gl, parent=None*)
 Bases: `gitlab.mixins.RetrieveMixin`, `gitlab.base.RESTManager`

class `gitlab.v4.objects.ProjectMilestone` (*manager, attrs*)
 Bases: `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

issues (***kwargs*)

List issues related to this milestone.

Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the list could not be retrieved

Returns The list of issues

Return type *RESTObjectList*

merge_requests (***kwargs*)

List the merge requests related to this milestone.

Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the list could not be retrieved

Returns The list of merge requests

Return type *RESTObjectList*

class `gitlab.v4.objects.ProjectMilestoneManager` (*gl, parent=None*)

Bases: `gitlab.mixins.CRUDMixin`, `gitlab.base.RESTManager`

Object listing filters

- `iids`
- `state`
- `search`

Object Creation

Mandatory attributes:

- `title`

Optional attributes:

- `description`
- `due_date`
- `start_date`
- `state_event`

Object update

Optional attributes for object update:

- `title`
- `description`
- `due_date`
- `start_date`
- `state_event`

class gitlab.v4.objects.**ProjectNote** (*manager, attrs*)

Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**ProjectNoteManager** (*gl, parent=None*)

Bases: *gitlab.mixins.RetrieveMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- body

class gitlab.v4.objects.**ProjectNotificationSettings** (*manager, attrs*)

Bases: *gitlab.v4.objects.NotificationSettings*

class gitlab.v4.objects.**ProjectNotificationSettingsManager** (*gl, parent=None*)

Bases: *gitlab.v4.objects.NotificationSettingsManager*

Object update

Optional attributes for object update:

- level
- notification_email
- new_note
- new_issue
- reopen_issue
- close_issue
- reassign_issue
- new_merge_request
- reopen_merge_request
- close_merge_request
- reassign_merge_request
- merge_merge_request

class gitlab.v4.objects.**ProjectPagesDomain** (*manager, attrs*)

Bases: *gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**ProjectPagesDomainManager** (*gl, parent=None*)

Bases: *gitlab.mixins.CRUDMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- domain

Optional attributes:

- certificate
- key

Object update

Optional attributes for object update:

- certificate
- key

class gitlab.v4.objects.**ProjectPipeline** (*manager, attrs*)

Bases: *gitlab.base.RESTObject, gitlab.mixins.RefreshMixin, gitlab.mixins.ObjectDeleteMixin*

cancel (***kwargs*)

Cancel the job.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabPipelineCancelError` – If the request failed

retry (***kwargs*)

Retry the job.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabPipelineRetryError` – If the request failed

class gitlab.v4.objects.**ProjectPipelineJob** (*manager, attrs*)

Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**ProjectPipelineJobManager** (*gl, parent=None*)

Bases: *gitlab.mixins.ListMixin, gitlab.base.RESTManager*

Object listing filters

- scope

class gitlab.v4.objects.**ProjectPipelineManager** (*gl, parent=None*)

Bases: *gitlab.mixins.RetrieveMixin, gitlab.mixins.CreateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager*

Object listing filters

- scope
- status
- ref
- sha
- yaml_errors
- name
- username
- order_by
- sort

Object Creation

Mandatory attributes:

- ref

create (*data*, ***kwargs*)
Creates a new object.

Parameters

- **data** (*dict*) – Parameters to send to the server to create the resource
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the server cannot perform the request

Returns

A new instance of the managed object class build with the data sent by the server

Return type *RESTObject*

class `gitlab.v4.objects.ProjectPipelineSchedule` (*manager*, *attrs*)
Bases: `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

take_ownership (***kwargs*)
Update the owner of a pipeline schedule.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabOwnershipError` – If the request failed

class `gitlab.v4.objects.ProjectPipelineScheduleManager` (*gl*, *parent=None*)
Bases: `gitlab.mixins.CRUDMixin`, `gitlab.base.RESTManager`

Object Creation

Mandatory attributes:

- `description`
- `ref`
- `cron`

Optional attributes:

- `cron_timezone`
- `active`

Object update

Optional attributes for object update:

- `description`
- `ref`
- `cron`
- `cron_timezone`
- `active`

```
class gitlab.v4.objects.ProjectPipelineScheduleVariable (manager, attrs)  
    Bases: gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectPipelineScheduleVariableManager (gl, parent=None)  
    Bases: gitlab.mixins.CreateMixin, gitlab.mixins.UpdateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- key
- value

Object update

Mandatory attributes for object update:

- key
- value

```
class gitlab.v4.objects.ProjectPipelineVariable (manager, attrs)  
    Bases: gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectPipelineVariableManager (gl, parent=None)  
    Bases: gitlab.mixins.ListMixin, gitlab.base.RESTManager
```

```
class gitlab.v4.objects.ProjectProtectedBranch (manager, attrs)  
    Bases: gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectProtectedBranchManager (gl, parent=None)  
    Bases: gitlab.mixins.NoUpdateMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- name

Optional attributes:

- push_access_level
- merge_access_level
- unprotect_access_level
- allowed_to_push
- allowed_to_merge
- allowed_to_unprotect

```
class gitlab.v4.objects.ProjectProtectedTag (manager, attrs)  
    Bases: gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectProtectedTagManager (gl, parent=None)  
    Bases: gitlab.mixins.NoUpdateMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- name

Optional attributes:

- `create_access_level`

class `gitlab.v4.objects.ProjectPushRules` (*manager, attrs*)

Bases: `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.ProjectPushRulesManager` (*gl, parent=None*)

Bases: `gitlab.mixins.GetWithoutIdMixin`, `gitlab.mixins.CreateMixin`, `gitlab.mixins.UpdateMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

Object Creation

Optional attributes:

- `deny_delete_tag`
- `member_check`
- `prevent_secrets`
- `commit_message_regex`
- `branch_name_regex`
- `author_email_regex`
- `file_name_regex`
- `max_file_size`

Object update

Optional attributes for object update:

- `deny_delete_tag`
- `member_check`
- `prevent_secrets`
- `commit_message_regex`
- `branch_name_regex`
- `author_email_regex`
- `file_name_regex`
- `max_file_size`

class `gitlab.v4.objects.ProjectRegistryRepository` (*manager, attrs*)

Bases: `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.ProjectRegistryRepositoryManager` (*gl, parent=None*)

Bases: `gitlab.mixins.DeleteMixin`, `gitlab.mixins.ListMixin`, `gitlab.base.RESTManager`

class `gitlab.v4.objects.ProjectRegistryTag` (*manager, attrs*)

Bases: `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

class `gitlab.v4.objects.ProjectRegistryTagManager` (*gl, parent=None*)

Bases: `gitlab.mixins.DeleteMixin`, `gitlab.mixins.RetrieveMixin`, `gitlab.base.RESTManager`

delete_in_bulk (***kwargs*)

Delete Tag in bulk

Parameters

- **name_regex** (*string*) – The regex of the name to delete. To delete all tags specify `.*`.
- **keep_n** (*integer*) – The amount of latest tags of given name to keep.
- **older_than** (*string*) – Tags to delete that are older than the given time, written in human readable form 1h, 1d, 1month.
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabDeleteError` – If the server cannot perform the request

```
class gitlab.v4.objects.ProjectRelease (manager, attrs)  
    Bases: gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectReleaseManager (gl, parent=None)  
    Bases: gitlab.mixins.NoUpdateMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- name
- tag_name
- description

Optional attributes:

- ref
- assets

```
class gitlab.v4.objects.ProjectRunner (manager, attrs)  
    Bases: gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectRunnerManager (gl, parent=None)  
    Bases: gitlab.mixins.NoUpdateMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- runner_id

```
class gitlab.v4.objects.ProjectService (manager, attrs)  
    Bases: gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectServiceManager (gl, parent=None)  
    Bases: gitlab.mixins.GetMixin, gitlab.mixins.UpdateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager
```

```
available (**kwargs)
```

List the services known by python-gitlab.

Returns The list of service code names.

Return type list (str)

```
get (id, **kwargs)
```

Retrieve a single object.

Parameters

- **id** (*int or str*) – ID of the object to retrieve
- **lazy** (*bool*) – If True, don't request the server, but create a shallow object giving access to the managers. This is useful if you want to avoid useless calls to the API.
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns The generated RESTObject.

Return type object

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server cannot perform the request

update (*id=None, new_data=None, **kwargs*)

Update an object on the server.

Parameters

- **id** – ID of the object to update (can be None if not required)
- **new_data** – the update data for the object
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns The new object data (*not* a RESTObject)

Return type dict

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabUpdateError` – If the server cannot perform the request

class `gitlab.v4.objects.ProjectSnippet` (*manager, attrs*)

Bases: `gitlab.mixins.UserAgentDetailMixin`, `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

content (***kwargs*)

Return the content of a snippet.

Parameters

- **streamed** (*bool*) – If True the data will be processed by chunks of *chunk_size* and each chunk is passed to *action* for treatment.
- **action** (*callable*) – Callable responsible of dealing with chunk of data
- **chunk_size** (*int*) – Size of each chunk
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the content could not be retrieved

Returns The snippet content

Return type str

class gitlab.v4.objects.**ProjectSnippetAwardEmoji** (*manager, attrs*)
Bases: *gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**ProjectSnippetAwardEmojiManager** (*gl, parent=None*)
Bases: *gitlab.mixins.NoUpdateMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- name

class gitlab.v4.objects.**ProjectSnippetDiscussion** (*manager, attrs*)
Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**ProjectSnippetDiscussionManager** (*gl, parent=None*)
Bases: *gitlab.mixins.RetrieveMixin, gitlab.mixins.CreateMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- body

Optional attributes:

- created_at

class gitlab.v4.objects.**ProjectSnippetDiscussionNote** (*manager, attrs*)
Bases: *gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**ProjectSnippetDiscussionNoteManager** (*gl, parent=None*)
Bases: *gitlab.mixins.GetMixin, gitlab.mixins.CreateMixin, gitlab.mixins.UpdateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- body

Optional attributes:

- created_at

Object update

Mandatory attributes for object update:

- body

class gitlab.v4.objects.**ProjectSnippetManager** (*gl, parent=None*)
Bases: *gitlab.mixins.CRUDMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- title
- file_name
- code

Optional attributes:

- lifetime
- visibility

Object update

Optional attributes for object update:

- title
- file_name
- code
- visibility

```
class gitlab.v4.objects.ProjectSnippetNote (manager, attrs)
    Bases: gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectSnippetNoteAwardEmoji (manager, attrs)
    Bases: gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.ProjectSnippetNoteAwardEmojiManager (gl, parent=None)
    Bases: gitlab.mixins.NoUpdateMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- name

```
class gitlab.v4.objects.ProjectSnippetNoteManager (gl, parent=None)
    Bases: gitlab.mixins.CRUDMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- body

Object update

Mandatory attributes for object update:

- body

