

---

# **python-gitlab Documentation**

*Release 1.0.2*

**Gauvain Pocentek, Mika Mäenpää**

**Sep 29, 2017**



<b>1</b>	<b>Installation</b>	<b>3</b>
<b>2</b>	<b>gitlab CLI usage</b>	<b>5</b>
2.1	Configuration . . . . .	5
2.2	CLI . . . . .	6
2.3	Examples . . . . .	7
<b>3</b>	<b>Getting started with the API</b>	<b>9</b>
3.1	Base types . . . . .	9
3.2	gitlab.Gitlab class . . . . .	9
3.3	API version . . . . .	10
3.4	Managers . . . . .	10
3.5	Gitlab Objects . . . . .	11
3.6	Lazy objects (v4 only) . . . . .	11
3.7	Pagination . . . . .	12
3.8	Sudo . . . . .	12
<b>4</b>	<b>Switching to GitLab API v4</b>	<b>13</b>
4.1	Using the v4 API . . . . .	13
4.2	Changes between v3 and v4 API . . . . .	13
<b>5</b>	<b>API examples</b>	<b>15</b>
5.1	Access requests . . . . .	15
5.2	Branches . . . . .	16
5.3	Protected branches . . . . .	17
5.4	Broadcast messages . . . . .	18
5.5	Jobs (v4 API) / Builds (v3 API) . . . . .	19
5.6	Commits . . . . .	23
5.7	Deploy keys . . . . .	25
5.8	Deployments . . . . .	27
5.9	Environments . . . . .	28
5.10	Groups . . . . .	28
5.11	Issues . . . . .	31
5.12	Labels . . . . .	33
5.13	Notification settings . . . . .	34
5.14	Merge requests . . . . .	36
5.15	Namespaces . . . . .	38

5.16	Milestones	38
5.17	Projects	39
5.18	Runners	54
5.19	Settings	56
5.20	Snippets	56
5.21	System hooks	57
5.22	Templates	58
5.23	Todos	60
5.24	Users	61
5.25	Sidekiq metrics	64
<b>6</b>	<b>gitlab package</b>	<b>65</b>
6.1	Subpackages	65
6.2	Submodules	164
6.3	gitlab.base module	164
6.4	gitlab.cli module	168
6.5	gitlab.config module	168
6.6	gitlab.const module	169
6.7	gitlab.exceptions module	169
6.8	gitlab.mixins module	172
6.9	gitlab.utils module	177
6.10	Module contents	177
<b>7</b>	<b>Release notes</b>	<b>189</b>
7.1	Changes from 0.21 to 1.0.0	189
7.2	Changes from 0.20 to 0.21	189
7.3	Changes from 0.19 to 0.20	190
<b>8</b>	<b>ChangeLog</b>	<b>191</b>
8.1	Version 1.0.2 - 2017-09-29	191
8.2	Version 1.0.1 - 2017-09-21	191
8.3	Version 1.0.0 - 2017-09-08	192
8.4	Version 0.21.2 - 2017-06-11	192
8.5	Version 0.21.1 - 2017-05-25	192
8.6	Version 0.21 - 2017-05-24	192
8.7	Version 0.20 - 2017-03-25	193
8.8	Version 0.19 - 2017-02-21	193
8.9	Version 0.18 - 2016-12-27	194
8.10	Version 0.17 - 2016-12-02	194
8.11	Version 0.16 - 2016-10-16	195
8.12	Version 0.15.1 - 2016-10-16	195
8.13	Version 0.15 - 2016-08-28	195
8.14	Version 0.14 - 2016-08-07	196
8.15	Version 0.13 - 2016-05-16	197
8.16	Version 0.12.2 - 2016-03-19	198
8.17	Version 0.12.1 - 2016-02-03	199
8.18	Version 0.12 - 2016-02-03	199
8.19	Version 0.11.1 - 2016-01-17	199
8.20	Version 0.11 - 2016-01-09	200
8.21	Version 0.10 - 2015-12-29	200
8.22	Version 0.9.2 - 2015-07-11	200
8.23	Version 0.9.1 - 2015-05-15	200
8.24	Version 0.9 - 2015-05-15	201
8.25	Version 0.8 - 2014-10-26	201

8.26	Version 0.7 - 2014-08-21 . . . . .	201
8.27	Version 0.6 - 2014-01-16 . . . . .	202
8.28	Version 0.5 - 2013-12-26 . . . . .	202
8.29	Version 0.4 - 2013-09-26 . . . . .	202
8.30	Version 0.3 - 2013-08-27 . . . . .	202
8.31	Version 0.2 - 2013-08-08 . . . . .	203
8.32	Version 0.1 - 2013-07-08 . . . . .	203
<b>9</b>	<b>Indices and tables</b>	<b>205</b>
	<b>Python Module Index</b>	<b>207</b>



Contents:





# CHAPTER 1

---

## Installation

---

python-gitlab is compatible with python 2 and 3.

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.

## Configuration

### 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.

### 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
api_version = 3

[somewhere]
url = https://some.whe.re
private_token = vTbFeqJYCY3sibBP7BZM
api_version = 4

[elsewhere]
```

```
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 2.1: 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>	<code>Integer</code>	Number of seconds to wait for an answer before failing.

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

Table 2.2: 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 <a href="#">the official documentation</a> to learn how to obtain a token.
<code>api_version</code>	GitLab API version to use (3 or 4). Defaults to 3 for now, but will switch to 4 eventually.
<code>http_username</code>	Username for optional HTTP authentication
<code>http_password</code>	Password for optional HTTP authentication

## CLI

### 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
```

## 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`.

**--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
```

## 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
```

---

## Getting started with the API

---

python-gitlab supports both GitLab v3 and v4 APIs.

v3 being deprecated by GitLab, its support in python-gitlab will be minimal. The development team will focus on v4. v3 is still the default API used by python-gitlab, for compatibility reasons..

### 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.

For v4 the following types are defined:

- `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.

For v3 the following types are defined:

- `gitlab.base.GitlabObject` is the base class for all the GitLab v3 objects. These objects provide an abstraction for GitLab resources (projects, groups, and so on).
- `gitlab.base.BaseManager` is the base class for v3 objects managers, providing the API to manipulate the resources and their attributes.

### `gitlab.Gitlab` class

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

```
import gitlab

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

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

# 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.

## API version

`python-gitlab` uses the v3 GitLab API by default. Use the `api_version` parameter to switch to v4:

```
import gitlab

gl = gitlab.Gitlab('http://10.0.0.1', 'JVNSEs8EwWRx5yDxM5q', api_version=4)
```

**Warning:** The `python-gitlab` API is not the same for v3 and v4. Make sure to read *Switching to GitLab API v4* before upgrading.

v4 will become the default in `python-gitlab`.

## 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 group in groups:
    print()

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



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 python introspection tools for v3, or the `attributes` attribute for v4:

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

# v3
print(vars(project))
# or
print(project.__dict__)

# v4
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()
```

## 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()
```

## Lazy objects (v4 only)

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

The following exemple will only make one API call to the GitLab server to star a project:

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

## 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)
```

---

**Note:** 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.owned(all=True)
```

**Warning:** python-gitlab will iterate over the list by calling the corresponding API multiple times. This might take some time if you have a lot of items to retrieve. This might also consume a lot of memory as all the items will be stored in RAM. If you're encountering the python recursion limit exception, use `safe_all=True` instead to stop pagination automatically if the recursion limit is hit.

With v4, `list()` methods can also return a generator object which will handle the next calls to the API when required:

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

## 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')
```

---

## 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 will stop supporting the v3 API soon, and you should consider switching to v4 if you use a recent version of GitLab ( $\geq 9.0$ ), or if you use <http://gitlab.com>.

### Using the v4 API

To use the new v4 API, explicitly define `api_version` `` in the `Gitlab` constructor:

```
gl = gitlab.Gitlab(..., api_version=4)
```

If you use the configuration file, also explicitly define the version:

```
[my_gitlab]
...
api_version = 4
```

### 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](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 *Gitlab* following 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`

### 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.MASTER_ACCESS: 40`
- `gitlab.OWNER_ACCESS: 50`

### 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`
- v3 API:
  - `gitlab.v3.objects.ProjectAccessRequest`
  - `gitlab.v3.objects.ProjectAccessRequestManager`

- `gitlab.v3.objects.Project.accessrequests`
- `gitlab.Gitlab.project_accessrequests`
- `gitlab.v3.objects.GroupAccessRequest`
- `gitlab.v3.objects.GroupAccessRequestManager`
- `gitlab.v3.objects.Group.accessrequests`
- `gitlab.Gitlab.group_accessrequests`

- GitLab API: [https://docs.gitlab.com/ce/api/access\\_requests.html](https://docs.gitlab.com/ce/api/access_requests.html)

## Examples

List access requests from projects and groups:

```
p_ar = project.accessrequests.list()
g_ar = group.accessrequests.list()
```

Get a single request:

```
p_ar = project.accessrequests.get(user_id)
g_ar = group.accessrequests.get(user_id)
```

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.MASTER_ACCESS) # explicitly set access level
```

Deny (delete) an access request:

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

## Branches

### References

- v4 API:
  - `gitlab.v4.objects.ProjectBranch`
  - `gitlab.v4.objects.ProjectBranchManager`
  - `gitlab.v4.objects.Project.branches`
- v3 API:
  - `gitlab.v3.objects.ProjectBranch`

- `gitlab.v3.objects.ProjectBranchManager`
- `gitlab.v3.objects.Project.branches`
- GitLab API: <https://docs.gitlab.com/ce/api/branches.html>

## 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:

```
# v4
branch = project.branches.create({'branch': 'feature1',
                                 'ref': 'master'})

# v3
branch = project.branches.create({'branch_name': '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)
```

## Protected branches

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

## 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](https://docs.gitlab.com/ce/api/protected_branches.html#protected-branches-api)

## 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'})
```

Delete a protected branch:

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

## 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.

## References

- v4 API:
  - `gitlab.v4.objects.BroadcastMessage`
  - `gitlab.v4.objects.BroadcastMessageManager`
  - `gitlab.Gitlab.broadcastmessages`
- v3 API:
  - `gitlab.v3.objects.BroadcastMessage`
  - `gitlab.v3.objects.BroadcastMessageManager`
  - `gitlab.Gitlab.broadcastmessages`
- GitLab API: [https://docs.gitlab.com/ce/api/broadcast\\_messages.html](https://docs.gitlab.com/ce/api/broadcast_messages.html)

## 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 `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()
```

## Jobs (v4 API) / Builds (v3 API)

Build and job are two classes representing the same object. Builds are used in v3 API, jobs in v4 API.

## 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`
- v3 API:
  - `gitlab.v3.objects.ProjectTrigger`
  - `gitlab.v3.objects.ProjectTriggerManager`
  - `gitlab.v3.objects.Project.triggers`
  - `gitlab.Gitlab.project_triggers`
- GitLab API: [https://docs.gitlab.com/ce/api/pipeline\\_triggers.html](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({})
```

Remove a trigger:

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

## 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*
- v3 API
  - *gitlab.v3.objects.ProjectVariable*
  - *gitlab.v3.objects.ProjectVariableManager*
  - *gitlab.v3.objects.Project.variables*
  - *gitlab.Gitlab.project\_variables*
- GitLab API
  - [https://docs.gitlab.com/ce/api/project\\_level\\_variables.html](https://docs.gitlab.com/ce/api/project_level_variables.html)
  - [https://docs.gitlab.com/ce/api/group\\_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(var_key)
g_var = group.variables.get(var_key)
```

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(var_key)
group.variables.delete(var_key)
# or
var.delete()
```

## Builds/Jobs

Builds/Jobs are associated to projects and commits. They provide information on the builds/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`
- v3 API
  - `gitlab.v3.objects.ProjectJob`
  - `gitlab.v3.objects.ProjectJobManager`
  - `gitlab.v3.objects.Project.jobs`
  - `gitlab.Gitlab.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:

Trigger a new job on a project:

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

List jobs for the project:

```
builds = project.builds.list() # v3
jobs = project.jobs.list() # v4
```

To list builds for a specific commit, create a *ProjectCommit* object and use its *builds* method (v3 only):

```
# v3 only
commit = gl.project_commits.get(commit_sha, project_id=1)
builds = commit.builds()
```

Get a job:

```
project.builds.get(build_id) # v3
project.jobs.get(job_id) # v4
```

Get a job artifact:

```
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
```

Mark a job artifact as kept when expiration is set:

```
build_or_job.keep_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()
```

## Commits

### Commits

#### Reference

- v4 API:
  - `gitlab.v4.objects.ProjectCommit`
  - `gitlab.v4.objects.ProjectCommitManager`
  - `gitlab.v4.objects.Project.commits`
- v3 API:
  - `gitlab.v3.objects.ProjectCommit`
  - `gitlab.v3.objects.ProjectCommitManager`
  - `gitlab.v3.objects.Project.commits`
  - `gitlab.Gitlab.project_commits`
- GitLab API: <https://docs.gitlab.com/ce/api/commits.html>

#### 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')
```

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_name': 'master', # v3
```

```
'branch': 'master', # v4
'commit_message': 'blah blah blah',
'actions': [
    {
        'action': 'create',
        'file_path': 'blah',
        'content': 'blah'
    }
]
}

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')
```

## Commit comments

### Reference

- v4 API:
  - *gitlab.v4.objects.ProjectCommitComment*
  - *gitlab.v4.objects.ProjectCommitCommentManager*
  - *gitlab.v4.objects.Commit.comments*
- v3 API:
  - *gitlab.v3.objects.ProjectCommit*
  - *gitlab.v3.objects.ProjectCommitManager*
  - *gitlab.v3.objects.Commit.comments*
  - *gitlab.v3.objects.Project.commit\_comments*
  - *gitlab.Gitlab.project\_commit\_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'})
```

## Commit status

### Reference

- v4 API:
  - *gitlab.v4.objects.ProjectCommitStatus*
  - *gitlab.v4.objects.ProjectCommitStatusManager*
  - *gitlab.v4.objects.Commit.statuses*
- v3 API:
  - *gitlab.v3.objects.ProjectCommit*
  - *gitlab.v3.objects.ProjectCommitManager*
  - *gitlab.v3.objects.Commit.statuses*
  - *gitlab.v3.objects.Project.commit\_statuses*
  - *gitlab.Gitlab.project\_commit\_statuses*
- GitLab API: <https://docs.gitlab.com/ce/api/commits.html>

### Examples

Get the statuses for a commit:

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

Change the status of a commit:

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

## Deploy keys

### Deploy keys

#### Reference

- v4 API:
  - *gitlab.v4.objects.DeployKey*
  - *gitlab.v4.objects.DeployKeyManager*

- `gitlab.Gitlab.deploykeys`
- v3 API:
  - `gitlab.v3.objects.Key`
  - `gitlab.v3.objects.KeyManager`
  - `gitlab.Gitlab.deploykeys`
- GitLab API: [https://docs.gitlab.com/ce/api/deploy\\_keys.html](https://docs.gitlab.com/ce/api/deploy_keys.html)

## Examples

List the deploy keys:

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

Get a single deploy key:

```
key = gl.deploykeys.get(key_id)
```

## 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`
- v3 API:
  - `gitlab.v3.objects.ProjectKey`
  - `gitlab.v3.objects.ProjectKeyManager`
  - `gitlab.v3.objects.Project.keys`
  - `gitlab.Gitlab.project_keys`
- GitLab API: [https://docs.gitlab.com/ce/api/deploy\\_keys.html](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() # v4
project.keys.disable(key_id) # v3
```

## Deployments

### Reference

- v4 API:
  - *gitlab.v4.objects.ProjectDeployment*
  - *gitlab.v4.objects.ProjectDeploymentManager*
  - *gitlab.v4.objects.Project.deployments*
- v3 API:
  - *gitlab.v3.objects.ProjectDeployment*
  - *gitlab.v3.objects.ProjectDeploymentManager*
  - *gitlab.v3.objects.Project.deployments*
  - *gitlab.Gitlab.project\_deployments*
- GitLab API: <https://docs.gitlab.com/ce/api/deployments.html>

### Examples

List deployments for a project:

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

Get a single deployment:

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

## Environments

### Reference

- v4 API:
  - `gitlab.v4.objects.ProjectEnvironment`
  - `gitlab.v4.objects.ProjectEnvironmentManager`
  - `gitlab.v4.objects.Project.environments`
- v3 API:
  - `gitlab.v3.objects.ProjectEnvironment`
  - `gitlab.v3.objects.ProjectEnvironmentManager`
  - `gitlab.v3.objects.Project.environments`
  - `gitlab.Gitlab.project_environments`
- GitLab API: <https://docs.gitlab.com/ce/api/environments.html>

### Examples

List environments for a project:

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

Get a single environment:

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

Create an environment for a project:

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

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()
```

## Groups

### Groups

#### Reference

- v4 API:

- `gitlab.v4.objects.Group`
- `gitlab.v4.objects.GroupManager`
- `gitlab.Gitlab.groups`
- v3 API:
  - `gitlab.v3.objects.Group`
  - `gitlab.v3.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()
```

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

## Group members

The following constants define the supported access levels:

- `gitlab.GUEST_ACCESS = 10`
- `gitlab.REPORTER_ACCESS = 20`
- `gitlab.DEVELOPER_ACCESS = 30`
- `gitlab.MASTER_ACCESS = 40`
- `gitlab.OWNER_ACCESS = 50`

## Reference

- v4 API:
  - `gitlab.v4.objects.GroupMember`
  - `gitlab.v4.objects.GroupMemberManager`
  - `gitlab.v4.objects.Group.members`
- v3 API:
  - `gitlab.v3.objects.GroupMember`
  - `gitlab.v3.objects.GroupMemberManager`
  - `gitlab.v3.objects.Group.members`
  - `gitlab.Gitlab.group_members`
- GitLab API: <https://docs.gitlab.com/ce/api/groups.html>

## Examples

List group members:

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

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()
```

## Issues

### Reported issues

#### Reference

- v4 API:
  - `gitlab.v4.objects.Issue`
  - `gitlab.v4.objects.IssueManager`
  - `gitlab.Gitlab.issues`
- v3 API:
  - `gitlab.v3.objects.Issue`
  - `gitlab.v3.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'])
```

## Group issues

#### Reference

- v4 API:
  - `gitlab.v4.objects.GroupIssue`
  - `gitlab.v4.objects.GroupIssueManager`
  - `gitlab.v4.objects.Group.issues`
- v3 API:
  - `gitlab.v3.objects.GroupIssue`
  - `gitlab.v3.objects.GroupIssueManager`
  - `gitlab.v3.objects.Group.issues`
  - `gitlab.Gitlab.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')
```

## Project issues

### Reference

- v4 API:
  - `gitlab.v4.objects.ProjectIssue`
  - `gitlab.v4.objects.ProjectIssueManager`
  - `gitlab.v4.objects.Project.issues`
- v3 API:
  - `gitlab.v3.objects.ProjectIssue`
  - `gitlab.v3.objects.ProjectIssueManager`
  - `gitlab.v3.objects.Project.issues`
  - `gitlab.Gitlab.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_id)
```

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(new_project_id)
```

Make an issue as todo:

```
issue.todo()
```

Get time tracking stats:

```
issue.time_stats()
```

Set a time estimate for an issue:

```
issue.set_time_estimate({'duration': '3h30m'})
```

Reset a time estimate for an issue:

```
issue.reset_time_estimate()
```

Add spent time for an issue:

```
issue.add_time_spent({'duration': '3h30m'})
```

Reset spent time for an issue:

```
issue.reset_time_spent()
```

## Labels

### Reference

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

- `gitlab.v4.objects.ProjectLabelManager`
- `gitlab.v4.objects.Project.labels`
- v3 API:
  - `gitlab.v3.objects.ProjectLabel`
  - `gitlab.v3.objects.ProjectLabelManager`
  - `gitlab.v3.objects.Project.labels`
  - `gitlab.Gitlab.project_labels`
- GitLab API: <https://docs.gitlab.com/ce/api/labels.html>

## Examples

List labels for a project:

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

Get a single label:

```
label = project.labels.get(label_name)
```

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()
```

## 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.

## 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`

- v3 API:

- `gitlab.v3.objects.NotificationSettings`
- `gitlab.v3.objects.NotificationSettingsManager`
- `gitlab.Gitlab.notificationsettings`
- `gitlab.v3.objects.GroupNotificationSettings`
- `gitlab.v3.objects.GroupNotificationSettingsManager`
- `gitlab.v3.objects.Group.notificationsettings`
- `gitlab.v3.objects.ProjectNotificationSettings`
- `gitlab.v3.objects.ProjectNotificationSettingsManager`
- `gitlab.v3.objects.Project.notificationsettings`

- GitLab API: [https://docs.gitlab.com/ce/api/notification\\_settings.html](https://docs.gitlab.com/ce/api/notification_settings.html)

## Examples

Get the 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 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()
```

## 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.

The v3 API uses the `id` attribute to identify a merge request, the v4 API uses the `iid` attribute.

## Reference

- v4 API:
  - `gitlab.v4.objects.ProjectMergeRequest`
  - `gitlab.v4.objects.ProjectMergeRequestManager`
  - `gitlab.v4.objects.Project.mergerequests`
- v3 API:
  - `gitlab.v3.objects.ProjectMergeRequest`
  - `gitlab.v3.objects.ProjectMergeRequestManager`
  - `gitlab.v3.objects.Project.mergerequests`
  - `gitlab.Gitlab.project_mergerequests`
- GitLab API: [https://docs.gitlab.com/ce/api/merge\\_requests.html](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:

- `iid`: iid (unique ID for the project) of the MR (v3 API)
- `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()
```

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'})
```

Update a MR:

```
mr.description = 'New description'  
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_build_succeeds() # v3  
mr.cancel_merge_when_pipeline_succeeds() # v4
```

List issues that will close on merge:

```
mr.closes_issues()
```

Subscribe/unsubscribe 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)
```

## Namespaces

### Reference

- v4 API:
  - `gitlab.v4.objects.Namespace`
  - `gitlab.v4.objects.NamespaceManager`
  - `gitlab.Gitlab.namespaces`
- v3 API:
  - `gitlab.v3.objects.Namespace`
  - `gitlab.v3.objects.NamespaceManager`
  - `gitlab.Gitlab.namespaces`
- GitLab API: <https://docs.gitlab.com/ce/api/namespaces.html>

### Examples

List namespaces:

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

Search namespaces:

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

## Milestones

### Reference

- v4 API:
  - `gitlab.v4.objects.ProjectMilestone`
  - `gitlab.v4.objects.ProjectMilestoneManager`
  - `gitlab.v4.objects.Project.milestones`
- v3 API:
  - `gitlab.v3.objects.ProjectMilestone`
  - `gitlab.v3.objects.ProjectMilestoneManager`
  - `gitlab.v3.objects.Project.milestones`
  - `gitlab.Gitlab.project_milestones`
- GitLab API: <https://docs.gitlab.com/ce/api/milestones.html>

## Examples

List the milestones for a project:

```
milestones = project.milestones.list()
```

You can filter the list using the following parameters:

- `iid`: unique ID of the milestone for the project
- `state`: either `active` or `closed`

```
milestones = project.milestones.list(state='closed')
```

Get a single milestone:

```
milestone = project.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()
```

## Projects

### Projects

#### Reference

- v4 API:
  - `gitlab.v4.objects.Project`
  - `gitlab.v4.objects.ProjectManager`

- `gitlab.Gitlab.projects`
- v3 API:
  - `gitlab.v3.objects.Project`
  - `gitlab.v3.objects.ProjectManager`
  - `gitlab.Gitlab.projects`
- GitLab API: <https://docs.gitlab.com/ce/api/projects.html>

## Examples

List projects:

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`)

```
# Active projects
projects = gl.projects.list()
# Archived projects
projects = gl.projects.list(archived=1)
# Limit to projects with a defined visibility
projects = gl.projects.list(visibility='public')

# List owned projects
projects = gl.projects.owned()

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

# List all the projects
projects = gl.projects.all()

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

Get a single project:

```
# Get a project by ID
project = gl.projects.get(10)
# 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'})
```

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()
```

Delete a project:

```
gl.projects.delete(1)
# or
project.delete()
```

Fork a project:

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

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

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

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

Star/unstar a project:

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

Archive/unarchive a project:

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

---

**Note:** Previous versions used `archive_` and `unarchive_` due to a naming issue, they have been deprecated but not yet removed.

---

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 of a file for a commit:

```
file_content = p.repository_blob('master', 'README.rst')
```

Get the repository archive:

```
# get the archive for the default branch
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]

# 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*.

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()
```

## Project files

### Reference

- v4 API:
  - `gitlab.v4.objects.ProjectFile`
  - `gitlab.v4.objects.ProjectFileManager`
  - `gitlab.v4.objects.Project.files`
- v3 API:
  - `gitlab.v3.objects.ProjectFile`



- `gitlab.v3.objects.ProjectFileManager`
- `gitlab.v3.objects.Project.files`
- `gitlab.Gitlab.project_files`

- GitLab API: [https://docs.gitlab.com/ce/api/repository\\_files.html](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:

```
# v4
f = project.files.create({'file_path': 'testfile',
                        'branch': 'master',
                        'content': file_content,
                        'commit_message': 'Create testfile'})

# v3
f = project.files.create({'file_path': 'testfile',
                        'branch_name': 'master',
                        'content': file_content,
                        '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') # v4
f.save(branch_name='master', commit_message='Update testfile') # v3

# 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')
```

## Project tags

### Reference

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

- `gitlab.v4.objects.ProjectTagManager`
- `gitlab.v4.objects.Project.tags`
- v3 API:
  - `gitlab.v3.objects.ProjectTag`
  - `gitlab.v3.objects.ProjectTagManager`
  - `gitlab.v3.objects.Project.tags`
  - `gitlab.Gitlab.project_tags`
- GitLab API: <https://docs.gitlab.com/ce/api/tags.html>

## Examples

List the project tags:

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

Get a tag:

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

## 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`

- v3 API:
  - `gitlab.v3.objects.ProjectSnippet`
  - `gitlab.v3.objects.ProjectSnippetManager`
  - `gitlab.v3.objects.Project.files`
  - `gitlab.Gitlab.project_files`
- GitLab API: [https://docs.gitlab.com/ce/api/project\\_snippets.html](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()
```

## Notes

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

- `ProjectIssue` with `ProjectIssueNote`
- `ProjectMergeRequest` with `ProjectMergeRequestNote`
- `ProjectSnippet` with `ProjectSnippetNote`

## 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`

- v3 API:

Issues:

- `gitlab.v3.objects.ProjectIssueNote`
- `gitlab.v3.objects.ProjectIssueNoteManager`
- `gitlab.v3.objects.ProjectIssue.notes`
- `gitlab.v3.objects.Project.issue_notes`
- `gitlab.Gitlab.project_issue_notes`

MergeRequests:

- `gitlab.v3.objects.ProjectMergeRequestNote`
- `gitlab.v3.objects.ProjectMergeRequestNoteManager`
- `gitlab.v3.objects.ProjectMergeRequest.notes`
- `gitlab.v3.objects.Project.mergerequest_notes`
- `gitlab.Gitlab.project_mergerequest_notes`

Snippets:

- `gitlab.v3.objects.ProjectSnippetNote`
- `gitlab.v3.objects.ProjectSnippetNoteManager`
- `gitlab.v3.objects.ProjectSnippet.notes`
- `gitlab.v3.objects.Project.snippet_notes`
- `gitlab.Gitlab.project_snippet_notes`

- GitLab API: [https://docs.gitlab.com/ce/api/repository\\_files.html](https://docs.gitlab.com/ce/api/repository_files.html)

## 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()
```

## Project events

### Reference

- v4 API:
  - `gitlab.v4.objects.ProjectEvent`
  - `gitlab.v4.objects.ProjectEventManager`
  - `gitlab.v4.objects.Project.events`
- v3 API:
  - `gitlab.v3.objects.ProjectEvent`
  - `gitlab.v3.objects.ProjectEventManager`
  - `gitlab.v3.objects.Project.events`
  - `gitlab.Gitlab.project_events`
- GitLab API: [https://docs.gitlab.com/ce/api/repository\\_files.html](https://docs.gitlab.com/ce/api/repository_files.html)

### Examples

List the project events:

```
gl.project_events.list(project_id=1)
# or
project.events.list()
```

## Project members

### Reference

- v4 API:
  - *gitlab.v4.objects.ProjectMember*
  - *gitlab.v4.objects.ProjectMemberManager*
  - *gitlab.v4.objects.Project.members*
- v3 API:
  - *gitlab.v3.objects.ProjectMember*
  - *gitlab.v3.objects.ProjectMemberManager*
  - *gitlab.v3.objects.Project.members*
  - *gitlab.Gitlab.project\_members*
- GitLab API: <https://docs.gitlab.com/ce/api/members.html>

### Examples

List the project members:

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

Search project members matching a query string:

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

Get a single project member:

```
member = project.members.get(1)
```

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.MASTER_ACCESS
member.save()
```

Remove a member from the project team:

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

Share the project with a group:

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

## Project hooks

### Reference

- v4 API:
  - `gitlab.v4.objects.ProjectHook`
  - `gitlab.v4.objects.ProjectHookManager`
  - `gitlab.v4.objects.Project.hooks`
- v3 API:
  - `gitlab.v3.objects.ProjectHook`
  - `gitlab.v3.objects.ProjectHookManager`
  - `gitlab.v3.objects.Project.hooks`
  - `gitlab.Gitlab.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(1)
```

Create a project hook:

```
hook = gl.project_hooks.create({'url': 'http://my/action/url',
                               'push_events': 1},
                              project_id=1)
# or
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(1)
# or
hook.delete()
```

## Project pipelines

### Reference

- v4 API:
  - `gitlab.v4.objects.ProjectPipeline`
  - `gitlab.v4.objects.ProjectPipelineManager`
  - `gitlab.v4.objects.Project.pipelines`
- v3 API:
  - `gitlab.v3.objects.ProjectPipeline`
  - `gitlab.v3.objects.ProjectPipelineManager`
  - `gitlab.v3.objects.Project.pipelines`
  - `gitlab.Gitlab.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)
```

Retry the failed builds for a pipeline:

```
pipeline.retry()
```

Cancel builds in a pipeline:

```
pipeline.cancel()
```

Create a pipeline for a particular reference:

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

## Project Services

### Reference

- v4 API:
  - `gitlab.v4.objects.ProjectService`
  - `gitlab.v4.objects.ProjectServiceManager`
  - `gitlab.v4.objects.Project.services`
- v3 API:



- `gitlab.v3.objects.ProjectService`
- `gitlab.v3.objects.ProjectServiceManager`
- `gitlab.v3.objects.Project.services`
- `gitlab.Gitlab.project_services`

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

## Examples

Get a service:

```
service = project.services.get(service_name='asana', project_id=1)
# display it's status (enabled/disabled)
print(service.active)
```

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

```
services = gl.project_services.available()
```

Configure and enable a service:

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

Disable a service:

```
service.delete()
```

## Issue boards

Boards are a visual representation of existing issues for a project. 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`
- v3 API:
  - `gitlab.v3.objects.ProjectBoard`
  - `gitlab.v3.objects.ProjectBoardManager`
  - `gitlab.v3.objects.Project.boards`
  - `gitlab.Gitlab.project_boards`
- GitLab API: <https://docs.gitlab.com/ce/api/boards.html>

## Examples

Get the list of existing boards for a project:

```
boards = project.boards.list()
```

Get a single board for a project:

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

## Board lists

### Reference

- v4 API:
  - `gitlab.v4.objects.ProjectBoardList`
  - `gitlab.v4.objects.ProjectBoardListManager`
  - `gitlab.v4.objects.Project.board_lists`
- v3 API:
  - `gitlab.v3.objects.ProjectBoardList`
  - `gitlab.v3.objects.ProjectBoardListManager`
  - `gitlab.v3.objects.ProjectBoard.lists`
  - `gitlab.v3.objects.Project.board_lists`
  - `gitlab.Gitlab.project_board_lists`
- GitLab API: <https://docs.gitlab.com/ce/api/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()
```

## File Uploads

### Reference

- v4 API:
  - `gitlab.v4.objects.Project.upload`
  - `gitlab.v4.objects.ProjectUpload`
- v3 API:
  - `gitlab.v3.objects.Project.upload`
  - `gitlab.v3.objects.ProjectUpload`
- Gitlab API: <https://docs.gitlab.com/ce/api/projects.html#upload-a-file>

### Examples

Upload a file into a project using a filesystem path:

```
# Or provide a full path to the uploaded file
project.upload("filename.txt", filepath="/some/path/filename.txt")
```

Upload a file into a project without a filesystem path:

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

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

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

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

```
})  
# project file upload markdown
```

## 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.

### Global runners (admin)

#### Reference

- v4 API:
  - `gitlab.v4.objects.Runner`
  - `gitlab.v4.objects.RunnerManager`
  - `gitlab.Gitlab.runners`
- v3 API:
  - `gitlab.v3.objects.Runner`
  - `gitlab.v3.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)
```

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()
```

## Project runners

### Reference

- v4 API:
  - `gitlab.v4.objects.ProjectRunner`
  - `gitlab.v4.objects.ProjectRunnerManager`
  - `gitlab.v4.objects.Project.runners`
- v3 API:
  - `gitlab.v3.objects.ProjectRunner`
  - `gitlab.v3.objects.ProjectRunnerManager`
  - `gitlab.v3.objects.Project.runners`
  - `gitlab.Gitlab.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)
```

## Settings

### Reference

- v4 API:
  - `gitlab.v4.objects.ApplicationSettings`
  - `gitlab.v4.objects.ApplicationSettingsManager`
  - `gitlab.Gitlab.settings`
- v3 API:
  - `gitlab.v3.objects.ApplicationSettings`
  - `gitlab.v3.objects.ApplicationSettingsManager`
  - `gitlab.Gitlab.settings`
- GitLab API: <https://docs.gitlab.com/ce/api/commits.html>

### Examples

Get the settings:

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

Update the settings:

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

## Snippets

You can store code snippets in Gitlab. Snippets can be attached to projects (see *Project snippets*), but can also be detached.