```
class gitlab.v4.objects.ProjectTag (manager, attrs)
    Bases: gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
set_release_description (**kwargs)
```

Set the release notes on the tag.

If the release doesn't exist yet, it will be created. If it already exists, its description will be updated.

Parameters

- **description** (*str*) – Description of the release.
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the server fails to create the release
- `GitlabUpdateError` – If the server fails to update the release

class gitlab.v4.objects.**ProjectTagManager** (*gl, parent=None*)
Bases: *gitlab.mixins.NoUpdateMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- tag_name
- ref

Optional attributes:

- message

class gitlab.v4.objects.**ProjectTrigger** (*manager, attrs*)
Bases: *gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

take_ownership (***kwargs*)
Update the owner of a trigger.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- GitlabAuthenticationError – If authentication is not correct
- GitlabOwnershipError – If the request failed

class gitlab.v4.objects.**ProjectTriggerManager** (*gl, parent=None*)
Bases: *gitlab.mixins.CRUDMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- description

Object update

Mandatory attributes for object update:

- description

class gitlab.v4.objects.**ProjectUser** (*manager, attrs*)
Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**ProjectUserManager** (*gl, parent=None*)
Bases: *gitlab.mixins.ListMixin, gitlab.base.RESTManager*

Object listing filters

- search

class gitlab.v4.objects.**ProjectVariable** (*manager, attrs*)
Bases: *gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**ProjectVariableManager** (*gl, parent=None*)
Bases: *gitlab.mixins.CRUDMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- key

- value

Object update

Mandatory attributes for object update:

- key
- value

class gitlab.v4.objects.**ProjectWiki** (*manager, attrs*)

Bases: *gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**ProjectWikiManager** (*gl, parent=None*)

Bases: *gitlab.mixins.CRUDMixin, gitlab.base.RESTManager*

Object listing filters

- with_content

Object Creation

Mandatory attributes:

- title
- content

Optional attributes:

- format

Object update

Optional attributes for object update:

- title
- content
- format

class gitlab.v4.objects.**Runner** (*manager, attrs*)

Bases: *gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**RunnerJob** (*manager, attrs*)

Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**RunnerJobManager** (*gl, parent=None*)

Bases: *gitlab.mixins.ListMixin, gitlab.base.RESTManager*

Object listing filters

- status

class gitlab.v4.objects.**RunnerManager** (*gl, parent=None*)

Bases: *gitlab.mixins.CRUDMixin, gitlab.base.RESTManager*

Object listing filters

- scope

Object Creation

Mandatory attributes:

- token

Optional attributes:

- `description`
- `info`
- `active`
- `locked`
- `run_untagged`
- `tag_list`
- `maximum_timeout`

Object update

Optional attributes for object update:

- `description`
- `active`
- `tag_list`
- `run_untagged`
- `locked`
- `access_level`
- `maximum_timeout`

all (***kwargs*)

List all the runners.

Parameters

- **scope** (*str*) – The scope of runners to show, one of: `specific`, `shared`, `active`, `paused`, `online`
- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the server failed to perform the request

Returns a list of runners matching the scope.

Return type `list(Runner)`

verify (***kwargs*)

Validates authentication credentials for a registered Runner.

Parameters

- **token** (*str*) – The runner's authentication token

- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabVerifyError` – If the server failed to verify the token

class `gitlab.v4.objects.SidekiqManager` (*gl*, *parent=None*)

Bases: `gitlab.base.RESTManager`

Manager for the Sidekiq methods.

This manager doesn't actually manage objects but provides helper function for the sidekiq metrics API.

compound_metrics (***kwargs*)

Return all available metrics and statistics.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the information couldn't be retrieved

Returns All available Sidekiq metrics and statistics

Return type dict

job_stats (***kwargs*)

Return statistics about the jobs performed.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the information couldn't be retrieved

Returns Statistics about the Sidekiq jobs performed

Return type dict

process_metrics (***kwargs*)

Return the registred sidekiq workers.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the information couldn't be retrieved

Returns Information about the register Sidekiq worker

Return type dict

queue_metrics (***kwargs*)

Return the registred queues information.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the information couldn't be retrieved

Returns Information about the Sidekiq queues

Return type dict

class gitlab.v4.objects.**Snippet** (*manager, attrs*)

Bases: *gitlab.mixins.UserAgentDetailMixin, gitlab.mixins.SaveMixin, gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

content (**kwargs)

Return the content of a snippet.

Parameters

- **streamed** (*bool*) – If True the data will be processed by chunks of *chunk_size* and each chunk is passed to *action* for treatment.
- **action** (*callable*) – Callable responsible of dealing with chunk of data
- **chunk_size** (*int*) – Size of each chunk
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the content could not be retrieved

Returns The snippet content

Return type str

class gitlab.v4.objects.**SnippetManager** (*gl, parent=None*)

Bases: *gitlab.mixins.CRUDMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- title
- file_name
- content

Optional attributes:

- lifetime
- visibility

Object update

Optional attributes for object update:

- title
- file_name
- content
- visibility

public (**kwargs)

List all the public snippets.

Parameters

- **all** (*bool*) – If True the returned object will be a list

- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises `GitlabListError` – If the list could not be retrieved

Returns A generator for the snippets list

Return type `RESTObjectList`

class `gitlab.v4.objects.TODO` (*manager, attrs*)

Bases: `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

mark_as_done (***kwargs*)

Mark the todo as done.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabTodoError` – If the server failed to perform the request

class `gitlab.v4.objects.TODOManager` (*gl, parent=None*)

Bases: `gitlab.mixins.ListMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

Object listing filters

- `action`
- `author_id`
- `project_id`
- `state`
- `type`

mark_all_as_done (***kwargs*)

Mark all the todos as done.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabTodoError` – If the server failed to perform the request

Returns The number of todos maked done

Return type `int`

class `gitlab.v4.objects.USER` (*manager, attrs*)

Bases: `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

block (***kwargs*)

Block the user.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabBlockError` – If the user could not be blocked

Returns Whether the user status has been changed

Return type bool

unblock (**kwargs)

Unblock the user.

Parameters **kwargs – Extra options to send to the server (e.g. sudo)

Raises

- GitlabAuthenticationError – If authentication is not correct
- GitlabUnblockError – If the user could not be unblocked

Returns Whether the user status has been changed

Return type bool

class gitlab.v4.objects.**UserActivities** (manager, attrs)

Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**UserActivitiesManager** (gl, parent=None)

Bases: *gitlab.mixins.ListMixin, gitlab.base.RESTManager*

class gitlab.v4.objects.**UserCustomAttribute** (manager, attrs)

Bases: *gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**UserCustomAttributeManager** (gl, parent=None)

Bases: *gitlab.mixins.RetrieveMixin, gitlab.mixins.SetMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager*

class gitlab.v4.objects.**UserEmail** (manager, attrs)

Bases: *gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**UserEmailManager** (gl, parent=None)

Bases: *gitlab.mixins.RetrieveMixin, gitlab.mixins.CreateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- email

class gitlab.v4.objects.**UserEvent** (manager, attrs)

Bases: *gitlab.v4.objects.Event*

class gitlab.v4.objects.**UserEventManager** (gl, parent=None)

Bases: *gitlab.v4.objects.EventManager*

Object listing filters

- action
- target_type
- before
- after
- sort

class gitlab.v4.objects.**UserGPGKey** (manager, attrs)

Bases: *gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject*

class gitlab.v4.objects.**UserGPGKeyManager** (gl, parent=None)

Bases: *gitlab.mixins.RetrieveMixin, gitlab.mixins.CreateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager*

Object Creation

Mandatory attributes:

- key

```
class gitlab.v4.objects.UserImpersonationToken (manager, attrs)
    Bases: gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.UserImpersonationTokenManager (gl, parent=None)
    Bases: gitlab.mixins.NoUpdateMixin, gitlab.base.RESTManager
```

Object listing filters

- state

Object Creation

Mandatory attributes:

- name
- scopes

Optional attributes:

- expires_at

```
class gitlab.v4.objects.UserKey (manager, attrs)
    Bases: gitlab.mixins.ObjectDeleteMixin, gitlab.base.RESTObject
```

```
class gitlab.v4.objects.UserKeyManager (gl, parent=None)
    Bases: gitlab.mixins.ListMixin, gitlab.mixins.CreateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager
```

Object Creation

Mandatory attributes:

- title
- key

```
class gitlab.v4.objects.UserManager (gl, parent=None)
    Bases: gitlab.mixins.CRUDMixin, gitlab.base.RESTManager
```

Object listing filters

- active
- blocked
- username
- extern_uid
- provider
- external
- search
- custom_attributes

Object Creation

Optional attributes:

- email

- username
- name
- password
- reset_password
- skype
- linkedin
- twitter
- projects_limit
- extern_uid
- provider
- bio
- admin
- can_create_group
- website_url
- skip_confirmation
- external
- organization
- location
- avatar

Object update

Mandatory attributes for object update:

- email
- username
- name

Optional attributes for object update:

- password
- skype
- linkedin
- twitter
- projects_limit
- extern_uid
- provider
- bio
- admin
- can_create_group
- website_url

- skip_confirmation
- external
- organization
- location
- avatar

class gitlab.v4.objects.**UserProject** (*manager, attrs*)

Bases: *gitlab.base.RESTObject*

class gitlab.v4.objects.**UserProjectManager** (*gl, parent=None*)

Bases: *gitlab.mixins.ListMixin, gitlab.mixins.CreateMixin, gitlab.base.RESTManager*

Object listing filters

- archived
- visibility
- order_by
- sort
- search
- simple
- owned
- membership
- starred
- statistics
- with_issues_enabled
- with_merge_requests_enabled

Object Creation

Mandatory attributes:

- name

Optional attributes:

- default_branch
- issues_enabled
- wall_enabled
- merge_requests_enabled
- wiki_enabled
- snippets_enabled
- public
- visibility
- description
- builds_enabled

- `public_builds`
- `import_url`
- `only_allow_merge_if_build_succeeds`

list (***kwargs*)

Retrieve a list of objects.

Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns The list of objects, or a generator if *as_list* is False

Return type list

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the server cannot perform the request

Module contents

7.2 Submodules

7.3 gitlab.base module

class `gitlab.base.RESTManager` (*gl, parent=None*)

Bases: `object`

Base class for CRUD operations on objects.

Derivated class must define `_path` and `_obj_cls`.

`_path`: Base URL path on which requests will be sent (e.g. `‘/projects’`) `_obj_cls`: The class of objects that will be created

parent_attrs

path

class `gitlab.base.RESTObject` (*manager, attrs*)

Bases: `object`

Represents an object built from server data.

It holds the attributes know from the server, and the updated attributes in another. This allows smart updates, if the object allows it.

You can redefine `_id_attr` in child classes to specify which attribute must be used as uniq ID. `None` means that the object can be updated without ID in the url.

attributes**get_id()**

Returns the id of the resource.

class gitlab.base.RESTObjectList (*manager, obj_cls, _list*)

Bases: object

Generator object representing a list of RESTObject's.

This generator uses the Gitlab pagination system to fetch new data when required.

Note: you should not instantiate such objects, they are returned by calls to RESTManager.list()

Parameters

- **manager** – Manager to attach to the created objects
- **obj_cls** – Type of objects to create from the json data
- **_list** – A GitlabList object

current_page

The current page number.

next()**next_page**

The next page number.

If None, the current page is the last.

per_page

The number of items per page.

prev_page

The next page number.

If None, the current page is the last.

total

The total number of items.

total_pages

The total number of pages.

7.4 gitlab.cli module

gitlab.cli.cls_to_what (*cls*)

gitlab.cli.die (*msg, e=None*)

gitlab.cli.main ()

gitlab.cli.register_custom_action (*cls_names, mandatory=(), optional=()*)

gitlab.cli.what_to_cls (*what*)

7.5 gitlab.config module

exception gitlab.config.ConfigError

Bases: exceptions.Exception

exception gitlab.config.GitlabConfigMissingError

Bases: *gitlab.config.ConfigError*

class gitlab.config.GitlabConfigParser (*gitlab_id=None, config_files=None*)

Bases: object

exception gitlab.config.GitlabDataError

Bases: *gitlab.config.ConfigError*

exception gitlab.config.GitlabIDError

Bases: *gitlab.config.ConfigError*

7.6 gitlab.const module

7.7 gitlab.exceptions module

exception gitlab.exceptions.GitlabAttachFileError (*error_message="",* *re-*
sponse_code=None, *re-*
sponse_body=None)

Bases: *gitlab.exceptions.GitlabOperationError*

exception gitlab.exceptions.GitlabAuthenticationError (*error_message="",* *re-*
sponse_code=None, *re-*
sponse_body=None)

Bases: *gitlab.exceptions.GitlabError*

exception gitlab.exceptions.GitlabBlockError (*error_message="", response_code=None,*
response_body=None)

Bases: *gitlab.exceptions.GitlabOperationError*

exception gitlab.exceptions.GitlabBuildCancelError (*error_message="",* *re-*
sponse_code=None, *re-*
sponse_body=None)

Bases: *gitlab.exceptions.GitlabCancelError*

exception gitlab.exceptions.GitlabBuildEraseError (*error_message="",* *re-*
sponse_code=None, *re-*
sponse_body=None)

Bases: *gitlab.exceptions.GitlabRetryError*

exception gitlab.exceptions.GitlabBuildPlayError (*error_message="",* *re-*
sponse_code=None, *re-*
sponse_body=None)

Bases: *gitlab.exceptions.GitlabRetryError*

exception gitlab.exceptions.GitlabBuildRetryError (*error_message="",* *re-*
sponse_code=None, *re-*
sponse_body=None)

Bases: *gitlab.exceptions.GitlabRetryError*

exception gitlab.exceptions.GitlabCancelError (*error_message="", response_code=None,*
response_body=None)

Bases: *gitlab.exceptions.GitlabOperationError*

exception gitlab.exceptions.GitlabCherryPickError (*error_message="",* *re-*
sponse_code=None, *re-*
sponse_body=None)

Bases: *gitlab.exceptions.GitlabOperationError*

```

exception gitlab.exceptions.GitlabConnectionError (error_message="",           re-
                                                    sponse_code=None,           re-
                                                    sponse_body=None)

    Bases: gitlab.exceptions.GitlabError

exception gitlab.exceptions.GitlabCreateError (error_message="", response_code=None,
                                                    response_body=None)

    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabDeleteError (error_message="", response_code=None,
                                                    response_body=None)

    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabError (error_message="",   response_code=None,   re-
                                                    sponse_body=None)

    Bases: exceptions.Exception

exception gitlab.exceptions.GitlabGetError (error_message="", response_code=None, re-
                                                    sponse_body=None)

    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabHousekeepingError (error_message="",           re-
                                                    sponse_code=None,           re-
                                                    sponse_body=None)

    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabHttpError (error_message="", response_code=None, re-
                                                    sponse_body=None)

    Bases: gitlab.exceptions.GitlabError

exception gitlab.exceptions.GitlabJobCancelError (error_message="",           re-
                                                    sponse_code=None,           re-
                                                    sponse_body=None)

    Bases: gitlab.exceptions.GitlabCancelError

exception gitlab.exceptions.GitlabJobEraseError (error_message="",           re-
                                                    sponse_code=None,           re-
                                                    sponse_body=None)

    Bases: gitlab.exceptions.GitlabRetryError

exception gitlab.exceptions.GitlabJobPlayError (error_message="",           re-
                                                    sponse_code=None,           re-
                                                    sponse_body=None)

    Bases: gitlab.exceptions.GitlabRetryError

exception gitlab.exceptions.GitlabJobRetryError (error_message="",           re-
                                                    sponse_code=None,           re-
                                                    sponse_body=None)

    Bases: gitlab.exceptions.GitlabRetryError

exception gitlab.exceptions.GitlabLicenseError (error_message="",           re-
                                                    sponse_code=None,           re-
                                                    sponse_body=None)

    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabListError (error_message="", response_code=None, re-
                                                    sponse_body=None)

    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabMRApprovalError (error_message="",           re-
                                                    sponse_code=None,           re-
                                                    sponse_body=None)

    Bases: gitlab.exceptions.GitlabOperationError

```

exception `gitlab.exceptions.GitlabMRClosedError` (`error_message=""`, `response_code=None`, `response_body=None`) re-re
Bases: `gitlab.exceptions.GitlabOperationError`

exception `gitlab.exceptions.GitlabMRForbiddenError` (`error_message=""`, `response_code=None`, `response_body=None`) re-re
Bases: `gitlab.exceptions.GitlabOperationError`

exception `gitlab.exceptions.GitlabMROnBuildSuccessError` (`error_message=""`, `response_code=None`, `response_body=None`) re-
Bases: `gitlab.exceptions.GitlabOperationError`

exception `gitlab.exceptions.GitlabMRRebaseError` (`error_message=""`, `response_code=None`, `response_body=None`) re-re
Bases: `gitlab.exceptions.GitlabOperationError`

exception `gitlab.exceptions.GitlabMarkdownError` (`error_message=""`, `response_code=None`, `response_body=None`) re-re
Bases: `gitlab.exceptions.GitlabOperationError`

exception `gitlab.exceptions.GitlabOperationError` (`error_message=""`, `response_code=None`, `response_body=None`) re-re
Bases: `gitlab.exceptions.GitlabError`

exception `gitlab.exceptions.GitlabOwnershipError` (`error_message=""`, `response_code=None`, `response_body=None`) re-re
Bases: `gitlab.exceptions.GitlabOperationError`

exception `gitlab.exceptions.GitlabParsingError` (`error_message=""`, `response_code=None`, `response_body=None`) re-re
Bases: `gitlab.exceptions.GitlabError`

exception `gitlab.exceptions.GitlabPipelineCancelError` (`error_message=""`, `response_code=None`, `response_body=None`) re-re
Bases: `gitlab.exceptions.GitlabCancelError`

exception `gitlab.exceptions.GitlabPipelineRetryError` (`error_message=""`, `response_code=None`, `response_body=None`) re-re
Bases: `gitlab.exceptions.GitlabRetryError`

exception `gitlab.exceptions.GitlabProjectDeployKeyError` (`error_message=""`, `response_code=None`, `response_body=None`) re-
Bases: `gitlab.exceptions.GitlabOperationError`

exception `gitlab.exceptions.GitlabProtectError` (`error_message=""`, `response_code=None`, `response_body=None`) re-re
Bases: `gitlab.exceptions.GitlabOperationError`

exception `gitlab.exceptions.GitlabRenderError` (`error_message=""`, `response_code=None`, `response_body=None`)
Bases: `gitlab.exceptions.GitlabOperationError`

```

exception gitlab.exceptions.GitlabRepairError (error_message="", response_code=None,
                                                response_body=None)
    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabRetryError (error_message="", response_code=None,
                                                response_body=None)
    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabSearchError (error_message="", response_code=None,
                                                response_body=None)
    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabSetError (error_message="", response_code=None, re-
                                                sponse_body=None)
    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabStopError (error_message="", response_code=None, re-
                                                sponse_body=None)
    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabSubscribeError (error_message="",                re-
                                                sponse_code=None,                re-
                                                sponse_body=None)
    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabTimeTrackingError (error_message="",                re-
                                                sponse_code=None,                re-
                                                sponse_body=None)
    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabTodoError (error_message="", response_code=None, re-
                                                sponse_body=None)
    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabTransferProjectError (error_message="",                re-
                                                sponse_code=None,                re-
                                                sponse_body=None)
    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabUnblockError (error_message="",                re-
                                                sponse_code=None,                re-
                                                sponse_body=None)
    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabUnsubscribeError (error_message="",                re-
                                                sponse_code=None,                re-
                                                sponse_body=None)
    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabUpdateError (error_message="", response_code=None,
                                                response_body=None)
    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabUploadError (error_message="", response_code=None,
                                                response_body=None)
    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.GitlabVerifyError (error_message="", response_code=None,
                                                response_body=None)
    Bases: gitlab.exceptions.GitlabOperationError

exception gitlab.exceptions.RedirectError (error_message="", response_code=None, re-
                                                sponse_body=None)
    Bases: gitlab.exceptions.GitlabError

```

`gitlab.exceptions.on_http_error` (*error*)
Manage GitlabHttpError exceptions.

This decorator function can be used to catch GitlabHttpError exceptions raise specialized exceptions instead.

Parameters `error` (*Exception*) – The exception type to raise – must inherit from GitlabError

7.8 gitlab.mixins module

class `gitlab.mixins.AccessRequestMixin`

Bases: `object`

approve (***kwargs*)

Approve an access request.

Parameters

- **access_level** (*int*) – The access level for the user
- ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabUpdateError` – If the server fails to perform the request

class `gitlab.mixins.BadgeRenderMixin`

Bases: `object`

render (***kwargs*)

Preview `link_url` and `image_url` after interpolation.

Parameters

- **link_url** (*str*) – URL of the badge link
- **image_url** (*str*) – URL of the badge image
- ****kwargs** – Extra options to send to the server (e.g. `sudo`)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabRenderError` – If the rendering failed

Returns The rendering properties

Return type `dict`

class `gitlab.mixins.CRUDMixin`

Bases: `gitlab.mixins.GetMixin`, `gitlab.mixins.ListMixin`, `gitlab.mixins.CreateMixin`, `gitlab.mixins.UpdateMixin`, `gitlab.mixins.DeleteMixin`

class `gitlab.mixins.CreateMixin`

Bases: `object`

create (***kwargs*)

Create a new object.

Parameters

- **data** (*dict*) – parameters to send to the server to create the resource

- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns

a new instance of the managed object class built with the data sent by the server

Return type *RESTObject*

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabCreateError` – If the server cannot perform the request

get_create_attrs()

Return the required and optional arguments.

Returns

2 items: list of required arguments and list of optional arguments for creation (in that order)

Return type tuple

```
class gitlab.mixins.DeleteMixin
```

Bases: object

```
delete (**kwargs)
```

Delete an object on the server.

Parameters

- **id** – ID of the object to delete
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabDeleteError` – If the server cannot perform the request

```
class gitlab.mixins.GetMixin
```

Bases: object

```
get (**kwargs)
```

Retrieve a single object.

Parameters

- **id** (*int or str*) – ID of the object to retrieve
- **lazy** (*bool*) – If True, don't request the server, but create a shallow object giving access to the managers. This is useful if you want to avoid useless calls to the API.
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns The generated RESTObject.

Return type object

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server cannot perform the request

```
class gitlab.mixins.GetWithoutIdMixin
```

Bases: object

get (**kwargs)

Retrieve a single object.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns The generated RESTObject

Return type object

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server cannot perform the request

class `gitlab.mixins.ListMixin`

Bases: object

list (**kwargs)

Retrieve a list of objects.

Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns The list of objects, or a generator if *as_list* is False

Return type list

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the server cannot perform the request

class `gitlab.mixins.NoUpdateMixin`

Bases: `gitlab.mixins.GetMixin`, `gitlab.mixins.ListMixin`, `gitlab.mixins.CreateMixin`, `gitlab.mixins.DeleteMixin`

class `gitlab.mixins.ObjectDeleteMixin`

Bases: object

Mixin for RESTObject's that can be deleted.

delete (**kwargs)

Delete the object from the server.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabDeleteError` – If the server cannot perform the request

class `gitlab.mixins.ParticipantsMixin`

Bases: object

participants (***kwargs*)

List the participants.

Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **per_page** (*int*) – Number of items to retrieve per request
- **page** (*int*) – ID of the page to return (starts with page 1)
- **as_list** (*bool*) – If set to False and no pagination option is defined, return a generator instead of a list
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabListError` – If the list could not be retrieved

Returns The list of participants

Return type *RESTObjectList*

class `gitlab.mixins.RefreshMixin`

Bases: `object`

refresh (***kwargs*)

Refresh a single object from server.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns None (updates the object)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server cannot perform the request

class `gitlab.mixins.RetrieveMixin`

Bases: `gitlab.mixins.ListMixin`, `gitlab.mixins.GetMixin`

class `gitlab.mixins.SaveMixin`

Bases: `object`

Mixin for `RESTObject`'s that can be updated.

save (***kwargs*)

Save the changes made to the object to the server.

The object is updated to match what the server returns.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raise: `GitlabAuthenticationError`: If authentication is not correct `GitlabUpdateError`: If the server cannot perform the request

class `gitlab.mixins.SetMixin`

Bases: `object`

set (***kwargs*)

Create or update the object.

Parameters

- **key** (*str*) – The key of the object to create/update
- **value** (*str*) – The value to set for the object
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabSetError` – If an error occurred

Returns The created/updated attribute

Return type `obj`

class `gitlab.mixins.SubscribableMixin`

Bases: `object`

subscribe (****kwargs**)

Subscribe to the object notifications.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabSubscribeError` – If the subscription cannot be done

unsubscribe (****kwargs**)

Unsubscribe from the object notifications.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabUnsubscribeError` – If the unsubscription cannot be done

class `gitlab.mixins.TimeTrackingMixin`

Bases: `object`

add_spent_time (****kwargs**)

Add time spent working on the object.

Parameters

- **duration** (*str*) – Duration in human format (e.g. 3h30)
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabTimeTrackingError` – If the time tracking update cannot be done

reset_spent_time (****kwargs**)

Resets the time spent working on the object.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabTimeTrackingError` – If the time tracking update cannot be done

reset_time_estimate (**kwargs)

Resets estimated time for the object to 0 seconds.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabTimeTrackingError` – If the time tracking update cannot be done

time_estimate (**kwargs)

Set an estimated time of work for the object.

Parameters

- **duration** (*str*) – Duration in human format (e.g. 3h30)
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabTimeTrackingError` – If the time tracking update cannot be done

time_stats (**kwargs)

Get time stats for the object.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabTimeTrackingError` – If the time tracking update cannot be done

class `gitlab.mixins.TODOMixin`

Bases: `object`

todo (**kwargs)

Create a todo associated to the object.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabTodoError` – If the todo cannot be set

class `gitlab.mixins.UpdateMixin`

Bases: `object`

get_update_attrs ()

Return the required and optional arguments.

Returns

2 items: list of required arguments and list of optional arguments for update (in that order)

Return type tuple

update (**kwargs)

Update an object on the server.

Parameters

- **id** – ID of the object to update (can be None if not required)
- **new_data** – the update data for the object
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns The new object data (*not* a RESTObject)

Return type dict

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabUpdateError` – If the server cannot perform the request

class `gitlab.mixins.UserAgentDetailMixin`

Bases: `object`

user_agent_detail (***kwargs*)

Get the user agent detail.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabGetError` – If the server cannot perform the request

7.9 gitlab.utils module

`gitlab.utils.clean_str_id` (*id*)

`gitlab.utils.copy_dict` (*dest, src*)

`gitlab.utils.response_content` (*response, streamed, action, chunk_size*)

`gitlab.utils.sanitized_url` (*url*)

7.10 Module contents

Wrapper for the GitLab API.

class `gitlab.Gitlab` (*url, private_token=None, oauth_token=None, email=None, password=None, ssl_verify=True, http_username=None, http_password=None, timeout=None, api_version='4', session=None, per_page=None*)

Bases: `object`

Represents a GitLab server connection.

Parameters

- **url** (*str*) – The URL of the GitLab server.
- **private_token** (*str*) – The user private token
- **oauth_token** (*str*) – An oauth token
- **email** (*str*) – The user email or login.
- **password** (*str*) – The user password (associated with email).

- **ssl_verify** (*bool/str*) – Whether SSL certificates should be validated. If the value is a string, it is the path to a CA file used for certificate validation.
- **timeout** (*float*) – Timeout to use for requests to the GitLab server.
- **http_username** (*str*) – Username for HTTP authentication
- **http_password** (*str*) – Password for HTTP authentication
- **api_version** (*str*) – Gitlab API version to use (support for 4 only)

api_url

The computed API base URL.

api_version

The API version used (4 only).

auth()

Performs an authentication.

Uses either the private token, or the email/password pair.

The *user* attribute will hold a *gitlab.objects.CurrentUser* object on success.

email = None

The user email

enable_debug()**classmethod from_config** (*gitlab_id=None, config_files=None*)

Create a Gitlab connection from configuration files.

Parameters

- **gitlab_id** (*str*) – ID of the configuration section.
- **list[str]** (*config_files*) – List of paths to configuration files.

Returns A Gitlab connection.

Return type (*gitlab.Gitlab*)

Raises *gitlab.config.GitlabDataError* – If the configuration is not correct.

get_license (***kwargs*)

Retrieve information about the current license.

Parameters ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- *GitlabAuthenticationError* – If authentication is not correct
- *GitlabGetError* – If the server cannot perform the request

Returns The current license information

Return type dict

headers = None

Headers that will be used in request to GitLab

http_delete (*path, **kwargs*)

Make a PUT request to the Gitlab server.

Parameters

- **path** (*str*) – Path or full URL to query ('/projects' or 'http://whatever/v4/api/projects')

- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns The requests object.

Raises `GitlabHttpError` – When the return code is not 2xx

http_get (*path*, *query_data=None*, *streamed=False*, *raw=False*, ***kwargs*)

Make a GET request to the Gitlab server.

Parameters

- **path** (*str*) – Path or full URL to query (‘/projects’ or ‘<http://whatever/v4/api/projects>’)
- **query_data** (*dict*) – Data to send as query parameters
- **streamed** (*bool*) – Whether the data should be streamed
- **raw** (*bool*) – If True do not try to parse the output as json
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns A requests result object is streamed is True or the content type is not json. The parsed json data otherwise.

Raises

- `GitlabHttpError` – When the return code is not 2xx
- `GitlabParsingError` – If the json data could not be parsed

http_list (*path*, *query_data=None*, *as_list=None*, ***kwargs*)

Make a GET request to the Gitlab server for list-oriented queries.

Parameters

- **path** (*str*) – Path or full URL to query (‘/projects’ or ‘<http://whatever/v4/api/projects>’)
- **query_data** (*dict*) – Data to send as query parameters
- ****kwargs** – Extra options to send to the server (e.g. sudo, page, per_page)

Returns A list of the objects returned by the server. If *as_list* is False and no pagination-related arguments (*page*, *per_page*, *all*) are defined then a `GitlabList` object (generator) is returned instead. This object will make API calls when needed to fetch the next items from the server.

Return type list

Raises

- `GitlabHttpError` – When the return code is not 2xx
- `GitlabParsingError` – If the json data could not be parsed

http_post (*path*, *query_data=None*, *post_data=None*, *files=None*, ***kwargs*)

Make a POST request to the Gitlab server.

Parameters

- **path** (*str*) – Path or full URL to query (‘/projects’ or ‘<http://whatever/v4/api/projects>’)
- **query_data** (*dict*) – Data to send as query parameters
- **post_data** (*dict*) – Data to send in the body (will be converted to json)
- **files** (*dict*) – The files to send to the server
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns The parsed json returned by the server if json is return, else the raw content

Raises

- `GitlabHttpError` – When the return code is not 2xx
- `GitlabParsingError` – If the json data could not be parsed

http_put (*path*, *query_data=None*, *post_data=None*, *files=None*, ***kwargs*)

Make a PUT request to the Gitlab server.

Parameters

- **path** (*str*) – Path or full URL to query ('/projects' or 'http://whatever/v4/api/projects')
- **query_data** (*dict*) – Data to send as query parameters
- **post_data** (*dict*) – Data to send in the body (will be converted to json)
- **files** (*dict*) – The files to send to the server
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns The parsed json returned by the server.

Raises

- `GitlabHttpError` – When the return code is not 2xx
- `GitlabParsingError` – If the json data could not be parsed

http_request (*verb*, *path*, *query_data=None*, *post_data=None*, *streamed=False*, *files=None*, ***kwargs*)

Make an HTTP request to the Gitlab server.

Parameters

- **verb** (*str*) – The HTTP method to call ('get', 'post', 'put', 'delete')
- **path** (*str*) – Path or full URL to query ('/projects' or 'http://whatever/v4/api/projects')
- **query_data** (*dict*) – Data to send as query parameters
- **post_data** (*dict*) – Data to send in the body (will be converted to json)
- **streamed** (*bool*) – Whether the data should be streamed
- **files** (*dict*) – The files to send to the server
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Returns A requests result object.

Raises `GitlabHttpError` – When the return code is not 2xx

lint (***kwargs*)

Validate a gitlab CI configuration.

Parameters

- **content** (*txt*) – The .gitlab-ci.yml content
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabVerifyError` – If the validation could not be done

Returns

(**True**, []) if the file is valid, (**False**, **errors(list)**) otherwise

Return type tuple

markdown (**kwargs)

Render an arbitrary Markdown document.

Parameters

- **text** (*str*) – The markdown text to render
- **gfm** (*bool*) – Render text using GitLab Flavored Markdown. Default is False
- **project** (*str*) – Full path of a project used a context when *gfm* is True
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabMarkdownError` – If the server cannot perform the request

Returns The HTML rendering of the markdown text.

Return type str

password = None

The user password (associated with email)

search (**kwargs)

Search GitLab resources matching the provided string.

Parameters

- **scope** (*str*) – Scope of the search
- **search** (*str*) – Search string
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabSearchError` – If the server failed to perform the request

Returns A list of dicts describing the resources found.

Return type *GitlabList*

session = None

Create a session object for requests

set_license (**kwargs)

Add a new license.

Parameters

- **license** (*str*) – The license string
- ****kwargs** – Extra options to send to the server (e.g. sudo)

Raises

- `GitlabAuthenticationError` – If authentication is not correct
- `GitlabPostError` – If the server cannot perform the request

Returns The new license information

Return type dict

ssl_verify = None

Whether SSL certificates should be validated

timeout = None

Timeout to use for requests to gitlab server

url

The user-provided server URL.

version ()

Returns the version and revision of the gitlab server.

Note that self.version and self.revision will be set on the gitlab object.

Returns

The server version and server revision. ('unknown', 'unknown') if the server doesn't perform as expected.

Return type tuple (str, str)

class gitlab.**GitlabList** (*gl, url, query_data, get_next=True, **kwargs*)

Bases: object

Generator representing a list of remote objects.

The object handles the links returned by a query to the API, and will call the API again when needed.

current_page

The current page number.

next ()

next_page

The next page number.

If None, the current page is the last.

per_page

The number of items per page.

prev_page

The next page number.

If None, the current page is the last.

total

The total number of items.

total_pages

The total number of pages.

This page describes important changes between python-gitlab releases.

8.1 Changes from 1.8 to 1.9

- `ProjectMemberManager.all()` and `GroupMemberManager.all()` now return a list of `ProjectMember` and `GroupMember` objects respectively, instead of a list of dicts.

8.2 Changes from 1.7 to 1.8

- You can now use the `query_parameters` argument in method calls to define arguments to send to the GitLab server. This allows to avoid conflicts between python-gitlab and GitLab server variables, and allows to use the python reserved keywords as GitLab arguments.

The following examples make the same GitLab request with the 2 syntaxes:

```
projects = gl.projects.list(owned=True, starred=True)
projects = gl.projects.list(query_parameters={'owned': True, 'starred': True})
```

The following example only works with the new parameter:

```
activities = gl.user_activities.list(
    query_parameters={'from': '2019-01-01'},
    all=True)
```

- Additionally the `all` parameter is not sent to the GitLab anymore.

8.3 Changes from 1.5 to 1.6

- When python-gitlab detects HTTP redirections from http to https it will raise a `RedirectionError` instead of a cryptic error.

Make sure to use an `https://` protocol in your GitLab URL parameter if the server requires it.

8.4 Changes from 1.4 to 1.5

- APIv3 support has been removed. Use the 1.4 release/branch if you need v3 support.
- GitLab EE features are now supported: Geo nodes, issue links, LDAP groups, project/group boards, project mirror pulling, project push rules, EE license configuration, epics.
- The `GetFromListMixin` class has been removed. The `get()` method is not available anymore for the following managers:
 - `UserKeyManager`
 - `DeployKeyManager`
 - `GroupAccessRequestManager`
 - `GroupIssueManager`
 - `GroupProjectManager`
 - `GroupSubgroupManager`
 - `IssueManager`
 - `ProjectCommitStatusManager`
 - `ProjectEnvironmentManager`
 - `ProjectLabelManager`
 - `ProjectPipelineJobManager`
 - `ProjectAccessRequestManager`
 - `TodoManager`
- `ProjectPipelineJob` do not heritate from `ProjectJob` anymore and thus can only be listed.

8.5 Changes from 1.3 to 1.4

- 1.4 is the last release supporting the v3 API, and the related code will be removed in the 1.5 version.

If you are using a Gitlab server version that does not support the v4 API you can:

- upgrade the server (recommended)
- make sure to use version 1.4 of python-gitlab (`pip install python-gitlab==1.4`)

See also the [Switching to GitLab API v4 documentation](#).

- python-gitlab now handles the server rate limiting feature. It will pause for the required time when reaching the limit ([documentation](#))

- The `GetFromListMixin.get()` method is deprecated and will be removed in the next python-gitlab version. The goal of this mixin/method is to provide a way to get an object by looping through a list for GitLab objects that don't support the GET method. The method is broken and conflicts with the GET method now supported by some GitLab objects.

You can implement your own method with something like:

```
def get_from_list(self, id):
    for obj in self.list(as_list=False):
        if obj.get_id() == id:
            return obj
```

- The `GroupMemberManager`, `NamespaceManager` and `ProjectBoardManager` managers now use the GET API from GitLab instead of the `GetFromListMixin.get()` method.

8.6 Changes from 1.2 to 1.3

- `gitlab.Gitlab` objects can be used as context managers in a `with` block.

8.7 Changes from 1.1 to 1.2

- python-gitlab now respects the `*_proxy`, `REQUESTS_CA_BUNDLE` and `CURL_CA_BUNDLE` environment variables (#352)
- The following deprecated methods and objects have been removed:
 - `gitlab.v3.object.Key` and `KeyManager` objects: use `DeployKey` and `DeployKeyManager` instead
 - `gitlab.v3.objects.Project.archive_` and `unarchive_` methods
 - `gitlab.Gitlab.credentials_auth`, `token_auth`, `set_url`, `set_token` and `set_credentials` methods. Once a Gitlab object has been created its URL and authentication information cannot be updated: create a new Gitlab object if you need to use new information
- The `todo()` method raises a `GitlabTodoError` exception on error

8.8 Changes from 1.0.2 to 1.1

- The `ProjectUser` class doesn't inherit from `User` anymore, and the `GroupProject` class doesn't inherit from `Project` anymore. The Gitlab API doesn't provide the same set of features for these objects, so python-gitlab objects shouldn't try to workaround that.

You can create `User` or `Project` objects from `ProjectUser` and `GroupProject` objects using the `id` attribute:

```
for gr_project in group.projects.list():
    # lazy object creation avoids a Gitlab API request
    project = gl.projects.get(gr_project.id, lazy=True)
    project.default_branch = 'develop'
    project.save()
```

8.9 Changes from 0.21 to 1.0.0

1.0.0 brings a stable python-gitlab API for the v4 Gitlab API. v3 is still used by default.

v4 is mostly compatible with the v3, but some important changes have been introduced. Make sure to read [Switching to GitLab API v4](#).

The development focus will be v4 from now on. v3 has been deprecated by GitLab and will disappear from python-gitlab at some point.

8.10 Changes from 0.20 to 0.21

- Initial support for the v4 API (experimental)

The support for v4 is stable enough to be tested, but some features might be broken. Please report issues to <https://github.com/python-gitlab/python-gitlab/issues/>

Be aware that the python-gitlab API for v4 objects might change in the next releases.

Warning: Consider defining explicitly which API version you want to use in the configuration files or in your `gitlab.Gitlab` instances. The default will change from v3 to v4 soon.

- Several methods have been deprecated in the `gitlab.Gitlab` class:
 - `credentials_auth()` is deprecated and will be removed. Call `auth()`.
 - `token_auth()` is deprecated and will be removed. Call `auth()`.
 - `set_url()` is deprecated, create a new `Gitlab` instance if you need an updated URL.
 - `set_token()` is deprecated, use the `private_token` argument of the `Gitlab` constructor.
 - `set_credentials()` is deprecated, use the `email` and `password` arguments of the `Gitlab` constructor.
- The service listing method (`ProjectServiceManager.list()`) now returns a python list instead of a JSON string.

8.11 Changes from 0.19 to 0.20

- The `projects` attribute of `Group` objects is not a list of `Project` objects anymore. It is a `Manager` object giving access to `GroupProject` objects. To get the list of projects use:

```
group.projects.list()
```

Documentation: http://python-gitlab.readthedocs.io/en/stable/gl_objects/groups.html#examples

Related issue: <https://github.com/python-gitlab/python-gitlab/issues/209>

- The `Key` objects are deprecated in favor of the new `DeployKey` objects. They are exactly the same but the name makes more sense.

Documentation: http://python-gitlab.readthedocs.io/en/stable/gl_objects/deploy_keys.html

Related issue: <https://github.com/python-gitlab/python-gitlab/issues/212>

9.1 Version 1.9.0 - 2019-06-19

9.1.1 Features

- implement artifacts deletion
- add endpoint to get the variables of a pipeline
- delete ProjectPipeline
- implement `__eq__` and `__hash__` methods
- Allow runpy invocation of CLI tool (python -m gitlab)
- add project releases api
- merged new release & registry apis

9.1.2 Bug Fixes

- convert # to %23 in URLs
- pep8 errors
- use python2 compatible syntax for super
- Make MemberManager.all() return a list of objects
- %d replaced by %s
- Re-enable command specific help messages
- dont ask for id attr if this is *Manager originating custom action
- fix -/_ replacament for *Manager custom actions
- fix repository_id marshaling in cli

- register cli action for delete_in_bulk

9.2 Version 1.8.0 - 2019-02-22

- docs(setup): use proper readme on PyPI
- docs(readme): provide commit message guidelines
- fix(api): make reset_time_estimate() work again
- fix: handle empty 'Retry-After' header from GitLab
- fix: remove decode() on error_message string
- chore: release tags to PyPI automatically
- fix(api): avoid parameter conflicts with python and gitlab
- fix(api): Don't try to parse raw downloads
- feat: Added approve & unapprove method for Mergerequests
- fix all kwarg behaviour

9.3 Version 1.7.0 - 2018-12-09

- [docs] Fix the owned/starred usage documentation
- [docs] Add a warning about http to https redirects
- Fix the https redirection test
- [docs] Add a note about GroupProject limited API
- Add missing comma in ProjectIssueManager _create_attrs
- More flexible docker image
- Add project protected tags management
- [cli] Print help and usage without config file
- Rename MASTER_ACCESS to MAINTAINER_ACCESS
- [docs] Add docs build information
- Use docker image with current sources
- [docs] Add PyYAML requirement notice
- Add Gitter badge to README
- [docs] Add an example of pipeline schedule vars listing
- [cli] Exit on config parse error, instead of crashing
- Add support for resource label events
- [docs] Fix the milestone filetring doc (iid -> iids)
- [docs] Fix typo in custom attributes example
- Improve error message handling in exceptions
- Add support for members all() method

- Add access control options to protected branch creation

9.4 Version 1.6.0 - 2018-08-25

- [docs] Don't use hardcoded values for ids
- [docs] Improve the snippets examples
- [cli] Output: handle bytes in API responses
- [cli] Fix the case where we have nothing to print
- Project import: fix the override_params parameter
- Support group and global MR listing
- Implement MR.pipelines()
- MR: add the squash attribute for create/update
- Added support for listing forks of a project
- [docs] Add/update notes about read-only objects
- Raise an exception on https redirects for PUT/POST
- [docs] Add a FAQ
- [cli] Fix the project-export download

9.5 Version 1.5.1 - 2018-06-23

- Fix the ProjectPipelineJob base class (regression)

9.6 Version 1.5.0 - 2018-06-22

- Drop API v3 support
- Drop GetFromListMixin
- Update the sphinx extension for v4 objects
- Add support for user avatar upload
- Add support for project import/export
- Add support for the search API
- Add a global per_page config option
- Add support for the discussions API
- Add support for merged branches deletion
- Add support for Project badges
- Implement user_agent_detail for snippets
- Implement commit.refs()
- Add commit.merge_requests() support

- Deployment: add list filters
- Deploy key: add missing attributes
- Add support for environment stop()
- Add feature flags deletion support
- Update some group attributes
- Issues: add missing attributes and methods
- Fix the participants() decorator
- Add support for group boards
- Implement the markdown rendering API
- Update MR attributes
- Add pipeline listing filters
- Add missing project attributes
- Implement runner jobs listing
- Runners can be created (registered)
- Implement runner token validation
- Update the settings attributes
- Add support for the gitlab CI lint API
- Add support for group badges
- Fix the IssueManager path to avoid redirections
- time_stats(): use an existing attribute if available
- Make ProjectCommitStatus.create work with CLI
- Tests: default to python 3
- ProjectPipelineJob was defined twice
- Silence logs/warnings in unittests
- Add support for MR approval configuration (EE)
- Change post_data default value to None
- Add geo nodes API support (EE)
- Add support for issue links (EE)
- Add support for LDAP groups (EE)
- Add support for board creation/deletion (EE)
- Add support for Project.pull_mirror (EE)
- Add project push rules configuration (EE)
- Add support for the EE license API
- Add support for the LDAP groups API (EE)
- Add support for epics API (EE)
- Fix the non-verbose output of ProjectCommitComment

9.7 Version 1.4.0 - 2018-05-19

- Require requests \geq 2.4.2
- ProjectKeys can be updated
- Add support for unsharing projects (v3/v4)
- [cli] fix listing for json and yaml output
- Fix typos in documentation
- Introduce RefreshMixin
- [docs] Fix the time tracking examples
- [docs] Commits: add an example of binary file creation
- [cli] Allow to read args from files
- Add support for recursive tree listing
- [cli] Restore the `-help` option behavior
- Add basic unit tests for v4 CLI
- [cli] Fix listing of strings
- Support downloading a single artifact file
- Update docs copyright years
- Implement attribute types to handle special cases
- [docs] fix GitLab reference for notes
- Expose additional properties for Gitlab objects
- Fix the impersonation token deletion example
- feat: obey the rate limit
- Fix URL encoding on branch methods
- [docs] add a code example for listing commits of a MR
- [docs] update `service.available()` example for API v4
- [tests] fix functional tests for python3
- api-usage: bit more detail for listing with *all*
- More efficient `.get()` for group members
- Add docs for the *files* arg in `http_*`
- Deprecate GetFromListMixin

9.8 Version 1.3.0 - 2018-02-18

- Add support for pipeline schedules and schedule variables
- Clarify information about supported python version
- Add manager for jobs within a pipeline
- Fix wrong tag example

- Update the groups documentation
- Add support for MR participants API
- Add support for getting list of user projects
- Add Gitlab and User events support
- Make trigger_pipeline return the pipeline
- Config: support api_version in the global section
- Gitlab can be used as context manager
- Default to API v4
- Add a simplified example for streamed artifacts
- Add documentation about labels update

9.9 Version 1.2.0 - 2018-01-01

- Add mattermost service support
- Add users custom attributes support
- [doc] Fix project.triggers.create example with v4 API
- OAuth token support
- Remove deprecated objects/methods
- Rework authentication args handling
- Add support for oauth and anonymous auth in config/CLI
- Add support for impersonation tokens API
- Add support for user activities
- Update user docs with gitlab URLs
- [docs] Bad arguments in projects file documentation
- Add support for user_agent_detail (issues)
- Add a SetMixin
- Add support for project housekeeping
- Expected HTTP response for subscribe is 201
- Update pagination docs for ProjectCommit
- Add doc to get issue from iid
- Make todo() raise GitlabTodoError on error
- Add support for award emojis
- Update project services docs for v4
- Avoid sending empty update data to issue.save
- [docstrings] Explicitly document pagination arguments
- [docs] Add a note about password auth being removed from GitLab

- Submanagers: allow having undefined parameters
- ProjectFile.create(): don't modify the input data
- Update testing tools for /session removal
- Update groups tests
- Allow per_page to be used with generators
- Add groups listing attributes
- Add support for subgroups listing
- Add supported python versions in setup.py
- Add support for pagesdomains
- Add support for features flags
- Add support for project and group custom variables
- Add support for user/group/project filter by custom attribute
- Respect content of REQUESTS_CA_BUNDLE and *_proxy envvars

9.10 Version 1.1.0 - 2017-11-03

- Fix trigger variables in v4 API
- Make the delete() method handle / in ids
- [docs] update the file upload samples
- Tags release description: support / in tag names
- [docs] improve the labels usage documentation
- Add support for listing project users
- ProjectFileManager.create: handle / in file paths
- Change ProjectUser and GroupProject base class
- [docs] document *get_create_attrs* in the API tutorial
- Document the Gitlab session parameter
- ProjectFileManager: custom update() method
- Project: add support for printing_merge_request_link_enabled attr
- Update the ssl_verify docstring
- Add support for group milestones
- Add support for GPG keys
- Add support for wiki pages
- Update the repository_blob documentation
- Fix the CLI for objects without ID (API v4)
- Add a contributed Dockerfile
- Pagination generators: expose more information

- Module's base objects serialization
- [doc] Add sample code for client-side certificates

9.11 Version 1.0.2 - 2017-09-29

- [docs] remove example usage of submanagers
- Properly handle the labels attribute in ProjectMergeRequest
- ProjectFile: handle / in path for delete() and save()

9.12 Version 1.0.1 - 2017-09-21

- Tags can be retrieved by ID
- Add the server response in GitlabError exceptions
- Add support for project file upload
- Minor typo fix in "Switching to v4" documentation
- Fix password authentication for v4
- Fix the labels attrs on MR and issues
- Exceptions: use a proper error message
- Fix http_get method in get artifacts and job trace
- CommitStatus: *sha* is parent attribute
- Fix a couple listing calls to allow proper pagination
- Add missing doc file

9.13 Version 1.0.0 - 2017-09-08

- Support for API v4. See <http://python-gitlab.readthedocs.io/en/master/switching-to-v4.html>
- Support SSL verification via internal CA bundle
- Docs: Add link to gitlab docs on obtaining a token
- Added dependency injection support for Session
- Fixed repository_compare examples
- Fix changelog and release notes inclusion in sdist
- Missing expires_at in GroupMembers update
- Add lower-level methods for Gitlab()

9.14 Version 0.21.2 - 2017-06-11

- Install doc: use sudo for system commands
- [v4] Make MR work properly
- Remove extra_attrs argument from _raw_list
- [v4] Make project issues work properly
- Fix urlencode() usage (python 2/3) (#268)
- Fixed spelling mistake (#269)
- Add new event types to ProjectHook

9.15 Version 0.21.1 - 2017-05-25

- Fix the manager name for jobs in the Project class
- Fix the docs

9.16 Version 0.21 - 2017-05-24

- Add time_stats to ProjectMergeRequest
- Update User options for creation and update (#246)
- Add milestone.merge_requests() API
- Fix docs typo (s/correspding/corresponding/)
- Support milestone start date (#251)
- Add support for priority attribute in labels (#256)
- Add support for nested groups (#257)
- Make GroupProjectManager a subclass of ProjectManager (#255)
- Available services: return a list instead of JSON (#258)
- MR: add support for time tracking features (#248)
- Fixed repository_tree and repository_blob path encoding (#265)
- Add 'search' attribute to projects.list()
- Initial gitlab API v4 support
- Reorganise the code to handle v3 and v4 objects
- Allow 202 as delete return code
- Deprecate parameter related methods in gitlab.Gitlab

9.17 Version 0.20 - 2017-03-25

- Add time tracking support (#222)
- Improve changelog (#229, #230)
- Make sure that manager objects are never overwritten (#209)
- Include chanlog and release notes in docs
- Add DeployKey{,Manager} classes (#212)
- Add support for merge request notes deletion (#227)
- Properly handle extra args when listing with all=True (#233)
- Implement pipeline creation API (#237)
- Fix spent_time methods
- Add 'delete source branch' option when creating MR (#241)
- Provide API wrapper for cherry picking commits (#236)
- Stop listing if recursion limit is hit (#234)

9.18 Version 0.19 - 2017-02-21

- Update project.archive() docs
- Support the scope attribute in runners.list()
- Add support for project runners
- Add support for commit creation
- Fix install doc
- Add builds-email and pipelines-email services
- Deploy keys: rework enable/disable
- Document the dynamic aspect of objects
- Add pipeline_events to ProjectHook attrs
- Add due_date attribute to ProjectIssue
- Handle settings.domain_whitelist, partly
- {Project,Group}Member: support expires_at attribute

9.19 Version 0.18 - 2016-12-27

- Fix JIRA service editing for GitLab 8.14+
- Add jira_issue_transition_id to the JIRA service optional fields
- Added support for Snippets (new API in Gitlab 8.15)
- [docs] update pagination section
- [docs] artifacts example: open file in wb mode

- [CLI] ignore empty arguments
- [CLI] Fix wrong use of arguments