- Object class: `gitlab.objects.Namespace`
- Manager object: `gitlab.Gitlab.snippets`

### Examples

List snippets woned 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.raw()
```

**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 a snippet:

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

Delete a snippet:

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

## System hooks

### Reference

- v4 API:
  - *gitlab.v4.objects.Hook*
  - *gitlab.v4.objects.HookManager*
  - *gitlab.Gitlab.hooks*
- v3 API:
  - *gitlab.v3.objects.Hook*
  - *gitlab.v3.objects.HookManager*
  - *gitlab.Gitlab.hooks*
- GitLab API: [https://docs.gitlab.com/ce/api/system\\_hooks.html](https://docs.gitlab.com/ce/api/system_hooks.html)

### Examples

List the system hooks:

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

Create a system hook:

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

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

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

Delete a system hook:

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

## Templates

You can request templates for different type of files:

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

### License templates

#### Reference

- v4 API:
  - `gitlab.v4.objects.License`
  - `gitlab.v4.objects.LicenseManager`
  - `gitlab.Gitlab.licenses`
- v3 API:
  - `gitlab.v3.objects.License`
  - `gitlab.v3.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)
```



## .gitignore templates

### Reference

- v4 API:
  - `gitlab.v4.objects.Gitignore`
  - `gitlab.v4.objects.GitignoreManager`
  - `gitlab.Gitlab.gitignores`
- v3 API:
  - `gitlab.v3.objects.Gitignore`
  - `gitlab.v3.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)
```

## GitLab CI templates

### Reference

- v4 API:
  - `gitlab.v4.objects.Gitlabciyaml`
  - `gitlab.v4.objects.GitlabciyamlManager`
  - `gitlab.Gitlab.gitlabciyaml`
- v3 API:
  - `gitlab.v3.objects.Gitlabciyaml`
  - `gitlab.v3.objects.GitlabciyamlManager`
  - `gitlab.Gitlab.gitlabciyaml`
- GitLab API: [https://docs.gitlab.com/ce/api/templates/gitlab\\_ci\\_ymls.html](https://docs.gitlab.com/ce/api/templates/gitlab_ci_ymls.html)

## Examples

List known GitLab CI templates:

```
gitlabciymls = gl.gitlabciymls.list()
```

Get a GitLab CI template:

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

## Dockerfile templates

### Reference

- v4 API:
  - `gitlab.v4.objects.Dockerfile`
  - `gitlab.v4.objects.DockerfileManager`
  - `gitlab.Gitlab.gitlabciymls`
- 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)
```

## Todos

Use `Todo` objects to manipulate todos. The `gitlab.Gitlab.todos` manager object provides helper functions.

## 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')
```

Get a single todo:

```
todo = gl.todos.get(todo_id)
```

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()
```

## Users

Use `User` objects to manipulate repository branches.

To create `User` objects use the `gitlab.Gitlab.users` manager.

## Examples

Get the list of users:

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

Search users whose username match the given string:

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

Get a single user:

```
# by ID
user = gl.users.get(2)
# 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(2)
user.delete()
```

Block/Unblock a user:

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

## SSH keys

Use the `UserKey` objects to manage user keys.

To create `UserKey` objects use the `User.keys` or `gitlab.Gitlab.user_keys` managers.

### Examples

List SSH keys for a user:

```
keys = gl.user_keys.list(user_id=1)
# or
keys = user.keys.list()
```

Get an SSH key for a user:

```
key = gl.user_keys.list(1, user_id=1)
# or
key = user.keys.get(1)
```

Create an SSH key for a user:

```
k = gl.user_keys.create({'title': 'my_key',
                        'key': open('/home/me/.ssh/id_rsa.pub').read()},
                        user_id=2)
# or
k = user.keys.create({'title': 'my_key',
                     'key': open('/home/me/.ssh/id_rsa.pub').read()})
```

Delete an SSH key for a user:

```
gl.user_keys.delete(1, user_id=1)
# or
user.keys.delete(1)
# or
key.delete()
```

## Emails

Use the `UserEmail` objects to manage user emails.

To create `UserEmail` objects use the `User.emails` or `gitlab.Gitlab.user_emails` managers.

## Examples

List emails for a user:

```
emails = gl.user_emails.list(user_id=1)
# or
emails = user.emails.list()
```

Get an email for a user:

```
email = gl.user_emails.list(1, user_id=1)
# or
email = user.emails.get(1)
```

Create an email for a user:

```
k = gl.user_emails.create({'email': 'foo@bar.com'}, user_id=2)
# or
k = user.emails.create({'email': 'foo@bar.com'})
```

Delete an email for a user:

```
gl.user_emails.delete(1, user_id=1)
# or
user.emails.delete(1)
# or
email.delete()
```

## Current User

Use the `CurrentUser` object to get information about the currently logged-in user.

Use the `CurrentUserKey` objects to manage user keys.

To create `CurrentUserKey` objects use the `gitlab.objects.CurrentUser.keys` manager.

Use the `CurrentUserEmail` objects to manage user emails.

To create `CurrentUserEmail` objects use the `gitlab.objects.CurrentUser.emails` manager.

## Examples

Get the current user:

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

List the current user SSH keys:

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

Get a key for the current user:

```
key = gl.user.keys.get(1)
```

Create a key for the current user:

```
key = gl.user.keys.create({'id': 'my_key', 'key': key_content})
```

Delete a key for the current user:

```
gl.user.keys.delete(1)
# or
key.delete()
```

List the current user emails:

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

Get an email for the current user:

```
email = gl.user.emails.get(1)
```

Create an email for the current user:

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

Delete an email for the current user:

```
gl.user.emails.delete(1)
# or
email.delete()
```

## Sidekiq metrics

### Reference

- v4 API:
  - *gitlab.v4.objects.SidekiqManager*
  - `gitlab.Gitlab.sidekiq`
- v3 API:
  - *gitlab.v3.objects.SidekiqManager*
  - `gitlab.Gitlab.sidekiq`
- GitLab API: [https://docs.gitlab.com/ce/api/sidekiq\\_metrics.html](https://docs.gitlab.com/ce/api/sidekiq_metrics.html)

### Examples

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

## Subpackages

### gitlab.v3 package

#### Submodules

#### gitlab.v3.objects module

**class** `gitlab.v3.objects.ApplicationSettings` (*gl*, *data=None*, *\*\*kwargs*)

Bases: `gitlab.base.GitlabObject`

**save** (*\*\*kwargs*)

Send the modified object to the GitLab server. The following attributes are sent:

- `after_sign_out_path` (optional)
- `container_registry_token_expire_delay` (optional)
- `default_branch_protection` (optional)
- `default_project_visibility` (optional)
- `default_projects_limit` (optional)
- `default_snippet_visibility` (optional)
- `domain_blacklist` (optional)
- `domain_blacklist_enabled` (optional)
- `domain_whitelist` (optional)
- `enabled_git_access_protocol` (optional)
- `gravatar_enabled` (optional)
- `home_page_url` (optional)

- max\_attachment\_size (optional)
- repository\_storage (optional)
- restricted\_signup\_domains (optional)
- restricted\_visibility\_levels (optional)
- session\_expire\_delay (optional)
- sign\_in\_text (optional)
- signin\_enabled (optional)
- signup\_enabled (optional)
- twitter\_sharing\_enabled (optional)
- user\_oauth\_applications (optional)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**canCreate = False**

**canDelete = False**

**canList = False**

**getRequiresId = False**

**optionalUpdateAttrs = ['after\_sign\_out\_path', 'container\_registry\_token\_expire\_delay', 'default\_branch\_protection']**

**class** gitlab.v3.objects.**ApplicationSettingsManager** (*gl, parent=None, args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ApplicationSettings'> objects.

<class 'gitlab.v3.objects.ApplicationSettings'> objects can be updated.

**get** (*\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.ApplicationSettings'>.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ApplicationSettings*

**class** gitlab.v3.objects.**BroadcastMessage** (*gl, data=None, \*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**save** (*\*\*kwargs*)

Send the modified object to the GitLab server. The following attributes are sent:

- message (optional)
- starts\_at (optional)
- ends\_at (optional)
- color (optional)
- font (optional)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)



```

optionalCreateAttrs = ['starts_at', 'ends_at', 'color', 'font']
optionalUpdateAttrs = ['message', 'starts_at', 'ends_at', 'color', 'font']
requiredCreateAttrs = ['message']
requiredUpdateAttrs = []

```

```

class gitlab.v3.objects.BroadcastMessageManager (gl, parent=None, args=[])
Bases: gitlab.base.BaseManager

```

Manager for <class 'gitlab.v3.objects.BroadcastMessage'> objects.

<class 'gitlab.v3.objects.BroadcastMessage'> objects can be updated.

**list** (\*\*kwargs)

Returns a list of objects of type <class 'gitlab.v3.objects.BroadcastMessage'>.

Available keys for kwargs are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (id, \*\*kwargs)

Get a single object of type <class 'gitlab.v3.objects.BroadcastMessage'> using its id.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**create** (data, \*\*kwargs)

Create an object of type <class 'gitlab.v3.objects.BroadcastMessage'>.

data is a dict defining the object attributes. Available attributes are:

- message (required)
- starts\_at (optional)
- ends\_at (optional)
- color (optional)
- font (optional)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**delete** (id, \*\*kwargs)

Delete the object with ID id.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *BroadcastMessage*

```

class gitlab.v3.objects.CurrentUser (gl, data=None, **kwargs)

```

```

Bases: gitlab.base.GitlabObject

```

```

canCreate = False

```

**canDelete** = False

**canList** = False

**canUpdate** = False

**managers** = (('emails', 'CurrentUserEmailManager', [('user\_id', 'id')]), ('keys', 'CurrentUserKeyManager', [('user\_id',

**shortPrintAttr** = 'username')

**class** gitlab.v3.objects.**CurrentUserEmail** (*gl, data=None, \*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**canUpdate** = False

**requiredCreateAttrs** = ['email']

**shortPrintAttr** = 'email'

**class** gitlab.v3.objects.**CurrentUserEmailManager** (*gl, parent=None, args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.CurrentUserEmail'> objects.

<class 'gitlab.v3.objects.CurrentUserEmail'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.CurrentUserEmail'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id, \*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.CurrentUserEmail'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data, \*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.CurrentUserEmail'>.

*data* is a dict defining the object attributes. Available attributes are:

- *email* (required)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**delete** (*id, \*\*kwargs*)

Delete the object with ID *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *CurrentUserEmail*

**class** gitlab.v3.objects.**CurrentUserKey** (*gl, data=None, \*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**canUpdate = False**

**requiredCreateAttrs = ['title', 'key']**

**shortPrintAttr = 'title'**

**class** `gitlab.v3.objects.CurrentUserKeyManager` (*gl*, *parent=None*, *args=[]*)

Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.CurrentUserKey'> objects.

<class 'gitlab.v3.objects.CurrentUserKey'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.CurrentUserKey'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.CurrentUserKey'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data*, *\*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.CurrentUserKey'>.

*data* is a dict defining the object attributes. Available attributes are:

- *title* (required)
- *key* (required)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, *\*\*kwargs*)

Delete the object with ID *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of `CurrentUserKey`

**class** `gitlab.v3.objects.DeployKey` (*gl*, *data=None*, *\*\*kwargs*)

Bases: `gitlab.base.GitlabObject`

**canCreate = False**

**canDelete = False**

**canGet = 'from\_list'**

**canUpdate = False**

**class** `gitlab.v3.objects.DeployKeyManager` (*gl*, *parent=None*, *args=[]*)

Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.DeployKey'> objects.

<class 'gitlab.v3.objects.DeployKey'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.DeployKey'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.DeployKey'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of `DeployKey`

**class** `gitlab.v3.objects.Gitignore` (*gl*, *data=None*, *\*\*kwargs*)

Bases: `gitlab.base.GitlabObject`

**canCreate** = False

**canDelete** = False

**canUpdate** = False

**idAttr** = 'name'

**class** `gitlab.v3.objects.GitignoreManager` (*gl*, *parent=None*, *args=[]*)

Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.Gitignore'> objects.

<class 'gitlab.v3.objects.Gitignore'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.Gitignore'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.Gitignore'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**  
alias of *Gitignore*

**class** `gitlab.v3.objects.Gitlabciyaml` (*gl, data=None, \*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**canCreate = False**

**canDelete = False**

**canUpdate = False**

**idAttr = 'name'**

**class** `gitlab.v3.objects.GitlabciyamlManager` (*gl, parent=None, args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.Gitlabciyaml'> objects.

<class 'gitlab.v3.objects.Gitlabciyaml'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.Gitlabciyaml'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id, \*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.Gitlabciyaml'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**  
alias of *Gitlabciyaml*

**class** `gitlab.v3.objects.Group` (*gl, data=None, \*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**accessrequests**

GroupAccessRequestManager - Manager for objects.

**members**

GroupMemberManager - Manager for objects.

**notificationsettings**

GroupNotificationSettingsManager - Manager for objects.

**projects**

GroupProjectManager - Manager for objects.

**issues**

GroupIssueManager - Manager for objects.

**save** (*\*\*kwargs*)

Send the modified object to the GitLab server. The following attributes are sent:

- *name* (optional)

- `path` (optional)
- `description` (optional)
- `visibility_level` (optional)
- `lfs_enabled` (optional)
- `request_access_enabled` (optional)

Available keys for `kwargs` are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**DEVELOPER\_ACCESS = 30**

**GUEST\_ACCESS = 10**

**MASTER\_ACCESS = 40**

**OWNER\_ACCESS = 50**

**REPORTER\_ACCESS = 20**

**VISIBILITY\_INTERNAL = 10**

**VISIBILITY\_PRIVATE = 0**

**VISIBILITY\_PUBLIC = 20**

**managers** = (('accessrequests', 'GroupAccessRequestManager', [('group\_id', 'id']]), ('members', 'GroupMemberManager'))

**optionalCreateAttrs** = ['description', 'visibility\_level', 'parent\_id', 'lfs\_enabled', 'request\_access\_enabled']

**optionalUpdateAttrs** = ['name', 'path', 'description', 'visibility\_level', 'lfs\_enabled', 'request\_access\_enabled']

**requiredCreateAttrs** = ['name', 'path']

**shortPrintAttr** = 'name'

**transfer\_project** (*id*, *\*\*kwargs*)

Transfers a project to this new groups.

**Parameters** **id** (*int*) – ID of the project to transfer.

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabTransferProjectError` – If the server fails to perform the request.

**class** `gitlab.v3.objects.GroupAccessRequest` (*gl*, *data=None*, *\*\*kwargs*)

Bases: `gitlab.base.GitlabObject`

**approve** (*access\_level=30*, *\*\*kwargs*)

Approve an access request.

**Parameters** **access\_level** (*int*) – The access level for the user.

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabUpdateError` – If the server fails to perform the request.

**canGet** = 'from\_list'

**canUpdate** = False

---

```

class gitlab.v3.objects.GroupAccessRequestManager (gl, parent=None, args=[])
    Bases: gitlab.base.BaseManager

    Manager for <class 'gitlab.v3.objects.GroupAccessRequest'> objects.

    <class 'gitlab.v3.objects.GroupAccessRequest'> objects cannot be updated.

    list (**kwargs)
        Returns a list of objects of type <class 'gitlab.v3.objects.GroupAccessRequest'>.

        Available keys for kwargs are:

            •per_page (int): number of item per page. May be limited by the server.

            •page (int): page to retrieve

            •all (bool): iterate over all the pages and return all the entries

            •sudo (string or int): run the request as another user (requires admin permissions)

    get (id, **kwargs)
        Get a single object of type <class 'gitlab.v3.objects.GroupAccessRequest'> using its id.

        Available keys for kwargs are:

            •sudo (string or int): run the request as another user (requires admin permissions)

    create (data, **kwargs)
        Create an object of type <class 'gitlab.v3.objects.GroupAccessRequest'>.

        data is a dict defining the object attributes. Available attributes are:

        Available keys for kwargs are:

            •sudo (string or int): run the request as another user (requires admin permissions)

    delete (id, **kwargs)
        Delete the object with ID id.

        Available keys for kwargs are:

            •sudo (string or int): run the request as another user (requires admin permissions)

    obj_cls
        alias of GroupAccessRequest

class gitlab.v3.objects.GroupIssue (gl, data=None, **kwargs)
    Bases: gitlab.base.GitlabObject

    canCreate = False
    canDelete = False
    canGet = 'from_list'
    canUpdate = False
    optionalListAttrs = ['state', 'labels', 'milestone', 'order_by', 'sort']
    requiredUrlAttrs = ['group_id']

class gitlab.v3.objects.GroupIssueManager (gl, parent=None, args=[])
    Bases: gitlab.base.BaseManager

    Manager for <class 'gitlab.v3.objects.GroupIssue'> objects.

    <class 'gitlab.v3.objects.GroupIssue'> objects cannot be updated.

```

**list** (\*\*kwargs)

Returns a list of objects of type <class 'gitlab.v3.objects.GroupIssue'>.

Available keys for kwargs are:

- state (optional)
- labels (optional)
- milestone (optional)
- order\_by (optional)
- sort (optional)
- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (id, \*\*kwargs)

Get a single object of type <class 'gitlab.v3.objects.GroupIssue'> using its id.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *GroupIssue*

**class** gitlab.v3.objects.**GroupManager** (gl, parent=None, args=[])

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.Group'> objects.

<class 'gitlab.v3.objects.Group'> objects can be updated.