- [docs] Add doc for snippets
- Fix duplicated data in API docs
- Update known attributes for projects
- sudo: always use strings

9.20 Version 0.17 - 2016-12-02

- README: add badges for pypi and RTD
- Fix ProjectBuild.play (raised error on success)
- Pass kwargs to the object factory
- Add .tox to ignore to respect default tox settings
- Convert response list to single data source for iid requests
- Add support for boards API
- Add support for Gitlab.version()
- Add support for broadcast messages API
- Add support for the notification settings API
- Don't overwrite attributes returned by the server
- Fix bug when retrieving changes for merge request
- Feature: enable / disable the deploy key in a project
- Docs: add a note for python 3.5 for file content update
- ProjectHook: support the token attribute
- Rework the API documentation
- Fix docstring for http_{username,password}
- Build managers on demand on GitlabObject's
- API docs: add managers doc in GitlabObject's
- Sphinx ext: factorize the build methods
- Implement __repr__ for gitlab objects
- Add a 'report a bug' link on doc
- Remove deprecated methods
- Implement merge requests diff support
- Make the manager objects creation more dynamic
- Add support for templates API
- Add attr 'created_at' to ProjectIssueNote
- Add attr 'updated_at' to ProjectIssue

- CLI: add support for project all --all
- Add support for triggering a new build
- Rework requests arguments (support latest requests release)
- Fix *should_remove_source_branch*

9.21 Version 0.16 - 2016-10-16

- Add the ability to fork to a specific namespace
- JIRA service - add api_url to optional attributes
- Fix bug: Missing coma concatenates array values
- docs: branch protection notes
- Create a project in a group
- Add only_allow_merge_if_build_succeeds option to project objects
- Add support for --all in CLI
- Fix examples for file modification
- Use the plural merge_requests URL everywhere
- Rework travis and tox setup
- Workaround gitlab setup failure in tests
- Add ProjectBuild.erase()
- Implement ProjectBuild.play()

9.22 Version 0.15.1 - 2016-10-16

- docs: improve the pagination section
- Fix and test pagination
- 'path' is an existing gitlab attr, don't use it as method argument

9.23 Version 0.15 - 2016-08-28

- Add a basic HTTP debug method
- Run more tests in travis
- Fix fork creation documentation
- Add more API examples in docs
- Update the ApplicationSettings attributes
- Implement the todo API
- Add sidekiq metrics support
- Move the constants at the gitlab root level

- Remove methods marked as deprecated 7 months ago
- Refactor the Gitlab class
- Remove `_get_list_or_object()` and its tests
- Fix `canGet` attribute (typo)
- Remove unused `ProjectTagReleaseManager` class
- Add support for project services API
- Add support for project pipelines
- Add support for access requests
- Add support for project deployments

9.24 Version 0.14 - 2016-08-07

- Remove `'next_url'` from kwargs before passing it to the cls constructor.
- List projects under group
- Add support for subscribe and unsubscribe in issues
- Project issue: doc and CLI for (un)subscribe
- Added support for HTTP basic authentication
- Add support for build artifacts and trace
- `-title` is a required argument for `ProjectMilestone`
- Commit status: add optional context url
- Commit status: optional get attrs
- Add support for commit comments
- Issues: add optional listing parameters
- Issues: add missing optional listing parameters
- Project issue: proper update attributes
- Add support for project-issue move
- Update `ProjectLabel` attributes
- Milestone: optional listing attrs
- Add support for namespaces
- Add support for label (un)subscribe
- MR: add (un)subscribe support
- Add `note_events` to project hooks attributes
- Add code examples for a bunch of resources
- Implement user emails support
- Project: add `VISIBILITY_*` constants
- Fix the `Project.archive` call

- Implement archive/unarchive for a projet
- Update ProjectSnippet attributes
- Fix ProjectMember update
- Implement sharing project with a group
- Implement CLI for project archive/unarchive/share
- Implement runners global API
- Gitlab: add managers for build-related resources
- Implement ProjectBuild.keep_artifacts
- Allow to stream the downloads when appropriate
- Groups can be updated
- Replace Snippet.Content() with a new content() method
- CLI: refactor _die()
- Improve commit statuses and comments
- Add support from listing group issues
- Added a new project attribute to enable the container registry.
- Add a contributing section in README
- Add support for global deploy key listing
- Add support for project environments
- MR: get list of changes and commits
- Fix the listing of some resources
- MR: fix updates
- Handle empty messages from server in exceptions
- MR (un)subscribe: don't fail if state doesn't change
- MR merge(): update the object

9.25 Version 0.13 - 2016-05-16

- Add support for MergeRequest validation
- MR: add support for cancel_merge_when_build_succeeds
- MR: add support for closes_issues
- Add “external” parameter for users
- Add deletion support for issues and MR
- Add missing group creation parameters
- Add a Session instance for all HTTP requests
- Enable updates on ProjectIssueNotes
- Add support for Project raw_blob

- Implement project compare
- Implement project contributors
- Drop the next_url attribute when listing
- Remove unnecessary canUpdate property from ProjectIssuesNote
- Add new optional attributes for projects
- Enable deprecation warnings for gitlab only
- Rework merge requests update
- Rework the Gitlab.delete method
- ProjectFile: file_path is required for deletion
- Rename some methods to better match the API URLs
- Deprecate the file_* methods in favor of the files manager
- Implement star/unstar for projects
- Implement list/get licenses
- Manage optional parameters for list() and get()

9.26 Version 0.12.2 - 2016-03-19

- Add new *ProjectHook* attributes
- Add support for user block/unblock
- Fix GitlabObject creation in _custom_list
- Add support for more CLI subcommands
- Add some unit tests for CLI
- Add a coverage tox env
- Define GitlabObject.as_dict() to dump object as a dict
- Define GitlabObject.__eq__() and __ne__() equivalence methods
- Define UserManager.search() to search for users
- Define UserManager.get_by_username() to get a user by username
- Implement “user search” CLI
- Improve the doc for UserManager
- CLI: implement user get-by-username
- Re-implement _custom_list in the Gitlab class
- Fix the ‘invalid syntax’ error on Python 3.2
- Gitlab.update(): use the proper attributes if defined

9.27 Version 0.12.1 - 2016-02-03

- Fix a broken upload to pypi

9.28 Version 0.12 - 2016-02-03

- Improve documentation
- Improve unit tests
- Improve test scripts
- Skip BaseManager attributes when encoding to JSON
- Fix the json() method for python 3
- Add Travis CI support
- Add a decode method for ProjectFile
- Make connection exceptions more explicit
- Fix ProjectLabel get and delete
- Implement ProjectMilestone.issues()
- ProjectTag supports deletion
- Implement setting release info on a tag
- Implement project triggers support
- Implement project variables support
- Add support for application settings
- Fix the 'password' requirement for User creation
- Add sudo support
- Fix project update
- Fix Project.tree()
- Add support for project builds

9.29 Version 0.11.1 - 2016-01-17

- Fix discovery of parents object attrs for managers
- Support setting commit status
- Support deletion without getting the object first
- Improve the documentation

9.30 Version 0.11 - 2016-01-09

- functional_tests.sh: support python 2 and 3
- Add a get method for GitlabObject
- CLI: Add the -g short option for -gitlab
- Provide a create method for GitlabObject's
- Rename the _created attribute _from_api
- More unit tests
- CLI: fix error when arguments are missing (python 3)
- Remove deprecated methods
- Implement managers to get access to resources
- Documentation improvements
- Add fork project support
- Deprecate the "old" Gitlab methods
- Add support for groups search

9.31 Version 0.10 - 2015-12-29

- Implement pagination for list() (#63)
- Fix url when fetching a single MergeRequest
- Add support to update MergeRequestNotes
- API: Provide a Gitlab.from_config method
- setup.py: require requests>=1 (#69)
- Fix deletion of object not using 'id' as ID (#68)
- Fix GET/POST for project files
- Make 'confirm' an optional attribute for user creation
- Python 3 compatibility fixes
- Add support for group members update (#73)

9.32 Version 0.9.2 - 2015-07-11

- CLI: fix the update and delete subcommands (#62)

9.33 Version 0.9.1 - 2015-05-15

- Fix the setup.py script

9.34 Version 0.9 - 2015-05-15

- Implement argparse library for parsing argument on CLI
- Provide unit tests and (a few) functional tests
- Provide PEP8 tests
- Use tox to run the tests
- CLI: provide a `--config-file` option
- Turn the gitlab module into a proper package
- Allow projects to be updated
- Use more pythonic names for some methods
- **Deprecate some Gitlab object methods:**
 - `raw*` methods should never have been exposed; replace them with `_raw_*` methods
 - `setCredentials` and `setToken` are replaced with `set_credentials` and `set_token`
- Sphinx: don't hardcode the version in `conf.py`

9.35 Version 0.8 - 2014-10-26

- Better python 2.6 and python 3 support
- Timeout support in HTTP requests
- `Gitlab.get()` raised `GitlabListError` instead of `GitlabGetError`
- Support api-objects which don't have id in api response
- Add `ProjectLabel` and `ProjectFile` classes
- Moved url attributes to separate list
- Added list for delete attributes

9.36 Version 0.7 - 2014-08-21

- Fix license classifier in `setup.py`
- Fix encoding error when printing to redirected output
- Fix encoding error when updating with redirected output
- Add support for `UserKey` listing and deletion
- Add support for branches creation and deletion
- Support `state_event` in `ProjectMilestone` (#30)
- Support `namespace/name` for project id (#28)
- Fix handling of boolean values (#22)

9.37 Version 0.6 - 2014-01-16

- IDs can be unicode (#15)
- ProjectMember: constructor should not create a User object
- Add support for extra parameters when listing all projects (#12)
- Projects listing: explicitly define arguments for pagination

9.38 Version 0.5 - 2013-12-26

- Add SSH key for user
- Fix comments
- Add support for project events
- Support creation of projects for users
- Project: add methods for create/update/delete files
- Support projects listing: search, all, owned
- System hooks can't be updated
- Project.archive(): download tarball of the project
- Define new optional attributes for user creation
- Provide constants for access permissions in groups

9.39 Version 0.4 - 2013-09-26

- Fix strings encoding (Closes #6)
- Allow to get a project commit (GitLab 6.1)
- ProjectMergeRequest: fix Note() method
- Gitlab 6.1 methods: diff, blob (commit), tree, blob (project)
- Add support for Gitlab 6.1 group members

9.40 Version 0.3 - 2013-08-27

- Use PRIVATE-TOKEN header for passing the auth token
- provide an AUTHORS file
- cli: support ssl_verify config option
- Add ssl_verify option to Gitlab object. Defaults to True
- Correct url for merge requests API.

9.41 Version 0.2 - 2013-08-08

- provide a pip requirements.txt
- drop some debug statements

9.42 Version 0.1 - 2013-07-08

- Initial release

CHAPTER 10

Indices and tables

- `genindex`
- `modindex`
- `search`

g

gitlab, 180
gitlab.base, 168
gitlab.cli, 169
gitlab.config, 169
gitlab.const, 170
gitlab.exceptions, 170
gitlab.mixins, 174
gitlab.utils, 180
gitlab.v4, 168
gitlab.v4.objects, 93

A

AccessRequestMixin (class in *gitlab.mixins*), 174
 add_ldap_group_link() (*gitlab.v4.objects.Group* method), 100
 add_spent_time() (*gitlab.mixins.TimeTrackingMixin* method), 178
 all() (*gitlab.v4.objects.GroupMemberManager* method), 106
 all() (*gitlab.v4.objects.ProjectMemberManager* method), 141
 all() (*gitlab.v4.objects.RunnerManager* method), 160
 api_url (*gitlab.Gitlab* attribute), 181
 api_version (*gitlab.Gitlab* attribute), 181
 ApplicationSettings (class in *gitlab.v4.objects*), 93
 ApplicationSettingsManager (class in *gitlab.v4.objects*), 93
 approve() (*gitlab.mixins.AccessRequestMixin* method), 174
 approve() (*gitlab.v4.objects.ProjectMergeRequest* method), 142
 archive() (*gitlab.v4.objects.Project* method), 113
 artifact() (*gitlab.v4.objects.Project* method), 113
 artifact() (*gitlab.v4.objects.ProjectJob* method), 135
 artifacts() (*gitlab.v4.objects.ProjectJob* method), 135
 attributes (*gitlab.base.RESTObject* attribute), 168
 auth() (*gitlab.Gitlab* method), 181
 available() (*gitlab.v4.objects.ProjectServiceManager* method), 154

B

BadgeRenderMixin (class in *gitlab.mixins*), 174
 block() (*gitlab.v4.objects.User* method), 163
 BroadcastMessage (class in *gitlab.v4.objects*), 96
 BroadcastMessageManager (class in *gitlab.v4.objects*), 96

C

cancel() (*gitlab.v4.objects.ProjectJob* method), 136
 cancel() (*gitlab.v4.objects.ProjectPipeline* method), 150
 cancel_merge_when_pipeline_succeeds() (*gitlab.v4.objects.ProjectMergeRequest* method), 142
 changes() (*gitlab.v4.objects.ProjectMergeRequest* method), 142
 cherry_pick() (*gitlab.v4.objects.ProjectCommit* method), 121
 clean_str_id() (in module *gitlab.utils*), 180