**list** (\*\*kwargs)

Returns a list of objects of type <class 'gitlab.v3.objects.Group'>.

Available keys for kwargs are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (id, \*\*kwargs)

Get a single object of type <class 'gitlab.v3.objects.Group'> using its id.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**create** (data, \*\*kwargs)

Create an object of type <class 'gitlab.v3.objects.Group'>.

data is a dict defining the object attributes. Available attributes are:

- name (required)
- path (required)



- description (optional)
- visibility\_level (optional)
- parent\_id (optional)
- lfs\_enabled (optional)
- request\_access\_enabled (optional)

Available keys for `kwargs` are:

- sudo (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, **\*\*kwargs**)

Delete the object with ID *id*.

Available keys for `kwargs` are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *Group*

**search** (*query*, **\*\*kwargs**)

Searches groups by name.

#### Parameters

- **query** (*str*) – The search string
- **all** (*bool*) – If True, return all the items, without pagination

**Returns** a list of matching groups.

**Return type** *list(Group)*

**class** `gitlab.v3.objects.GroupMember` (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**save** (**\*\*kwargs**)

Send the modified object to the GitLab server. The following attributes are sent:

- access\_level (required)

Available keys for `kwargs` are:

- sudo (string or int): run the request as another user (requires admin permissions)

**canGet** = 'from\_list'

**optionalCreateAttrs** = ['expires\_at']

**requiredCreateAttrs** = ['access\_level', 'user\_id']

**requiredUpdateAttrs** = ['access\_level']

**requiredUrlAttrs** = ['group\_id']

**shortPrintAttr** = 'username'

**class** `gitlab.v3.objects.GroupMemberManager` (*gl*, *parent=None*, *args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.GroupMember'> objects.

<class 'gitlab.v3.objects.GroupMember'> objects can be updated.

**list** (\*\*kwargs)

Returns a list of objects of type <class 'gitlab.v3.objects.GroupMember'>.

Available keys for kwargs are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (id, \*\*kwargs)

Get a single object of type <class 'gitlab.v3.objects.GroupMember'> using its id.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**create** (data, \*\*kwargs)

Create an object of type <class 'gitlab.v3.objects.GroupMember'>.

data is a dict defining the object attributes. Available attributes are:

- group\_id (required if not discovered on the parent objects)
- access\_level (required)
- user\_id (required)
- expires\_at (optional)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**delete** (id, \*\*kwargs)

Delete the object with ID id.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *GroupMember*

**class** gitlab.v3.objects.**GroupNotificationSettings** (gl, data=None, \*\*kwargs)

Bases: *gitlab.v3.objects.NotificationSettings*

**save** (\*\*kwargs)

Send the modified object to the GitLab server. The following attributes are sent:

- level (optional)
- notification\_email (optional)
- new\_note (optional)
- new\_issue (optional)
- reopen\_issue (optional)
- close\_issue (optional)
- reassign\_issue (optional)
- new\_merge\_request (optional)

- reopen\_merge\_request (optional)
- close\_merge\_request (optional)
- reassign\_merge\_request (optional)
- merge\_merge\_request (optional)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**requiredUrlAttrs = ['group\_id']**

**class** `gitlab.v3.objects.GroupNotificationSettingsManager` (*gl*, *parent=None*, *args=[]*)  
 Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.GroupNotificationSettings'> objects.

<class 'gitlab.v3.objects.GroupNotificationSettings'> objects can be updated.

**get** (*\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.GroupNotificationSettings'>.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of `GroupNotificationSettings`

**class** `gitlab.v3.objects.GroupProject` (*\*args*, *\*\*kwargs*)  
 Bases: `gitlab.v3.objects.Project`

**canCreate = False**

**canDelete = False**

**canGet = 'from\_list'**

**canUpdate = False**

**optionalListAttrs = ['archived', 'visibility', 'order\_by', 'sort', 'search', 'ci\_enabled\_first']**

**class** `gitlab.v3.objects.GroupProjectManager` (*gl*, *parent=None*, *args=[]*)  
 Bases: `gitlab.v3.objects.ProjectManager`

Manager for <class 'gitlab.v3.objects.GroupProject'> objects.

<class 'gitlab.v3.objects.GroupProject'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.GroupProject'>.

Available keys for kwargs are:

- archived (optional)
- visibility (optional)
- order\_by (optional)
- sort (optional)
- search (optional)
- ci\_enabled\_first (optional)
- per\_page (int): number of item per page. May be limited by the server.

- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (*id*, **\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.GroupProject'> using its *id*.

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *GroupProject*

**class** gitlab.v3.objects.**Hook** (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**canUpdate = False**

**requiredCreateAttrs = ['url']**

**shortPrintAttr = 'url'**

**class** gitlab.v3.objects.**HookManager** (*gl*, *parent=None*, *args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.Hook'> objects.

<class 'gitlab.v3.objects.Hook'> objects **cannot** be updated.

**list** (**\*\*kwargs**)

Returns a list of objects of type <class 'gitlab.v3.objects.Hook'>.

Available keys for **kwargs** are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (*id*, **\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.Hook'> using its *id*.

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**create** (*data*, **\*\*kwargs**)

Create an object of type <class 'gitlab.v3.objects.Hook'>.

*data* is a dict defining the object attributes. Available attributes are:

- url (required)

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, **\*\*kwargs**)

Delete the object with ID *id*.

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**  
alias of *Hook*

**class** `gitlab.v3.objects.Issue` (*gl*, *data=None*, *\*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**canCreate** = False

**canDelete** = False

**canGet** = 'from\_list'

**canUpdate** = False

**optionalListAttrs** = ['state', 'labels', 'order\_by', 'sort']

**shortPrintAttr** = 'title'

**class** `gitlab.v3.objects.IssueManager` (*gl*, *parent=None*, *args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.Issue'> objects.

<class 'gitlab.v3.objects.Issue'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.Issue'>.

Available keys for *kwargs* are:

- *state* (optional)
- *labels* (optional)
- *order\_by* (optional)
- *sort* (optional)
- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.Issue'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**  
alias of *Issue*

**class** `gitlab.v3.objects.Key` (*\*args*, *\*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**canCreate** = False

**canDelete** = False

**canGet** = 'from\_list'

**canUpdate** = False

**class** `gitlab.v3.objects.KeyManager` (*gl*, *parent=None*, *args=[]*)

Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.Key'> objects.

<class 'gitlab.v3.objects.Key'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.Key'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.Key'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of `Key`

**class** `gitlab.v3.objects.License` (*gl*, *data=None*, *\*\*kwargs*)

Bases: `gitlab.base.GitlabObject`

**canCreate** = `False`

**canDelete** = `False`

**canUpdate** = `False`

**idAttr** = `'key'`

**optionalGetAttrs** = `['project', 'fullname']`

**optionalListAttrs** = `['popular']`

**class** `gitlab.v3.objects.LicenseManager` (*gl*, *parent=None*, *args=[]*)

Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.License'> objects.

<class 'gitlab.v3.objects.License'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.License'>.

Available keys for *kwargs* are:

- *popular* (optional)
- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, **\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.License'> using its *id*.

Available keys for **kwargs** are:

- **project** (optional)
- **fullname** (optional)
- **sudo** (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *License*

**class** `gitlab.v3.objects.Namespace` (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**canCreate** = False

**canDelete** = False

**canGet** = 'from\_list'

**canUpdate** = False

**optionalListAttrs** = ['search']

**class** `gitlab.v3.objects.NamespaceManager` (*gl*, *parent=None*, *args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.Namespace'> objects.

<class 'gitlab.v3.objects.Namespace'> objects **cannot** be updated.

**list** (**\*\*kwargs**)

Returns a list of objects of type <class 'gitlab.v3.objects.Namespace'>.

Available keys for **kwargs** are:

- **search** (optional)
- **per\_page** (int): number of item per page. May be limited by the server.
- **page** (int): page to retrieve
- **all** (bool): iterate over all the pages and return all the entries
- **sudo** (string or int): run the request as another user (requires admin permissions)

**get** (*id*, **\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.Namespace'> using its *id*.

Available keys for **kwargs** are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *Namespace*

**class** `gitlab.v3.objects.NotificationSettings` (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**save** (**\*\*kwargs**)

Send the modified object to the GitLab server. The following attributes are sent:

- **level** (optional)
- **notification\_email** (optional)

- new\_note (optional)
- new\_issue (optional)
- reopen\_issue (optional)
- close\_issue (optional)
- reassign\_issue (optional)
- new\_merge\_request (optional)
- reopen\_merge\_request (optional)
- close\_merge\_request (optional)
- reassign\_merge\_request (optional)
- merge\_merge\_request (optional)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**canCreate = False**

**canDelete = False**

**canList = False**

**getRequiresId = False**

**optionalUpdateAttrs = ['level', 'notification\_email', 'new\_note', 'new\_issue', 'reopen\_issue', 'close\_issue', 'reassign**

**class** gitlab.v3.objects.**NotificationSettingsManager** (*gl, parent=None, args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.NotificationSettings'> objects.

<class 'gitlab.v3.objects.NotificationSettings'> objects can be updated.

**get** (*\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.NotificationSettings'>.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *NotificationSettings*

**class** gitlab.v3.objects.**Project** (*gl, data=None, \*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**accessrequests**

ProjectAccessRequestManager - Manager for objects.

**boards**

ProjectBoardManager - Manager for objects.

**board\_lists**

ProjectBoardListManager - Manager for objects.

**branches**

ProjectBranchManager - Manager for objects.

**builds**

ProjectBuildManager - Manager for objects.



**commits**

ProjectCommitManager - Manager for objects.

**deployments**

ProjectDeploymentManager - Manager for objects.

**environments**

ProjectEnvironmentManager - Manager for objects.

**events**

ProjectEventManager - Manager for objects.

**files**

ProjectFileManager - Manager for objects.

**forks**

ProjectForkManager - Manager for objects.

**hooks**

ProjectHookManager - Manager for objects.

**keys**

ProjectKeyManager - Manager for objects.

**issues**

ProjectIssueManager - Manager for objects.

**labels**

ProjectLabelManager - Manager for objects.

**members**

ProjectMemberManager - Manager for objects.

**mergerequests**

ProjectMergeRequestManager - Manager for objects.

**milestones**

ProjectMilestoneManager - Manager for objects.

**notes**

ProjectNoteManager - Manager for objects.

**notificationsettings**

ProjectNotificationSettingsManager - Manager for objects.

**pipelines**

ProjectPipelineManager - Manager for objects.

**runners**

ProjectRunnerManager - Manager for objects.

**services**

ProjectServiceManager - Manager for objects.

**snippets**

ProjectSnippetManager - Manager for objects.

**tags**

ProjectTagManager - Manager for objects.

**triggers**

ProjectTriggerManager - Manager for objects.

### variables

ProjectVariableManager - Manager for objects.

### save (\*\*kwargs)

Send the modified object to the GitLab server. The following attributes are sent:

- name (optional)
- path (optional)
- default\_branch (optional)
- description (optional)
- issues\_enabled (optional)
- merge\_requests\_enabled (optional)
- builds\_enabled (optional)
- wiki\_enabled (optional)
- snippets\_enabled (optional)
- container\_registry\_enabled (optional)
- shared\_runners\_enabled (optional)
- public (optional)
- visibility\_level (optional)
- import\_url (optional)
- public\_builds (optional)
- only\_allow\_merge\_if\_build\_succeeds (optional)
- only\_allow\_merge\_if\_all\_discussions\_are\_resolved (optional)
- lfs\_enabled (optional)
- request\_access\_enabled (optional)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**VISIBILITY\_INTERNAL = 10**

**VISIBILITY\_PRIVATE = 0**

**VISIBILITY\_PUBLIC = 20**

### archive (\*\*kwargs)

Archive a project.

**Returns** the updated Project

**Return type** *Project*

### Raises

- GitlabCreateError – If the action cannot be done
- GitlabConnectionError – If the server cannot be reached.

### archive\_ (\*\*kwargs)

**create\_fork\_relation** (*forked\_from\_id*)

Create a forked from/to relation between existing projects.

**Parameters** **forked\_from\_id** (*int*) – The ID of the project that was forked from

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabCreateError` – If the server fails to perform the request.

**delete\_fork\_relation** ()

Delete a forked relation between existing projects.

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabDeleteError` – If the server fails to perform the request.

**managers** = (('accessrequests', 'ProjectAccessRequestManager', [('project\_id', 'id'])), ('boards', 'ProjectBoardManager'

**optionalCreateAttrs** = ['path', 'namespace\_id', 'description', 'issues\_enabled', 'merge\_requests\_enabled', 'builds'

**optionalListAttrs** = ['search']

**optionalUpdateAttrs** = ['name', 'path', 'default\_branch', 'description', 'issues\_enabled', 'merge\_requests\_enabled']

**repository\_archive** (*sha=None, streamed=False, action=None, chunk\_size=1024, \*\*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.

**Returns** The binary data of the archive.

**Return type** str

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabGetError` – If the server fails to perform the request.

**repository\_blob** (*sha, filepath, streamed=False, action=None, chunk\_size=1024, \*\*kwargs*)

Return the content of a file for a commit.

**Parameters**

- **sha** (*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.

**Returns** The file content

**Return type** str

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabGetError` – If the server fails to perform the request.

**repository\_compare** (*from\_, to, \*\*kwargs*)

Returns a diff between two branches/commits.

**Parameters**

- **from** (*str*) – orig branch/SHA
- **to** (*str*) – dest branch/SHA

**Returns** The diff

**Return type** str

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabGetError` – If the server fails to perform the request.

**repository\_contributors** ()

Returns a list of contributors for the project.

**Returns** The contributors

**Return type** *list*

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabGetError` – If the server fails to perform the request.

**repository\_raw\_blob** (*sha, streamed=False, action=None, chunk\_size=1024, \*\*kwargs*)

Returns the raw file contents for a blob by blob SHA.

**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.

**Returns** The blob content

**Return type** str

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabGetError` – If the server fails to perform the request.

**repository\_tree** (*path='', ref\_name='', \*\*kwargs*)

Return a list of files in the repository.

**Parameters**

- **path** (*str*) – Path of the top folder (/ by default)

- **ref\_name** (*str*) – Reference to a commit or branch

**Returns** The json representation of the tree.

**Return type** *str*

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabGetError` – If the server fails to perform the request.

**requiredCreateAttrs** = ['name']

**share** (*group\_id*, *group\_access*, *\*\*kwargs*)

Share the project with a group.

**Parameters**

- **group\_id** (*int*) – ID of the group.
- **group\_access** (*int*) – Access level for the group.

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabCreateError` – If the server fails to perform the request.

**shortPrintAttr** = 'path'

**star** (*\*\*kwargs*)

Star a project.

**Returns** the updated Project

**Return type** *Project*

**Raises**

- `GitlabCreateError` – If the action cannot be done
- `GitlabConnectionError` – If the server cannot be reached.

**trigger\_build** (*ref*, *token*, *variables={}*, *\*\*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 commit SHA, a branch name, ...
- **token** (*str*) – The trigger token
- **variables** (*dict*) – Variables passed to the build script

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabCreateError` – If the server fails to perform the request.

**unarchive** (*\*\*kwargs*)

Unarchive a project.

**Returns** the updated Project

**Return type** *Project*

**Raises**

- `GitlabDeleteError` – If the action cannot be done
- `GitlabConnectionError` – If the server cannot be reached.

`unarchive_` (\*\*kwargs)

`unstar` (\*\*kwargs)

Unstar a project.

**Returns** the updated Project

**Return type** *Project*

**Raises**

- `GitlabDeleteError` – If the action cannot be done
- `GitlabConnectionError` – If the server cannot be reached.

`upload` (filename, filedata=None, filepath=None, \*\*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.v3.objects.ProjectAccessRequest` (*gl*, *data=None*, \*\*kwargs)

Bases: `gitlab.base.GitlabObject`

`approve` (*access\_level=30*, \*\*kwargs)

Approve an access request.

**Parameters** **access\_level** (*int*) – The access level for the user.

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabUpdateError` – If the server fails to perform the request.

**canGet** = 'from\_list'

**canUpdate** = False

**class** `gitlab.v3.objects.ProjectAccessRequestManager` (*gl*, *parent=None*, *args=[]*)  
 Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.ProjectAccessRequest'> objects.

<class 'gitlab.v3.objects.ProjectAccessRequest'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectAccessRequest'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.ProjectAccessRequest'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data*, *\*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.ProjectAccessRequest'>.

*data* is a dict defining the object attributes. Available attributes are:

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, *\*\*kwargs*)

Delete the object with ID *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of `ProjectAccessRequest`

**class** `gitlab.v3.objects.ProjectBoard` (*gl*, *data=None*, *\*\*kwargs*)  
 Bases: `gitlab.base.GitlabObject`

**canCreate** = False

**canDelete** = False

**canGet** = 'from\_list'

**canUpdate** = False

**managers** = (('lists', 'ProjectBoardListManager', [(('project\_id', 'project\_id'), ('board\_id', 'id'))]),)

**requiredUrlAttrs** = ['project\_id']

**class** `gitlab.v3.objects.ProjectBoardList` (*gl*, *data=None*, *\*\*kwargs*)  
Bases: `gitlab.base.GitlabObject`

**save** (*\*\*kwargs*)  
Send the modified object to the GitLab server. The following attributes are sent:

- `position` (required)

Available keys for `kwargs` are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**requiredCreateAttrs** = ['label\_id']

**requiredUpdateAttrs** = ['position']

**requiredUrlAttrs** = ['project\_id', 'board\_id']

**class** `gitlab.v3.objects.ProjectBoardListManager` (*gl*, *parent=None*, *args=[]*)  
Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.ProjectBoardList'> objects.  
<class 'gitlab.v3.objects.ProjectBoardList'> objects can be updated.