 closed_by() (*gitlab.v4.objects.ProjectIssue* method), 131
 closes_issues() (*gitlab.v4.objects.ProjectMergeRequest* method), 142
 cls_to_what() (in module *gitlab.cli*), 169
 commits() (*gitlab.v4.objects.ProjectMergeRequest* method), 143
 compound_metrics() (*gitlab.v4.objects.SidekiqManager* method), 161
 ConfigError, 169
 content() (*gitlab.v4.objects.ProjectSnippet* method), 155
 content() (*gitlab.v4.objects.Snippet* method), 162
 copy_dict() (in module *gitlab.utils*), 180
 create() (*gitlab.mixins.CreateMixin* method), 174
 create() (*gitlab.v4.objects.GroupEpicIssueManager* method), 102
 create() (*gitlab.v4.objects.ProjectCommitStatusManager* method), 124
 create() (*gitlab.v4.objects.ProjectFileManager* method), 127
 create() (*gitlab.v4.objects.ProjectIssueLinkManager* method), 133
 create() (*gitlab.v4.objects.ProjectPipelineManager* method), 150

create_fork_relation() (*gitlab.v4.objects.Project* method), 114
 CreateMixin (*class in gitlab.mixins*), 174
 CRUDMixin (*class in gitlab.mixins*), 174
 current_failures() (*gitlab.v4.objects.GeoNodeManager* method), 99
 current_page (*gitlab.base.RESTObjectList* attribute), 169
 current_page (*gitlab.GitlabList* attribute), 185
 CurrentUser (*class in gitlab.v4.objects*), 97
 CurrentUserEmail (*class in gitlab.v4.objects*), 97
 CurrentUserEmailManager (*class in gitlab.v4.objects*), 97
 CurrentUserGPGKey (*class in gitlab.v4.objects*), 97
 CurrentUserGPGKeyManager (*class in gitlab.v4.objects*), 97
 CurrentUserKey (*class in gitlab.v4.objects*), 97
 CurrentUserKeyManager (*class in gitlab.v4.objects*), 97
 CurrentUserManager (*class in gitlab.v4.objects*), 97

D

decode() (*gitlab.v4.objects.ProjectFile* method), 126
 delete() (*gitlab.mixins.DeleteMixin* method), 175
 delete() (*gitlab.mixins.ObjectDeleteMixin* method), 176
 delete() (*gitlab.v4.objects.GroupLabelManager* method), 105
 delete() (*gitlab.v4.objects.ProjectFile* method), 126
 delete() (*gitlab.v4.objects.ProjectFileManager* method), 128
 delete() (*gitlab.v4.objects.ProjectLabelManager* method), 138
 delete_artifacts() (*gitlab.v4.objects.ProjectJob* method), 136
 delete_fork_relation() (*gitlab.v4.objects.Project* method), 114
 delete_in_bulk() (*gitlab.v4.objects.ProjectRegistryTagManager* method), 153
 delete_ldap_group_link() (*gitlab.v4.objects.Group* method), 100
 delete_merged_branches() (*gitlab.v4.objects.Project* method), 114
 DeleteMixin (*class in gitlab.mixins*), 175
 DeployKey (*class in gitlab.v4.objects*), 97
 DeployKeyManager (*class in gitlab.v4.objects*), 97
 die() (*in module gitlab.cli*), 169
 diff() (*gitlab.v4.objects.ProjectCommit* method), 122
 Dockerfile (*class in gitlab.v4.objects*), 97
 DockerfileManager (*class in gitlab.v4.objects*), 97
 download() (*gitlab.v4.objects.ProjectExport* method), 126

E

email (*gitlab.Gitlab* attribute), 181
 enable() (*gitlab.v4.objects.ProjectKeyManager* method), 137
 enable_debug() (*gitlab.Gitlab* method), 181
 erase() (*gitlab.v4.objects.ProjectJob* method), 136
 Event (*class in gitlab.v4.objects*), 97
 EventManager (*class in gitlab.v4.objects*), 98

F

Feature (*class in gitlab.v4.objects*), 98
 FeatureManager (*class in gitlab.v4.objects*), 98
 from_config() (*gitlab.Gitlab* class method), 181

G

GeoNode (*class in gitlab.v4.objects*), 98
 GeoNodeManager (*class in gitlab.v4.objects*), 99
 get() (*gitlab.mixins.GetMixin* method), 175
 get() (*gitlab.mixins.GetWithoutIdMixin* method), 175
 get() (*gitlab.v4.objects.ProjectFileManager* method), 128
 get() (*gitlab.v4.objects.ProjectServiceManager* method), 154
 get_create_attrs() (*gitlab.mixins.CreateMixin* method), 175
 get_id() (*gitlab.base.RESTObject* method), 169
 get_license() (*gitlab.Gitlab* method), 181
 get_update_attrs() (*gitlab.mixins.UpdateMixin* method), 179
 GetMixin (*class in gitlab.mixins*), 175
 GetWithoutIdMixin (*class in gitlab.mixins*), 175
 Gitignore (*class in gitlab.v4.objects*), 99
 GitignoreManager (*class in gitlab.v4.objects*), 99
 Gitlab (*class in gitlab*), 180
 gitlab (*module*), 180
 gitlab.base (*module*), 168
 gitlab.cli (*module*), 169
 gitlab.config (*module*), 169
 gitlab.const (*module*), 170
 gitlab.exceptions (*module*), 170
 gitlab.mixins (*module*), 174
 gitlab.utils (*module*), 180
 gitlab.v4 (*module*), 168
 gitlab.v4.objects (*module*), 93
 GitlabAttachFileError, 170
 GitlabAuthenticationError, 170
 GitlabBlockError, 170
 GitlabBuildCancelError, 170
 GitlabBuildEraseError, 170
 GitlabBuildPlayError, 170
 GitlabBuildRetryError, 170
 GitlabCancelError, 170
 GitlabCherryPickError, 170

- Gitlabciyaml (*class in gitlab.v4.objects*), 99
 - GitlabciyamlManager (*class in gitlab.v4.objects*), 99
 - GitlabConfigMissingError, 170
 - GitlabConfigParser (*class in gitlab.config*), 170
 - GitlabConnectionError, 170
 - GitlabCreateError, 171
 - GitlabDataError, 170
 - GitlabDeleteError, 171
 - GitlabError, 171
 - GitlabGetError, 171
 - GitlabHousekeepingError, 171
 - GitlabHttpError, 171
 - GitlabIDError, 170
 - GitlabJobCancelError, 171
 - GitlabJobEraseError, 171
 - GitlabJobPlayError, 171
 - GitlabJobRetryError, 171
 - GitlabLicenseError, 171
 - GitlabList (*class in gitlab*), 185
 - GitlabListError, 171
 - GitlabMarkdownError, 172
 - GitlabMRApprovalError, 171
 - GitlabMRClosedError, 171
 - GitlabMRForbiddenError, 172
 - GitlabMROnBuildSuccessError, 172
 - GitlabMRRebaseError, 172
 - GitlabOperationError, 172
 - GitlabOwnershipError, 172
 - GitlabParsingError, 172
 - GitlabPipelineCancelError, 172
 - GitlabPipelineRetryError, 172
 - GitlabProjectDeployKeyError, 172
 - GitlabProtectError, 172
 - GitlabRenderError, 172
 - GitlabRepairError, 173
 - GitlabRetryError, 173
 - GitlabSearchError, 173
 - GitlabSetError, 173
 - GitlabStopError, 173
 - GitlabSubscribeError, 173
 - GitlabTimeTrackingError, 173
 - GitlabTodoError, 173
 - GitlabTransferProjectError, 173
 - GitlabUnblockError, 173
 - GitlabUnsubscribeError, 173
 - GitlabUpdateError, 173
 - GitlabUploadError, 173
 - GitlabVerifyError, 173
 - Group (*class in gitlab.v4.objects*), 99
 - GroupAccessRequest (*class in gitlab.v4.objects*), 101
 - GroupAccessRequestManager (*class in gitlab.v4.objects*), 101
 - GroupBadge (*class in gitlab.v4.objects*), 101
 - GroupBadgeManager (*class in gitlab.v4.objects*), 101
 - GroupBoard (*class in gitlab.v4.objects*), 101
 - GroupBoardList (*class in gitlab.v4.objects*), 101
 - GroupBoardListManager (*class in gitlab.v4.objects*), 101
 - GroupBoardManager (*class in gitlab.v4.objects*), 102
 - GroupCustomAttribute (*class in gitlab.v4.objects*), 102
 - GroupCustomAttributeManager (*class in gitlab.v4.objects*), 102
 - GroupEpic (*class in gitlab.v4.objects*), 102
 - GroupEpicIssue (*class in gitlab.v4.objects*), 102
 - GroupEpicIssueManager (*class in gitlab.v4.objects*), 102
 - GroupEpicManager (*class in gitlab.v4.objects*), 103
 - GroupEpicResourceLabelEvent (*class in gitlab.v4.objects*), 103
 - GroupEpicResourceLabelEventManager (*class in gitlab.v4.objects*), 103
 - GroupIssue (*class in gitlab.v4.objects*), 103
 - GroupIssueManager (*class in gitlab.v4.objects*), 103
 - GroupLabel (*class in gitlab.v4.objects*), 104
 - GroupLabelManager (*class in gitlab.v4.objects*), 104
 - GroupManager (*class in gitlab.v4.objects*), 105
 - GroupMember (*class in gitlab.v4.objects*), 106
 - GroupMemberManager (*class in gitlab.v4.objects*), 106
 - GroupMergeRequest (*class in gitlab.v4.objects*), 107
 - GroupMergeRequestManager (*class in gitlab.v4.objects*), 107
 - GroupMilestone (*class in gitlab.v4.objects*), 107
 - GroupMilestoneManager (*class in gitlab.v4.objects*), 108
 - GroupNotificationSettings (*class in gitlab.v4.objects*), 108
 - GroupNotificationSettingsManager (*class in gitlab.v4.objects*), 109
 - GroupProject (*class in gitlab.v4.objects*), 109
 - GroupProjectManager (*class in gitlab.v4.objects*), 109
 - GroupSubgroup (*class in gitlab.v4.objects*), 109
 - GroupSubgroupManager (*class in gitlab.v4.objects*), 109
 - GroupVariable (*class in gitlab.v4.objects*), 110
 - GroupVariableManager (*class in gitlab.v4.objects*), 110
- ## H
- headers (*gitlab.Gitlab attribute*), 181
 - Hook (*class in gitlab.v4.objects*), 110
 - HookManager (*class in gitlab.v4.objects*), 110
 - housekeeping() (*gitlab.v4.objects.Project method*), 114
 - http_delete() (*gitlab.Gitlab method*), 181

http_get() (*gitlab.Gitlab method*), 182
 http_list() (*gitlab.Gitlab method*), 182
 http_post() (*gitlab.Gitlab method*), 182
 http_put() (*gitlab.Gitlab method*), 183
 http_request() (*gitlab.Gitlab method*), 183

I

import_project() (*gitlab.v4.objects.ProjectManager method*), 140

Issue (*class in gitlab.v4.objects*), 110
 IssueManager (*class in gitlab.v4.objects*), 110
 issues() (*gitlab.v4.objects.GroupMilestone method*), 107
 issues() (*gitlab.v4.objects.ProjectMilestone method*), 147

J

job_stats() (*gitlab.v4.objects.SidekiqManager method*), 161

K

keep_artifacts() (*gitlab.v4.objects.ProjectJob method*), 136

L

languages() (*gitlab.v4.objects.Project method*), 114
 ldap_sync() (*gitlab.v4.objects.Group method*), 100
 LDAPGroup (*class in gitlab.v4.objects*), 111
 LDAPGroupManager (*class in gitlab.v4.objects*), 111
 License (*class in gitlab.v4.objects*), 111
 LicenseManager (*class in gitlab.v4.objects*), 111
 lint() (*gitlab.Gitlab method*), 183
 list() (*gitlab.mixins.ListMixin method*), 176
 list() (*gitlab.v4.objects.LDAPGroupManager method*), 111
 list() (*gitlab.v4.objects.ProjectForkManager method*), 129
 list() (*gitlab.v4.objects.UserProjectManager method*), 168
 ListMixin (*class in gitlab.mixins*), 176

M

main() (*in module gitlab.cli*), 169
 mark_all_as_done() (*gitlab.v4.objects.TODOManager method*), 163
 mark_as_done() (*gitlab.v4.objects.TODO method*), 163
 markdown() (*gitlab.Gitlab method*), 184
 merge() (*gitlab.v4.objects.ProjectMergeRequest method*), 143
 merge_requests() (*gitlab.v4.objects.GroupMilestone method*), 108

merge_requests() (*gitlab.v4.objects.ProjectCommit method*), 122
 merge_requests() (*gitlab.v4.objects.ProjectMilestone method*), 148

MergeRequest (*class in gitlab.v4.objects*), 112
 MergeRequestManager (*class in gitlab.v4.objects*), 112
 mirror_pull() (*gitlab.v4.objects.Project method*), 114
 move() (*gitlab.v4.objects.ProjectIssue method*), 131

N

Namespace (*class in gitlab.v4.objects*), 112
 NamespaceManager (*class in gitlab.v4.objects*), 112
 next() (*gitlab.base.RESTObjectList method*), 169
 next() (*gitlab.GitlabList method*), 185
 next_page (*gitlab.base.RESTObjectList attribute*), 169
 next_page (*gitlab.GitlabList attribute*), 185
 NotificationSettings (*class in gitlab.v4.objects*), 112
 NotificationSettingsManager (*class in gitlab.v4.objects*), 112
 NoUpdateMixin (*class in gitlab.mixins*), 176

O

ObjectDeleteMixin (*class in gitlab.mixins*), 176
 on_http_error() (*in module gitlab.exceptions*), 173

P

PagesDomain (*class in gitlab.v4.objects*), 113
 PagesDomainManager (*class in gitlab.v4.objects*), 113
 parent_attrs (*gitlab.base.RESTManager attribute*), 168
 participants() (*gitlab.mixins.ParticipantsMixin method*), 176
 ParticipantsMixin (*class in gitlab.mixins*), 176
 password (*gitlab.Gitlab attribute*), 184
 path (*gitlab.base.RESTManager attribute*), 168
 per_page (*gitlab.base.RESTObjectList attribute*), 169
 per_page (*gitlab.GitlabList attribute*), 185
 pipelines() (*gitlab.v4.objects.ProjectMergeRequest method*), 143
 play() (*gitlab.v4.objects.ProjectJob method*), 136
 prev_page (*gitlab.base.RESTObjectList attribute*), 169
 prev_page (*gitlab.GitlabList attribute*), 185
 process_metrics() (*gitlab.v4.objects.SidekiqManager method*), 161
 Project (*class in gitlab.v4.objects*), 113
 ProjectAccessRequest (*class in gitlab.v4.objects*), 119

- ProjectAccessRequestManager (class in *gitlab.v4.objects*), 119
- ProjectApproval (class in *gitlab.v4.objects*), 119
- ProjectApprovalManager (class in *gitlab.v4.objects*), 120
- ProjectBadge (class in *gitlab.v4.objects*), 120
- ProjectBadgeManager (class in *gitlab.v4.objects*), 120
- ProjectBoard (class in *gitlab.v4.objects*), 120
- ProjectBoardList (class in *gitlab.v4.objects*), 120
- ProjectBoardListManager (class in *gitlab.v4.objects*), 120
- ProjectBoardManager (class in *gitlab.v4.objects*), 121
- ProjectBranch (class in *gitlab.v4.objects*), 121
- ProjectBranchManager (class in *gitlab.v4.objects*), 121
- ProjectCommit (class in *gitlab.v4.objects*), 121
- ProjectCommitComment (class in *gitlab.v4.objects*), 122
- ProjectCommitCommentManager (class in *gitlab.v4.objects*), 122
- ProjectCommitDiscussion (class in *gitlab.v4.objects*), 123
- ProjectCommitDiscussionManager (class in *gitlab.v4.objects*), 123
- ProjectCommitDiscussionNote (class in *gitlab.v4.objects*), 123
- ProjectCommitDiscussionNoteManager (class in *gitlab.v4.objects*), 123
- ProjectCommitManager (class in *gitlab.v4.objects*), 123
- ProjectCommitStatus (class in *gitlab.v4.objects*), 124
- ProjectCommitStatusManager (class in *gitlab.v4.objects*), 124
- ProjectCustomAttribute (class in *gitlab.v4.objects*), 124
- ProjectCustomAttributeManager (class in *gitlab.v4.objects*), 124
- ProjectDeployment (class in *gitlab.v4.objects*), 124
- ProjectDeploymentManager (class in *gitlab.v4.objects*), 125
- ProjectEnvironment (class in *gitlab.v4.objects*), 125
- ProjectEnvironmentManager (class in *gitlab.v4.objects*), 125
- ProjectEvent (class in *gitlab.v4.objects*), 125