**list** (*\*\*kwargs*)  
Returns a list of objects of type <class 'gitlab.v3.objects.ProjectBoardList'>.

Available keys for `kwargs` are:

- `per_page` (int): number of item per page. May be limited by the server.
- `page` (int): page to retrieve
- `all` (bool): iterate over all the pages and return all the entries
- `sudo` (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)  
Get a single object of type <class 'gitlab.v3.objects.ProjectBoardList'> using its `id`.

Available keys for `kwargs` are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**create** (*data*, *\*\*kwargs*)  
Create an object of type <class 'gitlab.v3.objects.ProjectBoardList'>.

`data` is a dict defining the object attributes. Available attributes are:

- `project_id` (required if not discovered on the parent objects)
- `board_id` (required if not discovered on the parent objects)
- `label_id` (required)

Available keys for `kwargs` are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, *\*\*kwargs*)  
Delete the object with ID `id`.

Available keys for `kwargs` are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**obj\_cls**  
alias of `ProjectBoardList`



---

```

class gitlab.v3.objects.ProjectBoardManager (gl, parent=None, args=[])
    Bases: gitlab.base.BaseManager

    Manager for <class 'gitlab.v3.objects.ProjectBoard'> objects.

    <class 'gitlab.v3.objects.ProjectBoard'> objects cannot be updated.

    list (**kwargs)
        Returns a list of objects of type <class 'gitlab.v3.objects.ProjectBoard'>.

        Available keys for kwargs are:

        •per_page (int): number of item per page. May be limited by the server.

        •page (int): page to retrieve

        •all (bool): iterate over all the pages and return all the entries

        •sudo (string or int): run the request as another user (requires admin permissions)

    get (id, **kwargs)
        Get a single object of type <class 'gitlab.v3.objects.ProjectBoard'> using its id.

        Available keys for kwargs are:

        •sudo (string or int): run the request as another user (requires admin permissions)

    obj_cls
        alias of ProjectBoard

class gitlab.v3.objects.ProjectBranch (gl, data=None, **kwargs)
    Bases: gitlab.base.GitlabObject

    canUpdate = False

    idAttr = 'name'

    protect (protect=True, **kwargs)
        Protects the branch.

    requiredCreateAttrs = ['branch_name', 'ref']

    requiredUrlAttrs = ['project_id']

    unprotect (**kwargs)
        Unprotects the branch.

class gitlab.v3.objects.ProjectBranchManager (gl, parent=None, args=[])
    Bases: gitlab.base.BaseManager

    Manager for <class 'gitlab.v3.objects.ProjectBranch'> objects.

    <class 'gitlab.v3.objects.ProjectBranch'> objects cannot be updated.

    list (**kwargs)
        Returns a list of objects of type <class 'gitlab.v3.objects.ProjectBranch'>.

        Available keys for kwargs are:

        •per_page (int): number of item per page. May be limited by the server.

        •page (int): page to retrieve

        •all (bool): iterate over all the pages and return all the entries

        •sudo (string or int): run the request as another user (requires admin permissions)

```

**get** (*id*, **\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.ProjectBranch'> using its *id*.

Available keys for **kwargs** are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**create** (*data*, **\*\*kwargs**)

Create an object of type <class 'gitlab.v3.objects.ProjectBranch'>.

*data* is a dict defining the object attributes. Available attributes are:

- **project\_id** (required if not discovered on the parent objects)
- **branch\_name** (required)
- **ref** (required)

Available keys for **kwargs** are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, **\*\*kwargs**)

Delete the object with ID *id*.

Available keys for **kwargs** are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectBranch*

**class** gitlab.v3.objects.**ProjectBuild** (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**artifacts** (*streamed=False*, *action=None*, *chunk\_size=1024*, **\*\*kwargs**)

Get the build 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.

**Returns** The artifacts if *streamed* is False, None otherwise.

**Return type** str

#### Raises

- *GitlabConnectionError* – If the server cannot be reached.
- *GitlabGetError* – If the artifacts are not available.

**canCreate** = False

**canDelete** = False

**canUpdate** = False

**cancel** (**\*\*kwargs**)

Cancel the build.

**erase** (**\*\*kwargs**)

Erase the build (remove build artifacts and trace).

**keep\_artifacts** (*\*\*kwargs*)

Prevent artifacts from being delete when expiration is set.

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabCreateError` – If the request failed.

**play** (*\*\*kwargs*)

Trigger a build explicitly.

**requiredUrlAttrs** = ['project\_id']

**retry** (*\*\*kwargs*)

Retry the build.

**trace** (*streamed=False, action=None, chunk\_size=1024, \*\*kwargs*)

Get the build 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.

**Returns** The trace.

**Return type** str

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabGetError` – If the trace is not available.

**class** `gitlab.v3.objects.ProjectBuildManager` (*gl, parent=None, args=[]*)

Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.ProjectBuild'> objects.

<class 'gitlab.v3.objects.ProjectBuild'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectBuild'>.

Available keys for *kwargs* are:

- **per\_page** (int): number of item per page. May be limited by the server.
- **page** (int): page to retrieve
- **all** (bool): iterate over all the pages and return all the entries
- **sudo** (string or int): run the request as another user (requires admin permissions)

**get** (*id, \*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.ProjectBuild'> using its *id*.

Available keys for *kwargs* are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of `ProjectBuild`

**class** `gitlab.v3.objects.ProjectCommit` (*gl*, *data=None*, *\*\*kwargs*)

Bases: `gitlab.base.GitlabObject`

**blob** (*filepath*, *streamed=False*, *action=None*, *chunk\_size=1024*, *\*\*kwargs*)

Generate the content of a file for this commit.

**Parameters**

- **filepath** (*str*) – Path of the file to request.
- **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.

**Returns** The content of the file

**Return type** `str`

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabGetError` – If the server fails to perform the request.

**builds** (*\*\*kwargs*)

List the build for this commit.

**Returns** A list of builds.

**Return type** `list(ProjectBuild)`

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabListError` – If the server fails to perform the request.

**canDelete** = `False`

**canUpdate** = `False`

**cherry\_pick** (*branch*, *\*\*kwargs*)

Cherry-pick a commit into a branch.

**Parameters** **branch** (*str*) – Name of target branch.

**Raises** `GitlabCherryPickError` – If the cherry pick could not be applied.

**diff** (*\*\*kwargs*)

Generate the commit diff.

**managers** = (('comments', 'ProjectCommitCommentManager', [('project\_id', 'project\_id'), ('commit\_id', 'id')]), ('status', 'ProjectCommitStatusManager', [('project\_id', 'project\_id'), ('commit\_id', 'id')]))

**optionalCreateAttrs** = ['author\_email', 'author\_name']

**requiredCreateAttrs** = ['branch\_name', 'commit\_message', 'actions']

**requiredUrlAttrs** = ['project\_id']

**shortPrintAttr** = 'title'

**class** `gitlab.v3.objects.ProjectCommitComment` (*gl*, *data=None*, *\*\*kwargs*)

Bases: `gitlab.base.GitlabObject`

**canDelete** = `False`

```

canGet = False
canUpdate = False
optionalCreateAttrs = ['path', 'line', 'line_type']
requiredCreateAttrs = ['note']
requiredUrlAttrs = ['project_id', 'commit_id']

```

```

class gitlab.v3.objects.ProjectCommitCommentManager (gl, parent=None, args=[])
    Bases: gitlab.base.BaseManager

```

Manager for <class 'gitlab.v3.objects.ProjectCommitComment'> objects.

<class 'gitlab.v3.objects.ProjectCommitComment'> objects **cannot** be updated.

**list** (\*\*kwargs)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectCommitComment'>.

Available keys for kwargs are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**create** (data, \*\*kwargs)

Create an object of type <class 'gitlab.v3.objects.ProjectCommitComment'>.

data is a dict defining the object attributes. Available attributes are:

- project\_id (required if not discovered on the parent objects)
- commit\_id (required if not discovered on the parent objects)
- note (required)
- path (optional)
- line (optional)
- line\_type (optional)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectCommitComment*

```

class gitlab.v3.objects.ProjectCommitManager (gl, parent=None, args=[])
    Bases: gitlab.base.BaseManager

```

Manager for <class 'gitlab.v3.objects.ProjectCommit'> objects.

<class 'gitlab.v3.objects.ProjectCommit'> objects **cannot** be updated.

**list** (\*\*kwargs)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectCommit'>.

Available keys for kwargs are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve

- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (*id*, **\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.ProjectCommit'> using its *id*.

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**create** (*data*, **\*\*kwargs**)

Create an object of type <class 'gitlab.v3.objects.ProjectCommit'>.

*data* is a dict defining the object attributes. Available attributes are:

- project\_id (required if not discovered on the parent objects)
- branch\_name (required)
- commit\_message (required)
- actions (required)
- author\_email (optional)
- author\_name (optional)

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectCommit*

**class** gitlab.v3.objects.**ProjectCommitStatus** (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**canDelete** = False

**canUpdate** = False

**optionalCreateAttrs** = ['description', 'name', 'context', 'ref', 'target\_url']

**optionalGetAttrs** = ['ref\_name', 'stage', 'name', 'all']

**requiredCreateAttrs** = ['state']

**requiredUrlAttrs** = ['project\_id', 'commit\_id']

**class** gitlab.v3.objects.**ProjectCommitStatusManager** (*gl*, *parent=None*, *args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectCommitStatus'> objects.

<class 'gitlab.v3.objects.ProjectCommitStatus'> objects **cannot** be updated.

**list** (**\*\*kwargs**)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectCommitStatus'>.

Available keys for **kwargs** are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (*id*, **\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.ProjectCommitStatus'> using its *id*.

Available keys for **kwargs** are:

- **ref\_name** (optional)
- **stage** (optional)
- **name** (optional)
- **all** (optional)
- **sudo** (string or int): run the request as another user (requires admin permissions)

**create** (*data*, **\*\*kwargs**)

Create an object of type <class 'gitlab.v3.objects.ProjectCommitStatus'>.

*data* is a dict defining the object attributes. Available attributes are:

- **project\_id** (required if not discovered on the parent objects)
- **commit\_id** (required if not discovered on the parent objects)
- **state** (required)
- **description** (optional)
- **name** (optional)
- **context** (optional)
- **ref** (optional)
- **target\_url** (optional)

Available keys for **kwargs** are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectCommitStatus*

**class** `gitlab.v3.objects.ProjectDeployment` (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**canCreate = False**

**canDelete = False**

**canUpdate = False**

**class** `gitlab.v3.objects.ProjectDeploymentManager` (*gl*, *parent=None*, *args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectDeployment'> objects.

<class 'gitlab.v3.objects.ProjectDeployment'> objects **cannot** be updated.

**list** (**\*\*kwargs**)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectDeployment'>.

Available keys for **kwargs** are:

- **per\_page** (int): number of item per page. May be limited by the server.
- **page** (int): page to retrieve
- **all** (bool): iterate over all the pages and return all the entries

- sudo (string or int): run the request as another user (requires admin permissions)

**get** (*id*, **\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.ProjectDeployment'> using its *id*.

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectDeployment*

**class** gitlab.v3.objects.**ProjectEnvironment** (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**save** (**\*\*kwargs**)

Send the modified object to the GitLab server. The following attributes are sent:

- name (optional)
- external\_url (optional)

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**canGet** = 'from\_list'

**optionalCreateAttrs** = ['external\_url']

**optionalUpdateAttrs** = ['name', 'external\_url']

**requiredCreateAttrs** = ['name']

**requiredUrlAttrs** = ['project\_id']

**class** gitlab.v3.objects.**ProjectEnvironmentManager** (*gl*, *parent=None*, *args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectEnvironment'> objects.

<class 'gitlab.v3.objects.ProjectEnvironment'> objects can be updated.

**list** (**\*\*kwargs**)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectEnvironment'>.

Available keys for **kwargs** are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (*id*, **\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.ProjectEnvironment'> using its *id*.

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**create** (*data*, **\*\*kwargs**)

Create an object of type <class 'gitlab.v3.objects.ProjectEnvironment'>.

*data* is a dict defining the object attributes. Available attributes are:

- project\_id (required if not discovered on the parent objects)



- name (required)
- external\_url (optional)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, **\*\*kwargs**)

Delete the object with ID *id*.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectEnvironment*

**class** `gitlab.v3.objects.ProjectEvent` (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**canCreate** = False

**canDelete** = False

**canGet** = 'from\_list'

**canUpdate** = False

**requiredUrlAttrs** = ['project\_id']

**shortPrintAttr** = 'target\_title'

**class** `gitlab.v3.objects.ProjectEventManager` (*gl*, *parent=None*, *args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectEvent'> objects.

<class 'gitlab.v3.objects.ProjectEvent'> objects **cannot** be updated.

**list** (**\*\*kwargs**)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectEvent'>.

Available keys for kwargs are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (*id*, **\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.ProjectEvent'> using its *id*.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectEvent*

**class** `gitlab.v3.objects.ProjectFile` (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**save** (**\*\*kwargs**)

Send the modified object to the GitLab server. The following attributes are sent:

- file\_path (required)
- branch\_name (required)
- content (required)
- commit\_message (required)
- encoding (optional)

Available keys for `kwargs` are:

- sudo (string or int): run the request as another user (requires admin permissions)

**canList = False**

**decode ()**

Returns the decoded content of the file.

**Returns** the decoded content.

**Return type** (str)

**getRequiresId = False**

**optionalCreateAttrs = ['encoding']**

**requiredCreateAttrs = ['file\_path', 'branch\_name', 'content', 'commit\_message']**

**requiredDeleteAttrs = ['branch\_name', 'commit\_message', 'file\_path']**

**requiredGetAttrs = ['file\_path', 'ref']**

**requiredUrlAttrs = ['project\_id']**

**shortPrintAttr = 'file\_path'**

**class** gitlab.v3.objects.**ProjectFileManager** (*gl, parent=None, args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectFile'> objects.

<class 'gitlab.v3.objects.ProjectFile'> objects can be updated.

**get** (*\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.ProjectFile'>.

Available keys for `kwargs` are:

- file\_path (required)
- ref (required)
- sudo (string or int): run the request as another user (requires admin permissions)

**create** (*data, \*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.ProjectFile'>.

`data` is a dict defining the object attributes. Available attributes are:

- project\_id (required if not discovered on the parent objects)
- file\_path (required)
- branch\_name (required)
- content (required)
- commit\_message (required)

- encoding (optional)

Available keys for `kwargs` are:

- sudo (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, **\*\*kwargs**)

Delete the object with ID *id*.

Available keys for `kwargs` are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectFile*

**class** `gitlab.v3.objects.ProjectFork` (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**canDelete** = False

**canGet** = False

**canList** = False

**canUpdate** = False

**optionalCreateAttrs** = ['namespace']

**requiredUrlAttrs** = ['project\_id']

**class** `gitlab.v3.objects.ProjectForkManager` (*gl*, *parent=None*, *args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectFork'> objects.

<class 'gitlab.v3.objects.ProjectFork'> objects **cannot** be updated.

**create** (*data*, **\*\*kwargs**)

Create an object of type <class 'gitlab.v3.objects.ProjectFork'>.

*data* is a dict defining the object attributes. Available attributes are:

- project\_id (required if not discovered on the parent objects)
- namespace (optional)

Available keys for `kwargs` are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectFork*

**class** `gitlab.v3.objects.ProjectHook` (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**save** (**\*\*kwargs**)

Send the modified object to the GitLab server. The following attributes are sent:

- url (required)
- push\_events (optional)
- issues\_events (optional)
- note\_events (optional)
- merge\_requests\_events (optional)

- tag\_push\_events (optional)
- build\_events (optional)
- enable\_ssl\_verification (optional)
- token (optional)
- pipeline\_events (optional)

Available keys for `kwargs` are:

- sudo (string or int): run the request as another user (requires admin permissions)

**optionalCreateAttrs** = ['push\_events', 'issues\_events', 'note\_events', 'merge\_requests\_events', 'tag\_push\_events', 'build\_events']

**requiredCreateAttrs** = ['url']

**requiredUrlAttrs** = ['project\_id']

**shortPrintAttr** = 'url'

**class** gitlab.v3.objects.**ProjectHookManager** (*gl, parent=None, args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectHook'> objects.

<class 'gitlab.v3.objects.ProjectHook'> objects can be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectHook'>.

Available keys for `kwargs` are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (*id, \*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.ProjectHook'> using its `id`.

Available keys for `kwargs` are:

- sudo (string or int): run the request as another user (requires admin permissions)

**create** (*data, \*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.ProjectHook'>.

`data` is a dict defining the object attributes. Available attributes are:

- project\_id (required if not discovered on the parent objects)
- url (required)
- push\_events (optional)
- issues\_events (optional)
- note\_events (optional)
- merge\_requests\_events (optional)
- tag\_push\_events (optional)
- build\_events (optional)

- `enable_ssl_verification` (optional)
- `token` (optional)
- `pipeline_events` (optional)

Available keys for `kwargs` are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, *\*\*kwargs*)

Delete the object with ID *id*.

Available keys for `kwargs` are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of `ProjectHook`

**class** `gitlab.v3.objects.ProjectIssue` (*gl*, *data=None*, *\*\*kwargs*)

Bases: `gitlab.base.GitlabObject`

**notes**

ProjectIssueNoteManager - Manager for objects.

**save** (*\*\*kwargs*)

Send the modified object to the GitLab server. The following attributes are sent:

- `title` (optional)
- `description` (optional)
- `assignee_id` (optional)
- `milestone_id` (optional)
- `labels` (optional)
- `created_at` (optional)
- `updated_at` (optional)
- `state_event` (optional)
- `due_date` (optional)

Available keys for `kwargs` are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**add\_spent\_time** (*\*\*kwargs*)

Set an estimated time of work for the issue.

**Raises** `GitlabConnectionError` – If the server cannot be reached.

**description\_attr** = `'description'`

**managers** = (`(('notes', 'ProjectIssueNoteManager', [(('project_id', 'project_id'), ('issue_id', 'id'))]),)`

**move** (*to\_project\_id*, *\*\*kwargs*)

Move the issue to another project.

**Raises** `GitlabConnectionError` – If the server cannot be reached.

**optionalCreateAttrs** = [`'description'`, `'assignee_id'`, `'milestone_id'`, `'labels'`, `'created_at'`, `'due_date'`]

**optionalListAttrs** = [`'state'`, `'labels'`, `'milestone'`, `'iid'`, `'order_by'`, `'sort'`]

`optionalUpdateAttrs = ['title', 'description', 'assignee_id', 'milestone_id', 'labels', 'created_at', 'updated_at', 'state']`

`project_id_attr = 'project_id'`

`requiredCreateAttrs = ['title']`

`requiredUrlAttrs = ['project_id']`

`reset_spent_time (**kwargs)`

Set an estimated time of work for the issue.

**Raises** `GitlabConnectionError` – If the server cannot be reached.

`reset_time_estimate (**kwargs)`

Resets estimated time for the issue to 0 seconds.

**Raises** `GitlabConnectionError` – If the server cannot be reached.

`shortPrintAttr = 'title'`

`subscribe (**kwargs)`

Subscribe to an issue.

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabSubscribeError` – If the subscription cannot be done

`time_estimate (**kwargs)`

Set an estimated time of work for the issue.

**Raises** `GitlabConnectionError` – If the server cannot be reached.

`time_stats (**kwargs)`

Get time stats for the issue.

**Raises** `GitlabConnectionError` – If the server cannot be reached.

`todo (**kwargs)`

Create a todo for the issue.

**Raises** `GitlabConnectionError` – If the server cannot be reached.

`unsubscribe (**kwargs)`

Unsubscribe an issue.

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabUnsubscribeError` – If the unsubscription cannot be done

**class** `gitlab.v3.objects.ProjectIssueManager` (*gl*, *parent=None*, *args=[]*)

Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.ProjectIssue'> objects.

<class 'gitlab.v3.objects.ProjectIssue'> objects can be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectIssue'>.

Available keys for *kwargs* are:

- *state* (optional)
- *labels* (optional)

- milestone (optional)
- iid (optional)
- order\_by (optional)
- sort (optional)
- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (*id*, **\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.ProjectIssue'> using its *id*.

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**create** (*data*, **\*\*kwargs**)

Create an object of type <class 'gitlab.v3.objects.ProjectIssue'>.

*data* is a dict defining the object attributes. Available attributes are:

- project\_id (required if not discovered on the parent objects)
- title (required)
- description (optional)
- assignee\_id (optional)
- milestone\_id (optional)
- labels (optional)
- created\_at (optional)
- due\_date (optional)

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, **\*\*kwargs**)

Delete the object with ID *id*.

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectIssue*

**class** gitlab.v3.objects.**ProjectIssueNote** (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**save** (**\*\*kwargs**)

Send the modified object to the GitLab server. The following attributes are sent:

- body (required)
- created\_at (optional)

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**canDelete = False**

**description\_attr = 'body'**

**optionalCreateAttrs = ['created\_at']**

**project\_id\_attr = 'project\_id'**

**requiredCreateAttrs = ['body']**

**requiredUrlAttrs = ['project\_id', 'issue\_id']**

**class** gitlab.v3.objects.**ProjectIssueNoteManager** (*gl*, *parent=None*, *args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectIssueNote'> objects.

<class 'gitlab.v3.objects.ProjectIssueNote'> objects can be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectIssueNote'>.

Available keys for *kwargs* are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.ProjectIssueNote'> using its *id*.

Available keys for *kwargs* are:

- sudo (string or int): run the request as another user (requires admin permissions)

**create** (*data*, *\*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.ProjectIssueNote'>.

*data* is a dict defining the object attributes. Available attributes are:

- project\_id (required if not discovered on the parent objects)
- issue\_id (required if not discovered on the parent objects)
- body (required)
- created\_at (optional)

Available keys for *kwargs* are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectIssueNote*

**class** gitlab.v3.objects.**ProjectKey** (*gl*, *data=None*, *\*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**canUpdate = False**

**requiredCreateAttrs = ['title', 'key']**

**requiredUrlAttrs = ['project\_id']**



---

**class** `gitlab.v3.objects.ProjectKeyManager` (*gl*, *parent=None*, *args=[]*)  
 Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.ProjectKey'> objects.

<class 'gitlab.v3.objects.ProjectKey'> objects **cannot** be updated.

**list** (*\*\*kwargs*)  
 Returns a list of objects of type <class 'gitlab.v3.objects.ProjectKey'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)  
 Get a single object of type <class 'gitlab.v3.objects.ProjectKey'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data*, *\*\*kwargs*)  
 Create an object of type <class 'gitlab.v3.objects.ProjectKey'>.

*data* is a dict defining the object attributes. Available attributes are:

- *project\_id* (required if not discovered on the parent objects)
- *title* (required)
- *key* (required)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, *\*\*kwargs*)  
 Delete the object with ID *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**disable** (*key\_id*)  
 Disable a deploy key for a project.

**enable** (*key\_id*)  
 Enable a deploy key for a project.

**obj\_cls**  
 alias of `ProjectKey`

**class** `gitlab.v3.objects.ProjectLabel` (*gl*, *data=None*, *\*\*kwargs*)  
 Bases: `gitlab.base.GitlabObject`

**save** (*\*\*kwargs*)  
 Send the modified object to the GitLab server. The following attributes are sent:

- *name* (required)
- *new\_name* (optional)

- color (optional)
- description (optional)
- priority (optional)

Available keys for `kwargs` are:

- sudo (string or int): run the request as another user (requires admin permissions)

**canGet** = 'from\_list'

**idAttr** = 'name'

**optionalCreateAttrs** = ['description', 'priority']

**optionalUpdateAttrs** = ['new\_name', 'color', 'description', 'priority']

**requiredCreateAttrs** = ['name', 'color']

**requiredDeleteAttrs** = ['name']

**requiredUpdateAttrs** = ['name']

**requiredUrlAttrs** = ['project\_id']

**subscribe** (\*\*kwargs)

Subscribe to a label.

#### Raises

- GitlabConnectionError – If the server cannot be reached.
- GitlabSubscribeError – If the subscription cannot be done

**unsubscribe** (\*\*kwargs)

Unsubscribe a label.

#### Raises

- GitlabConnectionError – If the server cannot be reached.
- GitlabUnsubscribeError – If the unsubscription cannot be done

**class** gitlab.v3.objects.**ProjectLabelManager** (gl, parent=None, args=[])

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectLabel'> objects.

<class 'gitlab.v3.objects.ProjectLabel'> objects can be updated.

**list** (\*\*kwargs)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectLabel'>.

Available keys for `kwargs` are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (id, \*\*kwargs)

Get a single object of type <class 'gitlab.v3.objects.ProjectLabel'> using its `id`.

Available keys for `kwargs` are:

- sudo (string or int): run the request as another user (requires admin permissions)

**create** (*data*, **\*\*kwargs**)

Create an object of type <class 'gitlab.v3.objects.ProjectLabel'>.

*data* is a dict defining the object attributes. Available attributes are:

- *project\_id* (required if not discovered on the parent objects)
- *name* (required)
- *color* (required)
- *description* (optional)
- *priority* (optional)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, **\*\*kwargs**)

Delete the object with ID *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectLabel*

**class** `gitlab.v3.objects.ProjectManager` (*gl*, *parent=None*, *args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.Project'> objects.

<class 'gitlab.v3.objects.Project'> objects can be updated.

**list** (**\*\*kwargs**)

Returns a list of objects of type <class 'gitlab.v3.objects.Project'>.

Available keys for *kwargs* are:

- *search* (optional)
- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, **\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.Project'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data*, **\*\*kwargs**)

Create an object of type <class 'gitlab.v3.objects.Project'>.

*data* is a dict defining the object attributes. Available attributes are:

- *name* (required)
- *path* (optional)
- *namespace\_id* (optional)

- description (optional)
- issues\_enabled (optional)
- merge\_requests\_enabled (optional)
- builds\_enabled (optional)
- wiki\_enabled (optional)
- snippets\_enabled (optional)
- container\_registry\_enabled (optional)
- shared\_runners\_enabled (optional)
- public (optional)
- visibility\_level (optional)
- import\_url (optional)
- public\_builds (optional)
- only\_allow\_merge\_if\_build\_succeeds (optional)
- only\_allow\_merge\_if\_all\_discussions\_are\_resolved (optional)
- lfs\_enabled (optional)
- request\_access\_enabled (optional)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, **\*\*kwargs**)

Delete the object with ID *id*.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**all** (**\*\*kwargs**)

List all the projects (need admin rights).

**Parameters**

- **all** (*bool*) – If True, return all the items, without pagination
- **\*\*kwargs** – Additional arguments to send to GitLab.

**Returns** The list of projects.

**Return type** *list*(gitlab.Gitlab.Project)

**obj\_cls**

alias of *Project*

**owned** (**\*\*kwargs**)

List owned projects.

**Parameters**

- **all** (*bool*) – If True, return all the items, without pagination
- **\*\*kwargs** – Additional arguments to send to GitLab.

**Returns** The list of owned projects.

**Return type** *list*(gitlab.Gitlab.Project)

**search** (*query*, *\*\*kwargs*)

Search projects by name.

API v3 only.

**Note:** The search is only performed on the project name (not on the namespace or the description). To perform a smarter search, use the `search` argument of the `list()` method:

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

### Parameters

- **query** (*str*) – The query string to send to GitLab for the search.
- **all** (*bool*) – If True, return all the items, without pagination
- **\*\*kwargs** – Additional arguments to send to GitLab.

**Returns** A list of matching projects.

**Return type** *list*(gitlab.Gitlab.Project)

**starred** (*\*\*kwargs*)

List starred projects.

### Parameters

- **all** (*bool*) – If True, return all the items, without pagination
- **\*\*kwargs** – Additional arguments to send to GitLab.

**Returns** The list of starred projects.

**Return type** *list*(gitlab.Gitlab.Project)

**class** `gitlab.v3.objects.ProjectMember` (*gl*, *data=None*, *\*\*kwargs*)

Bases: `gitlab.base.GitlabObject`

**save** (*\*\*kwargs*)

Send the modified object to the GitLab server. The following attributes are sent:

- `access_level` (required)

Available keys for `kwargs` are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**optionalCreateAttrs** = ['expires\_at']

**requiredCreateAttrs** = ['access\_level', 'user\_id']

**requiredUpdateAttrs** = ['access\_level']

**requiredUrlAttrs** = ['project\_id']

**shortPrintAttr** = 'username'

**class** `gitlab.v3.objects.ProjectMemberManager` (*gl*, *parent=None*, *args=[]*)

Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.ProjectMember'> objects.

<class 'gitlab.v3.objects.ProjectMember'> objects can be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectMember'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.ProjectMember'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data*, *\*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.ProjectMember'>.

*data* is a dict defining the object attributes. Available attributes are:

- *project\_id* (required if not discovered on the parent objects)
- *access\_level* (required)
- *user\_id* (required)
- *expires\_at* (optional)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, *\*\*kwargs*)

Delete the object with ID *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectMember*

**class** `gitlab.v3.objects.ProjectMergeRequest` (*gl*, *data=None*, *\*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**notes**

*ProjectMergeRequestNoteManager* - Manager for objects.

**diffs**

*ProjectMergeRequestDiffManager* - Manager for objects.

**save** (*\*\*kwargs*)

Send the modified object to the GitLab server. The following attributes are sent:

- *target\_branch* (optional)
- *assignee\_id* (optional)
- *title* (optional)
- *description* (optional)
- *state\_event* (optional)

- `labels` (optional)
- `milestone_id` (optional)

Available keys for `kwargs` are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**add\_spent\_time** (*\*\*kwargs*)

Set an estimated time of work for the merge request.

**Raises** `GitlabConnectionError` – If the server cannot be reached.

**cancel\_merge\_when\_build\_succeeds** (*\*\*kwargs*)

Cancel merge when build succeeds.

**changes** (*\*\*kwargs*)

List the merge request changes.

**Returns** List of changes

**Return type** *list* (dict)

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabListError` – If the server fails to perform the request.

**closes\_issues** (*\*\*kwargs*)

List issues closed by the MR.

**Returns** List of closed issues

**Return type** *list* (`ProjectIssue`)

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabGetError` – If the server fails to perform the request.

**commits** (*\*\*kwargs*)

List the merge request commits.

**Returns** List of commits

**Return type** *list* (`ProjectCommit`)

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabListError` – If the server fails to perform the request.

**managers** = (('notes', 'ProjectMergeRequestNoteManager', [(('project\_id', 'project\_id'), ('merge\_request\_id', 'id'))], ('d

**merge** (*merge\_commit\_message=None, should\_remove\_source\_branch=False, merge\_when\_build\_succeeds=False, \*\*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\_build\_succeeds** (*bool*) – Wait for the build to succeed, then merge

**Returns** The updated MR

**Return type** *ProjectMergeRequest*

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabMRForbiddenError` – If the user doesn't have permission to close the MR
- `GitlabMRClosedError` – If the MR is already closed

**optionalCreateAttrs** = ['assignee\_id', 'description', 'target\_project\_id', 'labels', 'milestone\_id', 'remove\_source\_br

**optionalListAttrs** = ['iid', 'state', 'order\_by', 'sort']

**optionalUpdateAttrs** = ['target\_branch', 'assignee\_id', 'title', 'description', 'state\_event', 'labels', 'milestone\_id']

**requiredCreateAttrs** = ['source\_branch', 'target\_branch', 'title']

**requiredUrlAttrs** = ['project\_id']

**reset\_spent\_time** (\*\*kwargs)

Set an estimated time of work for the merge request.

**Raises** `GitlabConnectionError` – If the server cannot be reached.

**reset\_time\_estimate** (\*\*kwargs)

Resets estimated time for the merge request to 0 seconds.

**Raises** `GitlabConnectionError` – If the server cannot be reached.

**subscribe** (\*\*kwargs)

Subscribe to a MR.

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabSubscribeError` – If the subscription cannot be done

**time\_estimate** (\*\*kwargs)

Set an estimated time of work for the merge request.

**Raises** `GitlabConnectionError` – If the server cannot be reached.

**time\_stats** (\*\*kwargs)

Get time stats for the merge request.

**Raises** `GitlabConnectionError` – If the server cannot be reached.

**todo** (\*\*kwargs)

Create a todo for the merge request.

**Raises** `GitlabConnectionError` – If the server cannot be reached.

**unsubscribe** (\*\*kwargs)

Unsubscribe a MR.

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabUnsubscribeError` – If the unsubscription cannot be done

**class** `gitlab.v3.objects.ProjectMergeRequestDiff` (*gl*, *data=None*, \*\*kwargs)

Bases: `gitlab.base.GitlabObject`

**canCreate** = False

**canDelete** = False



**canUpdate = False**

**requiredUrlAttrs = ['project\_id', 'merge\_request\_id']**

**class** `gitlab.v3.objects.ProjectMergeRequestDiffManager` (*gl*, *parent=None*, *args=[]*)  
 Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.ProjectMergeRequestDiff'> objects.

<class 'gitlab.v3.objects.ProjectMergeRequestDiff'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectMergeRequestDiff'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.ProjectMergeRequestDiff'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of `ProjectMergeRequestDiff`

**class** `gitlab.v3.objects.ProjectMergeRequestManager` (*gl*, *parent=None*, *args=[]*)  
 Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.ProjectMergeRequest'> objects.

<class 'gitlab.v3.objects.ProjectMergeRequest'> objects can be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectMergeRequest'>.

Available keys for *kwargs* are:

- *iid* (optional)
- *state* (optional)
- *order\_by* (optional)
- *sort* (optional)
- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.ProjectMergeRequest'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data*, *\*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.ProjectMergeRequest'>.