- ProjectEventManager (class in *gitlab.v4.objects*), 125
- ProjectExport (class in *gitlab.v4.objects*), 125
- ProjectExportManager (class in *gitlab.v4.objects*), 126
- ProjectFile (class in *gitlab.v4.objects*), 126
- ProjectFileManager (class in *gitlab.v4.objects*), 127
- ProjectFork (class in *gitlab.v4.objects*), 129
- ProjectForkManager (class in *gitlab.v4.objects*), 129
- ProjectHook (class in *gitlab.v4.objects*), 130
- ProjectHookManager (class in *gitlab.v4.objects*), 130
- ProjectImport (class in *gitlab.v4.objects*), 131
- ProjectImportManager (class in *gitlab.v4.objects*), 131
- ProjectIssue (class in *gitlab.v4.objects*), 131
- ProjectIssueAwardEmoji (class in *gitlab.v4.objects*), 132
- ProjectIssueAwardEmojiManager (class in *gitlab.v4.objects*), 132
- ProjectIssueDiscussion (class in *gitlab.v4.objects*), 132
- ProjectIssueDiscussionManager (class in *gitlab.v4.objects*), 132
- ProjectIssueDiscussionNote (class in *gitlab.v4.objects*), 132
- ProjectIssueDiscussionNoteManager (class in *gitlab.v4.objects*), 132
- ProjectIssueLink (class in *gitlab.v4.objects*), 132
- ProjectIssueLinkManager (class in *gitlab.v4.objects*), 132
- ProjectIssueManager (class in *gitlab.v4.objects*), 133
- ProjectIssueNote (class in *gitlab.v4.objects*), 134
- ProjectIssueNoteAwardEmoji (class in *gitlab.v4.objects*), 134
- ProjectIssueNoteAwardEmojiManager (class in *gitlab.v4.objects*), 134
- ProjectIssueNoteManager (class in *gitlab.v4.objects*), 134
- ProjectIssueResourceLabelEvent (class in *gitlab.v4.objects*), 135
- ProjectIssueResourceLabelEventManager (class in *gitlab.v4.objects*), 135
- ProjectJob (class in *gitlab.v4.objects*), 135
- ProjectJobManager (class in *gitlab.v4.objects*), 137
- ProjectKey (class in *gitlab.v4.objects*), 137
- ProjectKeyManager (class in *gitlab.v4.objects*), 137
- ProjectLabel (class in *gitlab.v4.objects*), 137
- ProjectLabelManager (class in *gitlab.v4.objects*), 138
- ProjectManager (class in *gitlab.v4.objects*), 139
- ProjectMember (class in *gitlab.v4.objects*), 141
- ProjectMemberManager (class in *gitlab.v4.objects*), 141
- ProjectMergeRequest (class in *gitlab.v4.objects*), 142
- ProjectMergeRequestApproval (class in *git-*

- lab.v4.objects*), 144
- ProjectMergeRequestApprovalManager (class in *gitlab.v4.objects*), 144
- ProjectMergeRequestAwardEmoji (class in *gitlab.v4.objects*), 144
- ProjectMergeRequestAwardEmojiManager (class in *gitlab.v4.objects*), 144
- ProjectMergeRequestDiff (class in *gitlab.v4.objects*), 144
- ProjectMergeRequestDiffManager (class in *gitlab.v4.objects*), 144
- ProjectMergeRequestDiscussion (class in *gitlab.v4.objects*), 145
- ProjectMergeRequestDiscussionManager (class in *gitlab.v4.objects*), 145
- ProjectMergeRequestDiscussionNote (class in *gitlab.v4.objects*), 145
- ProjectMergeRequestDiscussionNoteManager (class in *gitlab.v4.objects*), 145
- ProjectMergeRequestManager (class in *gitlab.v4.objects*), 145
- ProjectMergeRequestNote (class in *gitlab.v4.objects*), 147
- ProjectMergeRequestNoteAwardEmoji (class in *gitlab.v4.objects*), 147
- ProjectMergeRequestNoteAwardEmojiManager (class in *gitlab.v4.objects*), 147
- ProjectMergeRequestNoteManager (class in *gitlab.v4.objects*), 147
- ProjectMergeRequestResourceLabelEvent (class in *gitlab.v4.objects*), 147
- ProjectMergeRequestResourceLabelEventManager (class in *gitlab.v4.objects*), 147
- ProjectMilestone (class in *gitlab.v4.objects*), 147
- ProjectMilestoneManager (class in *gitlab.v4.objects*), 148
- ProjectNote (class in *gitlab.v4.objects*), 148
- ProjectNoteManager (class in *gitlab.v4.objects*), 149
- ProjectNotificationSettings (class in *gitlab.v4.objects*), 149
- ProjectNotificationSettingsManager (class in *gitlab.v4.objects*), 149
- ProjectPagesDomain (class in *gitlab.v4.objects*), 149
- ProjectPagesDomainManager (class in *gitlab.v4.objects*), 149
- ProjectPipeline (class in *gitlab.v4.objects*), 150
- ProjectPipelineJob (class in *gitlab.v4.objects*), 150
- ProjectPipelineJobManager (class in *gitlab.v4.objects*), 150
- ProjectPipelineManager (class in *gitlab.v4.objects*), 150
- ProjectPipelineSchedule (class in *gitlab.v4.objects*), 151
- ProjectPipelineScheduleManager (class in *gitlab.v4.objects*), 151
- ProjectPipelineScheduleVariable (class in *gitlab.v4.objects*), 151
- ProjectPipelineScheduleVariableManager (class in *gitlab.v4.objects*), 152
- ProjectPipelineVariable (class in *gitlab.v4.objects*), 152
- ProjectPipelineVariableManager (class in *gitlab.v4.objects*), 152
- ProjectProtectedBranch (class in *gitlab.v4.objects*), 152
- ProjectProtectedBranchManager (class in *gitlab.v4.objects*), 152
- ProjectProtectedTag (class in *gitlab.v4.objects*), 152
- ProjectProtectedTagManager (class in *gitlab.v4.objects*), 152
- ProjectPushRules (class in *gitlab.v4.objects*), 153
- ProjectPushRulesManager (class in *gitlab.v4.objects*), 153
- ProjectRegistryRepository (class in *gitlab.v4.objects*), 153
- ProjectRegistryRepositoryManager (class in *gitlab.v4.objects*), 153
- ProjectRegistryTag (class in *gitlab.v4.objects*), 153
- ProjectRegistryTagManager (class in *gitlab.v4.objects*), 153
- ProjectRelease (class in *gitlab.v4.objects*), 154
- ProjectReleaseManager (class in *gitlab.v4.objects*), 154
- ProjectRunner (class in *gitlab.v4.objects*), 154
- ProjectRunnerManager (class in *gitlab.v4.objects*), 154
- ProjectService (class in *gitlab.v4.objects*), 154
- ProjectServiceManager (class in *gitlab.v4.objects*), 154
- ProjectSnippet (class in *gitlab.v4.objects*), 155
- ProjectSnippetAwardEmoji (class in *gitlab.v4.objects*), 155
- ProjectSnippetAwardEmojiManager (class in *gitlab.v4.objects*), 156
- ProjectSnippetDiscussion (class in *gitlab.v4.objects*), 156
- ProjectSnippetDiscussionManager (class in *gitlab.v4.objects*), 156
- ProjectSnippetDiscussionNote (class in *gitlab.v4.objects*), 156
- ProjectSnippetDiscussionNoteManager (class in *gitlab.v4.objects*), 156
- ProjectSnippetManager (class in *git-*

- lab.v4.objects*), 156
- ProjectSnippetNote (class in *gitlab.v4.objects*), 157
- ProjectSnippetNoteAwardEmoji (class in *gitlab.v4.objects*), 157
- ProjectSnippetNoteAwardEmojiManager (class in *gitlab.v4.objects*), 157
- ProjectSnippetNoteManager (class in *gitlab.v4.objects*), 157
- ProjectTag (class in *gitlab.v4.objects*), 157
- ProjectTagManager (class in *gitlab.v4.objects*), 157
- ProjectTrigger (class in *gitlab.v4.objects*), 158
- ProjectTriggerManager (class in *gitlab.v4.objects*), 158
- ProjectUser (class in *gitlab.v4.objects*), 158
- ProjectUserManager (class in *gitlab.v4.objects*), 158
- ProjectVariable (class in *gitlab.v4.objects*), 158
- ProjectVariableManager (class in *gitlab.v4.objects*), 158
- ProjectWiki (class in *gitlab.v4.objects*), 159
- ProjectWikiManager (class in *gitlab.v4.objects*), 159
- protect() (*gitlab.v4.objects.ProjectBranch* method), 121
- public() (*gitlab.v4.objects.SnippetManager* method), 162
- ## Q
- queue_metrics() (*gitlab.v4.objects.SidekiqManager* method), 161
- ## R
- raw() (*gitlab.v4.objects.ProjectFileManager* method), 128
- rebase() (*gitlab.v4.objects.ProjectMergeRequest* method), 144
- RedirectError, 173
- refresh() (*gitlab.mixins.RefreshMixin* method), 177
- RefreshMixin (class in *gitlab.mixins*), 177
- refs() (*gitlab.v4.objects.ProjectCommit* method), 122
- register_custom_action() (in module *gitlab.cli*), 169
- related_merge_requests() (*gitlab.v4.objects.ProjectIssue* method), 131
- render() (*gitlab.mixins.BadgeRenderMixin* method), 174
- repair() (*gitlab.v4.objects.GeoNode* method), 98
- repository_archive() (*gitlab.v4.objects.Project* method), 115
- repository_blob() (*gitlab.v4.objects.Project* method), 115
- repository_compare() (*gitlab.v4.objects.Project* method), 115
- repository_contributors() (*gitlab.v4.objects.Project* method), 116
- repository_raw_blob() (*gitlab.v4.objects.Project* method), 116
- repository_tree() (*gitlab.v4.objects.Project* method), 116
- reset_spent_time() (*gitlab.mixins.TimeTrackingMixin* method), 178
- reset_time_estimate() (*gitlab.mixins.TimeTrackingMixin* method), 178
- response_content() (in module *gitlab.utils*), 180
- RESTManager (class in *gitlab.base*), 168
- RESTObject (class in *gitlab.base*), 168
- RESTObjectList (class in *gitlab.base*), 169
- RetrieveMixin (class in *gitlab.mixins*), 177
- retry() (*gitlab.v4.objects.ProjectJob* method), 136
- retry() (*gitlab.v4.objects.ProjectPipeline* method), 150
- Runner (class in *gitlab.v4.objects*), 159
- RunnerJob (class in *gitlab.v4.objects*), 159
- RunnerJobManager (class in *gitlab.v4.objects*), 159
- RunnerManager (class in *gitlab.v4.objects*), 159
- ## S
- sanitized_url() (in module *gitlab.utils*), 180
- save() (*gitlab.mixins.SaveMixin* method), 177
- save() (*gitlab.v4.objects.GroupEpicIssue* method), 102
- save() (*gitlab.v4.objects.GroupLabel* method), 104
- save() (*gitlab.v4.objects.ProjectFile* method), 126
- save() (*gitlab.v4.objects.ProjectLabel* method), 138
- SaveMixin (class in *gitlab.mixins*), 177
- search() (*gitlab.Gitlab* method), 184
- search() (*gitlab.v4.objects.Group* method), 100
- search() (*gitlab.v4.objects.Project* method), 117
- session (*gitlab.Gitlab* attribute), 184
- set() (*gitlab.mixins.SetMixin* method), 177
- set() (*gitlab.v4.objects.FeatureManager* method), 98
- set_approvers() (*gitlab.v4.objects.ProjectApprovalManager* method), 120
- set_approvers() (*gitlab.v4.objects.ProjectMergeRequestApprovalManager* method), 144
- set_license() (*gitlab.Gitlab* method), 184
- set_release_description() (*gitlab.v4.objects.ProjectTag* method), 157
- SetMixin (class in *gitlab.mixins*), 177
- share() (*gitlab.v4.objects.Project* method), 117
- SidekiqManager (class in *gitlab.v4.objects*), 161
- snapshot() (*gitlab.v4.objects.Project* method), 117
- Snippet (class in *gitlab.v4.objects*), 162
- SnippetManager (class in *gitlab.v4.objects*), 162

ssl_verify (*gitlab.Gitlab attribute*), 184
 star() (*gitlab.v4.objects.Project method*), 118
 status() (*gitlab.v4.objects.GeoNode method*), 98
 status() (*gitlab.v4.objects.GeoNodeManager method*), 99
 stop() (*gitlab.v4.objects.ProjectEnvironment method*), 125
 SubscribableMixin (*class in gitlab.mixins*), 178
 subscribe() (*gitlab.mixins.SubscribableMixin method*), 178

T

take_ownership() (*gitlab.v4.objects.ProjectPipelineSchedule method*), 151
 take_ownership() (*gitlab.v4.objects.ProjectTrigger method*), 158
 time_estimate() (*gitlab.mixins.TimeTrackingMixin method*), 179
 time_stats() (*gitlab.mixins.TimeTrackingMixin method*), 179
 timeout (*gitlab.Gitlab attribute*), 185
 TimeTrackingMixin (*class in gitlab.mixins*), 178
 Todo (*class in gitlab.v4.objects*), 163
 todo() (*gitlab.mixins.TodoMixin method*), 179
 TodoManager (*class in gitlab.v4.objects*), 163
 TodoMixin (*class in gitlab.mixins*), 179
 total (*gitlab.base.RESTObjectList attribute*), 169
 total (*gitlab.GitlabList attribute*), 185
 total_pages (*gitlab.base.RESTObjectList attribute*), 169
 total_pages (*gitlab.GitlabList attribute*), 185
 trace() (*gitlab.v4.objects.ProjectJob method*), 136
 transfer_project() (*gitlab.v4.objects.Group method*), 101
 transfer_project() (*gitlab.v4.objects.Project method*), 118
 trigger_pipeline() (*gitlab.v4.objects.Project method*), 118

U

unapprove() (*gitlab.v4.objects.ProjectMergeRequest method*), 144
 unarchive() (*gitlab.v4.objects.Project method*), 118
 unblock() (*gitlab.v4.objects.User method*), 164
 unprotect() (*gitlab.v4.objects.ProjectBranch method*), 121
 unshare() (*gitlab.v4.objects.Project method*), 118
 unstar() (*gitlab.v4.objects.Project method*), 119
 unsubscribe() (*gitlab.mixins.SubscribableMixin method*), 178
 update() (*gitlab.mixins.UpdateMixin method*), 179
 update() (*gitlab.v4.objects.ApplicationSettingsManager method*), 96

update() (*gitlab.v4.objects.GroupLabelManager method*), 105
 update() (*gitlab.v4.objects.ProjectFileManager method*), 129
 update() (*gitlab.v4.objects.ProjectLabelManager method*), 138
 update() (*gitlab.v4.objects.ProjectServiceManager method*), 155
 UpdateMixin (*class in gitlab.mixins*), 179
 upload() (*gitlab.v4.objects.Project method*), 119
 url (*gitlab.Gitlab attribute*), 185
 User (*class in gitlab.v4.objects*), 163
 user_agent_detail() (*gitlab.mixins.UserAgentDetailMixin method*), 180
 UserActivities (*class in gitlab.v4.objects*), 164
 UserActivitiesManager (*class in gitlab.v4.objects*), 164
 UserAgentDetailMixin (*class in gitlab.mixins*), 180
 UserCustomAttribute (*class in gitlab.v4.objects*), 164
 UserCustomAttributeManager (*class in gitlab.v4.objects*), 164
 UserEmail (*class in gitlab.v4.objects*), 164
 UserEmailManager (*class in gitlab.v4.objects*), 164
 UserEvent (*class in gitlab.v4.objects*), 164
 UserEventManager (*class in gitlab.v4.objects*), 164
 UserGPGKey (*class in gitlab.v4.objects*), 164
 UserGPGKeyManager (*class in gitlab.v4.objects*), 164
 UserImpersonationToken (*class in gitlab.v4.objects*), 165
 UserImpersonationTokenManager (*class in gitlab.v4.objects*), 165
 UserKey (*class in gitlab.v4.objects*), 165
 UserKeyManager (*class in gitlab.v4.objects*), 165
 UserManager (*class in gitlab.v4.objects*), 165
 UserProject (*class in gitlab.v4.objects*), 167
 UserProjectManager (*class in gitlab.v4.objects*), 167

V

verify() (*gitlab.v4.objects.RunnerManager method*), 160
 version() (*gitlab.Gitlab method*), 185

W

what_to_cls() (*in module gitlab.cli*), 169