*data* is a dict defining the object attributes. Available attributes are:

- *project\_id* (required if not discovered on the parent objects)
- *source\_branch* (required)
- *target\_branch* (required)
- *title* (required)
- *assignee\_id* (optional)
- *description* (optional)
- *target\_project\_id* (optional)
- *labels* (optional)
- *milestone\_id* (optional)
- *remove\_source\_branch* (optional)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, *\*\*kwargs*)

Delete the object with ID *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectMergeRequest*

**class** `gitlab.v3.objects.ProjectMergeRequestNote` (*gl*, *data=None*, *\*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**save** (*\*\*kwargs*)

Send the modified object to the GitLab server. The following attributes are sent:

- *body* (required)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**requiredCreateAttrs** = ['body']

**requiredUrlAttrs** = ['project\_id', 'merge\_request\_id']

**class** `gitlab.v3.objects.ProjectMergeRequestNoteManager` (*gl*, *parent=None*, *args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectMergeRequestNote'> objects.

<class 'gitlab.v3.objects.ProjectMergeRequestNote'> objects can be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectMergeRequestNote'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.

- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (*id*, **\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.ProjectMergeRequestNote'> using its *id*.

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**create** (*data*, **\*\*kwargs**)

Create an object of type <class 'gitlab.v3.objects.ProjectMergeRequestNote'>.

*data* is a dict defining the object attributes. Available attributes are:

- project\_id (required if not discovered on the parent objects)
- merge\_request\_id (required if not discovered on the parent objects)
- body (required)

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, **\*\*kwargs**)

Delete the object with ID *id*.

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectMergeRequestNote*

**class** gitlab.v3.objects.**ProjectMilestone** (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**save** (**\*\*kwargs**)

Send the modified object to the GitLab server. The following attributes are sent:

- title (optional)
- description (optional)
- due\_date (optional)
- start\_date (optional)
- state\_event (optional)

Available keys for **kwargs** are:

- sudo (string or int): run the request as another user (requires admin permissions)

**canDelete = False**

**issues** (**\*\*kwargs**)

**merge\_requests** (**\*\*kwargs**)

List the merge requests related to this milestone

**Returns** List of merge requests

**Return type** *list* (*ProjectMergeRequest*)

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabListError` – If the server fails to perform the request.

**optionalCreateAttrs** = ['description', 'due\_date', 'start\_date', 'state\_event']

**optionalListAttrs** = ['iid', 'state']

**optionalUpdateAttrs** = ['title', 'description', 'due\_date', 'start\_date', 'state\_event']

**requiredCreateAttrs** = ['title']

**requiredUrlAttrs** = ['project\_id']

**shortPrintAttr** = 'title'

**class** `gitlab.v3.objects.ProjectMilestoneManager` (*gl*, *parent=None*, *args=[]*)

Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.ProjectMilestone'> objects.

<class 'gitlab.v3.objects.ProjectMilestone'> objects can be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectMilestone'>.

Available keys for *kwargs* are:

- *iid* (optional)
- *state* (optional)
- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.ProjectMilestone'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data*, *\*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.ProjectMilestone'>.

*data* is a dict defining the object attributes. Available attributes are:

- *project\_id* (required if not discovered on the parent objects)
- *title* (required)
- *description* (optional)
- *due\_date* (optional)
- *start\_date* (optional)
- *state\_event* (optional)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**  
alias of *ProjectMilestone*

**class** gitlab.v3.objects.**ProjectNote** (*gl, data=None, \*\*kwargs*)  
Bases: *gitlab.base.GitlabObject*

**canDelete = False**

**canUpdate = False**

**requiredCreateAttrs = ['body']**

**requiredUrlAttrs = ['project\_id']**

**class** gitlab.v3.objects.**ProjectNoteManager** (*gl, parent=None, args=[]*)  
Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectNote'> objects.

<class 'gitlab.v3.objects.ProjectNote'> objects **cannot** be updated.

**list** (*\*\*kwargs*)  
Returns a list of objects of type <class 'gitlab.v3.objects.ProjectNote'>.

Available keys for kwargs are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id, \*\*kwargs*)  
Get a single object of type <class 'gitlab.v3.objects.ProjectNote'> using its *id*.

Available keys for kwargs are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data, \*\*kwargs*)  
Create an object of type <class 'gitlab.v3.objects.ProjectNote'>.

*data* is a dict defining the object attributes. Available attributes are:

- *project\_id* (required if not discovered on the parent objects)
- *body* (required)

Available keys for kwargs are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**  
alias of *ProjectNote*

**class** gitlab.v3.objects.**ProjectNotificationSettings** (*gl, data=None, \*\*kwargs*)  
Bases: *gitlab.v3.objects.NotificationSettings*

**save** (*\*\*kwargs*)  
Send the modified object to the GitLab server. The following attributes are sent:

- *level* (optional)
- *notification\_email* (optional)
- *new\_note* (optional)

- new\_issue (optional)
- reopen\_issue (optional)
- close\_issue (optional)
- reassign\_issue (optional)
- new\_merge\_request (optional)
- reopen\_merge\_request (optional)
- close\_merge\_request (optional)
- reassign\_merge\_request (optional)
- merge\_merge\_request (optional)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**requiredUrlAttrs = ['project\_id']**

**class** gitlab.v3.objects.**ProjectNotificationSettingsManager** (*gl*, *parent=None*,  
*args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectNotificationSettings'> objects.

<class 'gitlab.v3.objects.ProjectNotificationSettings'> objects can be updated.

**get** (*\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.ProjectNotificationSettings'>.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectNotificationSettings*

**class** gitlab.v3.objects.**ProjectPipeline** (*gl*, *data=None*, *\*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**canDelete = False**

**canUpdate = False**

**cancel** (*\*\*kwargs*)

Cancel builds in a pipeline.

**Raises**

- *GitlabConnectionError* – If the server cannot be reached.
- *GitlabPipelineCancelError* – If the retry cannot be done.

**requiredCreateAttrs = ['ref']**

**requiredUrlAttrs = ['project\_id']**

**retry** (*\*\*kwargs*)

Retries failed builds in a pipeline.

**Raises**

- *GitlabConnectionError* – If the server cannot be reached.
- *GitlabPipelineRetryError* – If the retry cannot be done.

---

**class** `gitlab.v3.objects.ProjectPipelineManager` (*gl*, *parent=None*, *args=[]*)  
 Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.ProjectPipeline'> objects.

<class 'gitlab.v3.objects.ProjectPipeline'> objects **cannot** be updated.

**list** (*\*\*kwargs*)  
 Returns a list of objects of type <class 'gitlab.v3.objects.ProjectPipeline'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)  
 Get a single object of type <class 'gitlab.v3.objects.ProjectPipeline'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data*, *\*\*kwargs*)  
 Create an object of type <class 'gitlab.v3.objects.ProjectPipeline'>.

*data* is a dict defining the object attributes. Available attributes are:

- *project\_id* (required if not discovered on the parent objects)
- *ref* (required)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**  
 alias of `ProjectPipeline`

**class** `gitlab.v3.objects.ProjectRunner` (*gl*, *data=None*, *\*\*kwargs*)  
 Bases: `gitlab.base.GitlabObject`

**canUpdate = False**

**requiredCreateAttrs = ['runner\_id']**

**class** `gitlab.v3.objects.ProjectRunnerManager` (*gl*, *parent=None*, *args=[]*)  
 Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.ProjectRunner'> objects.

<class 'gitlab.v3.objects.ProjectRunner'> objects **cannot** be updated.

**list** (*\*\*kwargs*)  
 Returns a list of objects of type <class 'gitlab.v3.objects.ProjectRunner'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, **\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.ProjectRunner'> using its *id*.

Available keys for **kwargs** are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**create** (*data*, **\*\*kwargs**)

Create an object of type <class 'gitlab.v3.objects.ProjectRunner'>.

*data* is a dict defining the object attributes. Available attributes are:

- **runner\_id** (required)

Available keys for **kwargs** are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, **\*\*kwargs**)

Delete the object with ID *id*.

Available keys for **kwargs** are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectRunner*

**class** gitlab.v3.objects.**ProjectService** (*gl*, *data=None*, **\*\*kwargs**)

Bases: *gitlab.base.GitlabObject*

**save** (**\*\*kwargs**)

Send the modified object to the GitLab server. The following attributes are sent:

Available keys for **kwargs** are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**canCreate = False**

**canList = False**

**getRequiresId = False**

**requiredUrlAttrs = ['project\_id', 'service\_name']**

**class** gitlab.v3.objects.**ProjectServiceManager** (*gl*, *parent=None*, *args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectService'> objects.

<class 'gitlab.v3.objects.ProjectService'> objects can be updated.

**get** (**\*\*kwargs**)

Get a single object of type <class 'gitlab.v3.objects.ProjectService'>.

Available keys for **kwargs** are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, **\*\*kwargs**)

Delete the object with ID *id*.

Available keys for **kwargs** are:

- **sudo** (string or int): run the request as another user (requires admin permissions)



**available** (\*\*kwargs)

List the services known by python-gitlab.

**Returns** The list of service code names.

**Return type** *list* (str)

**obj\_cls**

alias of *ProjectService*

**class** gitlab.v3.objects.**ProjectSnippet** (gl, data=None, \*\*kwargs)

Bases: *gitlab.base.GitlabObject*

**notes**

ProjectSnippetNoteManager - Manager for objects.

**save** (\*\*kwargs)

Send the modified object to the GitLab server. The following attributes are sent:

- **title** (optional)
- **file\_name** (optional)
- **code** (optional)
- **visibility\_level** (optional)

Available keys for kwargs are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**content** (streamed=False, action=None, chunk\_size=1024, \*\*kwargs)

Return the raw 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.

**Returns** The snippet content

**Return type** str

**Raises**

- *GitlabConnectionError* – If the server cannot be reached.
- *GitlabGetError* – If the server fails to perform the request.

**managers** = (('notes', 'ProjectSnippetNoteManager', [(('project\_id', 'project\_id'), ('snippet\_id', 'id'))],)

**optionalCreateAttrs** = ['lifetime', 'visibility\_level']

**optionalUpdateAttrs** = ['title', 'file\_name', 'code', 'visibility\_level']

**requiredCreateAttrs** = ['title', 'file\_name', 'code']

**requiredUrlAttrs** = ['project\_id']

**shortPrintAttr** = 'title'

**class** gitlab.v3.objects.**ProjectSnippetManager** (gl, parent=None, args=[])

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectSnippet'> objects.

<class 'gitlab.v3.objects.ProjectSnippet'> objects can be updated.

**list** (\*\*kwargs)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectSnippet'>.

Available keys for kwargs are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (id, \*\*kwargs)

Get a single object of type <class 'gitlab.v3.objects.ProjectSnippet'> using its id.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**create** (data, \*\*kwargs)

Create an object of type <class 'gitlab.v3.objects.ProjectSnippet'>.

data is a dict defining the object attributes. Available attributes are:

- project\_id (required if not discovered on the parent objects)
- title (required)
- file\_name (required)
- code (required)
- lifetime (optional)
- visibility\_level (optional)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**delete** (id, \*\*kwargs)

Delete the object with ID id.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectSnippet*

**class** gitlab.v3.objects.**ProjectSnippetNote** (gl, data=None, \*\*kwargs)

Bases: *gitlab.base.GitlabObject*

**canDelete** = False

**canUpdate** = False

**requiredCreateAttrs** = ['body']

**requiredUrlAttrs** = ['project\_id', 'snippet\_id']

**class** gitlab.v3.objects.**ProjectSnippetNoteManager** (gl, parent=None, args=[])

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectSnippetNote'> objects.

<class 'gitlab.v3.objects.ProjectSnippetNote'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectSnippetNote'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.ProjectSnippetNote'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data*, *\*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.ProjectSnippetNote'>.

*data* is a dict defining the object attributes. Available attributes are:

- *project\_id* (required if not discovered on the parent objects)
- *snippet\_id* (required if not discovered on the parent objects)
- *body* (required)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectSnippetNote*

**class** `gitlab.v3.objects.ProjectTag` (*gl*, *data=None*, *\*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**canGet** = 'from\_list'

**canUpdate** = False

**idAttr** = 'name'

**optionalCreateAttrs** = ['message']

**requiredCreateAttrs** = ['tag\_name', 'ref']

**requiredUrlAttrs** = ['project\_id']

**set\_release\_description** (*description*)

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.

**Raises**

- *GitlabConnectionError* – If the server cannot be reached.
- *GitlabCreateError* – If the server fails to create the release.
- *GitlabUpdateError* – If the server fails to update the release.

**shortPrintAttr** = 'name'

**class** gitlab.v3.objects.**ProjectTagManager** (*gl*, *parent=None*, *args=[]*)

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectTag'> objects.

<class 'gitlab.v3.objects.ProjectTag'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectTag'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.ProjectTag'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data*, *\*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.ProjectTag'>.

*data* is a dict defining the object attributes. Available attributes are:

- *project\_id* (required if not discovered on the parent objects)
- *tag\_name* (required)
- *ref* (required)
- *message* (optional)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, *\*\*kwargs*)

Delete the object with ID *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectTag*

**class** gitlab.v3.objects.**ProjectTagRelease** (*gl*, *data=None*, *\*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**save** (*\*\*kwargs*)

Send the modified object to the GitLab server. The following attributes are sent:

- *description* (required)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**canDelete** = False

**canList = False**

**requiredCreateAttrs = ['description']**

**requiredUrlAttrs = ['project\_id', 'tag\_name']**

**shortPrintAttr = 'description'**

**class** gitlab.v3.objects.**ProjectTrigger** (*gl, data=None, \*\*kwargs*)  
Bases: *gitlab.base.GitlabObject*

**canUpdate = False**

**idAttr = 'token'**

**requiredUrlAttrs = ['project\_id']**

**class** gitlab.v3.objects.**ProjectTriggerManager** (*gl, parent=None, args=[]*)  
Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectTrigger'> objects.

<class 'gitlab.v3.objects.ProjectTrigger'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectTrigger'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id, \*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.ProjectTrigger'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data, \*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.ProjectTrigger'>.

*data* is a dict defining the object attributes. Available attributes are:

- *project\_id* (required if not discovered on the parent objects)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**delete** (*id, \*\*kwargs*)

Delete the object with ID *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectTrigger*

**class** gitlab.v3.objects.**ProjectVariable** (*gl, data=None, \*\*kwargs*)  
Bases: *gitlab.base.GitlabObject*

**save** (\*\*kwargs)

Send the modified object to the GitLab server. The following attributes are sent:

- key (required)
- value (required)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**idAttr** = 'key'

**requiredCreateAttrs** = ['key', 'value']

**requiredUrlAttrs** = ['project\_id']

**class** gitlab.v3.objects.**ProjectVariableManager** (gl, parent=None, args=[])

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.ProjectVariable'> objects.

<class 'gitlab.v3.objects.ProjectVariable'> objects can be updated.

**list** (\*\*kwargs)

Returns a list of objects of type <class 'gitlab.v3.objects.ProjectVariable'>.

Available keys for kwargs are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (id, \*\*kwargs)

Get a single object of type <class 'gitlab.v3.objects.ProjectVariable'> using its id.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**create** (data, \*\*kwargs)

Create an object of type <class 'gitlab.v3.objects.ProjectVariable'>.

data is a dict defining the object attributes. Available attributes are:

- project\_id (required if not discovered on the parent objects)
- key (required)
- value (required)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**delete** (id, \*\*kwargs)

Delete the object with ID id.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *ProjectVariable*

**class** `gitlab.v3.objects.Runner` (*gl*, *data=None*, *\*\*kwargs*)

Bases: `gitlab.base.GitlabObject`

**save** (*\*\*kwargs*)

Send the modified object to the GitLab server. The following attributes are sent:

- `description` (optional)
- `active` (optional)
- `tag_list` (optional)

Available keys for `kwargs` are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**canCreate = False**

**optionalListAttrs = ['scope']**

**optionalUpdateAttrs = ['description', 'active', 'tag\_list']**

**class** `gitlab.v3.objects.RunnerManager` (*gl*, *parent=None*, *args=[]*)

Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.Runner'> objects.

<class 'gitlab.v3.objects.Runner'> objects can be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.Runner'>.

Available keys for `kwargs` are:

- `scope` (optional)
- `per_page` (int): number of item per page. May be limited by the server.
- `page` (int): page to retrieve
- `all` (bool): iterate over all the pages and return all the entries
- `sudo` (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.Runner'> using its `id`.

Available keys for `kwargs` are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, *\*\*kwargs*)

Delete the object with ID `id`.

Available keys for `kwargs` are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**all** (*scope=None*, *\*\*kwargs*)

List all the runners.

**Parameters** `scope` (*str*) – The scope of runners to show, one of: specific, shared, active, paused, online

**Returns** a list of runners matching the scope.

**Return type** `list(Runner)`

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabListError` – If the resource cannot be found

**obj\_cls**

alias of `Runner`

**class** `gitlab.v3.objects.SidekiqManager` (*gl*)

Bases: `object`

Manager for the Sidekiq methods.

This manager doesn't actually manage objects but provides helper function for the sidekiq metrics API.

**compound\_metrics** (*\*\*kwargs*)

Returns all available metrics and statistics.

**job\_stats** (*\*\*kwargs*)

Returns statistics about the jobs performed.

**process\_metrics** (*\*\*kwargs*)

Returns the registred sidekiq workers.

**queue\_metrics** (*\*\*kwargs*)

Returns the registred queues information.

**class** `gitlab.v3.objects.Snippet` (*gl*, *data=None*, *\*\*kwargs*)

Bases: `gitlab.base.GitlabObject`

**save** (*\*\*kwargs*)

Send the modified object to the GitLab server. The following attributes are sent:

- `title` (optional)
- `file_name` (optional)
- `content` (optional)
- `visibility_level` (optional)

Available keys for `kwargs` are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**optionalCreateAttrs** = ['lifetime', 'visibility\_level']

**optionalUpdateAttrs** = ['title', 'file\_name', 'content', 'visibility\_level']

**raw** (*streamed=False*, *action=None*, *chunk\_size=1024*, *\*\*kwargs*)

Return the raw 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.

**Returns** The snippet content.

**Return type** str

#### Raises

- `GitlabConnectionError` – If the server cannot be reached.



- `GitlabGetError` – If the server fails to perform the request.

**requiredCreateAttrs** = ['title', 'file\_name', 'content']

**shortPrintAttr** = 'title'

**class** `gitlab.v3.objects.SnippetManager` (*gl*, *parent=None*, *args=[]*)

Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.Snippet'> objects.

<class 'gitlab.v3.objects.Snippet'> objects can be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.Snippet'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id*, *\*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.Snippet'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data*, *\*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.Snippet'>.

*data* is a dict defining the object attributes. Available attributes are:

- *title* (required)
- *file\_name* (required)
- *content* (required)
- *lifetime* (optional)
- *visibility\_level* (optional)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, *\*\*kwargs*)

Delete the object with ID *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of `Snippet`

**public** (*\*\*kwargs*)

List all the public snippets.

#### Parameters

- **all** (*bool*) – If True, return all the items, without pagination

- **\*\*kwargs** – Additional arguments to send to GitLab.

**Returns** The list of snippets.

**Return type** *list*(gitlab.Gitlab.Snippet)

**class** gitlab.v3.objects.**Team**(gl, data=None, \*\*kwargs)

Bases: *gitlab.base.GitlabObject*

**canUpdate** = False

**managers** = (('members', 'TeamMemberManager', [('team\_id', 'id')]), ('projects', 'TeamProjectManager', [('team\_id',

requiredCreateAttrs = ['name', 'path']

shortPrintAttr = 'name'

**class** gitlab.v3.objects.**TeamManager**(gl, parent=None, args=[])

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.Team'> objects.

<class 'gitlab.v3.objects.Team'> objects **cannot** be updated.

**list** (\*\*kwargs)

Returns a list of objects of type <class 'gitlab.v3.objects.Team'>.

Available keys for kwargs are:

- **per\_page** (int): number of item per page. May be limited by the server.
- **page** (int): page to retrieve
- **all** (bool): iterate over all the pages and return all the entries
- **sudo** (string or int): run the request as another user (requires admin permissions)

**get** (id, \*\*kwargs)

Get a single object of type <class 'gitlab.v3.objects.Team'> using its id.

Available keys for kwargs are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**create** (data, \*\*kwargs)

Create an object of type <class 'gitlab.v3.objects.Team'>.

data is a dict defining the object attributes. Available attributes are:

- **name** (required)
- **path** (required)

Available keys for kwargs are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**delete** (id, \*\*kwargs)

Delete the object with ID id.

Available keys for kwargs are:

- **sudo** (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *Team*

**class** gitlab.v3.objects.**TeamMember**(gl, data=None, \*\*kwargs)

Bases: *gitlab.base.GitlabObject*

**canUpdate = False**

**requiredCreateAttrs = ['access\_level']**

**requiredUrlAttrs = ['teamd\_id']**

**shortPrintAttr = 'username'**

**class** gitlab.v3.objects.**TeamMemberManager** (*gl, parent=None, args=[]*)  
 Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.TeamMember'> objects.

<class 'gitlab.v3.objects.TeamMember'> objects **cannot** be updated.

**list** (*\*\*kwargs*)

Returns a list of objects of type <class 'gitlab.v3.objects.TeamMember'>.

Available keys for *kwargs* are:

- *per\_page* (int): number of item per page. May be limited by the server.
- *page* (int): page to retrieve
- *all* (bool): iterate over all the pages and return all the entries
- *sudo* (string or int): run the request as another user (requires admin permissions)

**get** (*id, \*\*kwargs*)

Get a single object of type <class 'gitlab.v3.objects.TeamMember'> using its *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**create** (*data, \*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.TeamMember'>.

*data* is a dict defining the object attributes. Available attributes are:

- *teamd\_id* (required if not discovered on the parent objects)
- *access\_level* (required)

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**delete** (*id, \*\*kwargs*)

Delete the object with ID *id*.

Available keys for *kwargs* are:

- *sudo* (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *TeamMember*

**class** gitlab.v3.objects.**TeamProject** (*gl, data=None, \*\*kwargs*)  
 Bases: *gitlab.base.GitlabObject*

**canUpdate = False**

**requiredCreateAttrs = ['greatest\_access\_level']**

**requiredUrlAttrs = ['team\_id']**

**shortPrintAttr = 'name'**

```
class gitlab.v3.objects.TeamProjectManager (gl, parent=None, args=[])
    Bases: gitlab.base.BaseManager

    Manager for <class 'gitlab.v3.objects.TeamProject'> objects.

    <class 'gitlab.v3.objects.TeamProject'> objects cannot be updated.

    list (**kwargs)
        Returns a list of objects of type <class 'gitlab.v3.objects.TeamProject'>.

        Available keys for kwargs are:

        • per_page (int): number of item per page. May be limited by the server.

        • page (int): page to retrieve

        • all (bool): iterate over all the pages and return all the entries

        • sudo (string or int): run the request as another user (requires admin permissions)

    get (id, **kwargs)
        Get a single object of type <class 'gitlab.v3.objects.TeamProject'> using its id.

        Available keys for kwargs are:

        • sudo (string or int): run the request as another user (requires admin permissions)

    create (data, **kwargs)
        Create an object of type <class 'gitlab.v3.objects.TeamProject'>.

        data is a dict defining the object attributes. Available attributes are:

        • team_id (required if not discovered on the parent objects)

        • greatest_access_level (required)

        Available keys for kwargs are:

        • sudo (string or int): run the request as another user (requires admin permissions)

    delete (id, **kwargs)
        Delete the object with ID id.

        Available keys for kwargs are:

        • sudo (string or int): run the request as another user (requires admin permissions)

    obj_cls
        alias of TeamProject

class gitlab.v3.objects.Todo (gl, data=None, **kwargs)
    Bases: gitlab.base.GitlabObject

    canCreate = False

    canGet = 'from_list'

    canUpdate = False

    optionalListAttrs = ['action', 'author_id', 'project_id', 'state', 'type']

class gitlab.v3.objects.TodoManager (gl, parent=None, args=[])
    Bases: gitlab.base.BaseManager

    Manager for <class 'gitlab.v3.objects.Todo'> objects.

    <class 'gitlab.v3.objects.Todo'> objects cannot be updated.
```

**list** (\*\*kwargs)

Returns a list of objects of type <class 'gitlab.v3.objects.TODO'>.

Available keys for kwargs are:

- action (optional)
- author\_id (optional)
- project\_id (optional)
- state (optional)
- type (optional)
- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (id, \*\*kwargs)

Get a single object of type <class 'gitlab.v3.objects.TODO'> using its id.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**delete** (id, \*\*kwargs)

Delete the object with ID id.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**delete\_all** (\*\*kwargs)

Mark all the todos as done.

#### Raises

- GitlabConnectionError – If the server cannot be reached.
- GitlabDeleteError – If the resource cannot be found

**Returns** The number of todos made done.

**obj\_cls**

alias of *TODO*

**class** gitlab.v3.objects.**User** (gl, data=None, \*\*kwargs)

Bases: *gitlab.base.GitlabObject*

**emails**

UserEmailManager - Manager for objects.

**keys**

UserKeyManager - Manager for objects.

**projects**

UserProjectManager - Manager for objects.

**save** (\*\*kwargs)

Send the modified object to the GitLab server. The following attributes are sent:

- email (required)

- username (required)
- name (required)
- password (optional)
- skype (optional)
- linkedin (optional)
- twitter (optional)
- projects\_limit (optional)
- extern\_uid (optional)
- provider (optional)
- bio (optional)
- admin (optional)
- can\_create\_group (optional)
- website\_url (optional)
- confirm (optional)
- external (optional)
- organization (optional)
- location (optional)

Available keys for `kwargs` are:

- sudo (string or int): run the request as another user (requires admin permissions)

**block** (*\*\*kwargs*)

Blocks the user.

**managers** = (('emails', 'UserEmailManager', [(('user\_id', 'id')]), ('keys', 'UserKeyManager', [(('user\_id', 'id')]), ('project

**optionalCreateAttrs** = ['password', 'reset\_password', 'skype', 'linkedin', 'twitter', 'projects\_limit', 'extern\_uid', 'p

**optionalUpdateAttrs** = ['password', 'skype', 'linkedin', 'twitter', 'projects\_limit', 'extern\_uid', 'provider', 'bio', 'ad

**requiredCreateAttrs** = ['email', 'username', 'name']

**requiredUpdateAttrs** = ['email', 'username', 'name']

**shortPrintAttr** = 'username'

**unblock** (*\*\*kwargs*)

Unblocks the user.

**class** gitlab.v3.objects.**UserEmail** (*gl, data=None, \*\*kwargs*)

Bases: *gitlab.base.GitlabObject*

**canUpdate** = False

**requiredCreateAttrs** = ['email']

**requiredUrlAttrs** = ['user\_id']

**shortPrintAttr** = 'email'

---

```

class gitlab.v3.objects.UserEmailManager (gl, parent=None, args=[])
    Bases: gitlab.base.BaseManager

    Manager for <class 'gitlab.v3.objects.UserEmail'> objects.

    <class 'gitlab.v3.objects.UserEmail'> objects cannot be updated.

    list (**kwargs)
        Returns a list of objects of type <class 'gitlab.v3.objects.UserEmail'>.

        Available keys for kwargs are:

            •per_page (int): number of item per page. May be limited by the server.

            •page (int): page to retrieve

            •all (bool): iterate over all the pages and return all the entries

            •sudo (string or int): run the request as another user (requires admin permissions)

    get (id, **kwargs)
        Get a single object of type <class 'gitlab.v3.objects.UserEmail'> using its id.

        Available keys for kwargs are:

            •sudo (string or int): run the request as another user (requires admin permissions)

    create (data, **kwargs)
        Create an object of type <class 'gitlab.v3.objects.UserEmail'>.

        data is a dict defining the object attributes. Available attributes are:

            •user_id (required if not discovered on the parent objects)

            •email (required)

        Available keys for kwargs are:

            •sudo (string or int): run the request as another user (requires admin permissions)

    delete (id, **kwargs)
        Delete the object with ID id.

        Available keys for kwargs are:

            •sudo (string or int): run the request as another user (requires admin permissions)

    obj_cls
        alias of UserEmail

class gitlab.v3.objects.UserKey (gl, data=None, **kwargs)
    Bases: gitlab.base.GitlabObject

    canGet = 'from_list'

    canUpdate = False

    requiredCreateAttrs = ['title', 'key']

    requiredUrlAttrs = ['user_id']

class gitlab.v3.objects.UserKeyManager (gl, parent=None, args=[])
    Bases: gitlab.base.BaseManager

    Manager for <class 'gitlab.v3.objects.UserKey'> objects.

    <class 'gitlab.v3.objects.UserKey'> objects cannot be updated.

```

**list** (\*\*kwargs)

Returns a list of objects of type <class 'gitlab.v3.objects.UserKey'>.

Available keys for kwargs are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (id, \*\*kwargs)

Get a single object of type <class 'gitlab.v3.objects.UserKey'> using its id.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**create** (data, \*\*kwargs)

Create an object of type <class 'gitlab.v3.objects.UserKey'>.

data is a dict defining the object attributes. Available attributes are:

- user\_id (required if not discovered on the parent objects)
- title (required)
- key (required)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**delete** (id, \*\*kwargs)

Delete the object with ID id.

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *UserKey*

**class** gitlab.v3.objects.**UserManager** (gl, parent=None, args=[])

Bases: *gitlab.base.BaseManager*

Manager for <class 'gitlab.v3.objects.User'> objects.

<class 'gitlab.v3.objects.User'> objects can be updated.

**list** (\*\*kwargs)

Returns a list of objects of type <class 'gitlab.v3.objects.User'>.

Available keys for kwargs are:

- per\_page (int): number of item per page. May be limited by the server.
- page (int): page to retrieve
- all (bool): iterate over all the pages and return all the entries
- sudo (string or int): run the request as another user (requires admin permissions)

**get** (id, \*\*kwargs)

Get a single object of type <class 'gitlab.v3.objects.User'> using its id.

Available keys for kwargs are:



- `sudo` (string or int): run the request as another user (requires admin permissions)

**create** (*data*, *\*\*kwargs*)

Create an object of type `<class 'gitlab.v3.objects.User'>`.

*data* is a dict defining the object attributes. Available attributes are:

- `email` (required)
- `username` (required)
- `name` (required)
- `password` (optional)
- `reset_password` (optional)
- `skype` (optional)
- `linkedin` (optional)
- `twitter` (optional)
- `projects_limit` (optional)
- `extern_uid` (optional)
- `provider` (optional)
- `bio` (optional)
- `admin` (optional)
- `can_create_group` (optional)
- `website_url` (optional)
- `confirm` (optional)
- `external` (optional)
- `organization` (optional)
- `location` (optional)

Available keys for *kwargs* are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**delete** (*id*, *\*\*kwargs*)

Delete the object with ID *id*.

Available keys for *kwargs* are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**get\_by\_username** (*username*, *\*\*kwargs*)

Get a user by its username.

#### Parameters

- **username** (*str*) – The name of the user.
- **\*\*kwargs** – Additional arguments to send to GitLab.

**Returns** The matching user.

**Return type** *User*

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabGetError` – If the server fails to perform the request.

**obj\_cls**

alias of `User`

**search** (*query*, *\*\*kwargs*)

Search users.

**Parameters**

- **query** (*str*) – The query string to send to GitLab for the search.
- **all** (*bool*) – If True, return all the items, without pagination
- **\*\*kwargs** – Additional arguments to send to GitLab.

**Returns** A list of matching users.

**Return type** `list(User)`

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabListError` – If the server fails to perform the request.

**class** `gitlab.v3.objects.UserProject` (*gl*, *data=None*, *\*\*kwargs*)

Bases: `gitlab.base.GitlabObject`

**canDelete** = False

**canGet** = False

**canList** = False

**canUpdate** = False

**optionalCreateAttrs** = ['default\_branch', 'issues\_enabled', 'wall\_enabled', 'merge\_requests\_enabled', 'wiki\_enabled']

**requiredCreateAttrs** = ['name']

**requiredUrlAttrs** = ['user\_id']

**class** `gitlab.v3.objects.UserProjectManager` (*gl*, *parent=None*, *args=[]*)

Bases: `gitlab.base.BaseManager`

Manager for <class 'gitlab.v3.objects.UserProject'> objects.

<class 'gitlab.v3.objects.UserProject'> objects **cannot** be updated.

**create** (*data*, *\*\*kwargs*)

Create an object of type <class 'gitlab.v3.objects.UserProject'>.

*data* is a dict defining the object attributes. Available attributes are:

- `user_id` (required if not discovered on the parent objects)
- `name` (required)
- `default_branch` (optional)
- `issues_enabled` (optional)
- `wall_enabled` (optional)
- `merge_requests_enabled` (optional)
- `wiki_enabled` (optional)

- snippets\_enabled (optional)
- public (optional)
- visibility\_level (optional)
- description (optional)
- builds\_enabled (optional)
- public\_builds (optional)
- import\_url (optional)
- only\_allow\_merge\_if\_build\_succeeds (optional)

Available keys for kwargs are:

- sudo (string or int): run the request as another user (requires admin permissions)

**obj\_cls**

alias of *UserProject*

## Module contents

### 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*

**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*

**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*

**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*

```
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.GetFromListMixin, 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.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

transfer_project (*args, **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.GetFromListMixin, gitlab.mixins.CreateMixin, gitlab.mixins.DeleteMixin, 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.GetFromListMixin, gitlab.base.RESTManager

class gitlab.v4.objects.GroupManager (gl, parent=None)
    Bases: gitlab.mixins.CRUDMixin, gitlab.base.RESTManager
```

```
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.GetFromListMixin, gitlab.mixins.CreateMixin, gitlab.mixins.UpdateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager

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

class gitlab.v4.objects.GroupProject (manager, attrs)
    Bases: gitlab.v4.objects.Project

class gitlab.v4.objects.GroupProjectManager (gl, parent=None)
    Bases: gitlab.mixins.GetFromListMixin, gitlab.base.RESTManager

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

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

class gitlab.v4.objects.Issue (manager, attrs)
    Bases: gitlab.base.RESTObject

class gitlab.v4.objects.IssueManager (gl, parent=None)
    Bases: gitlab.mixins.GetFromListMixin, gitlab.base.RESTManager

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

class gitlab.v4.objects.Namespace (manager, attrs)
    Bases: gitlab.base.RESTObject

class gitlab.v4.objects.NamespaceManager (gl, parent=None)
    Bases: gitlab.mixins.GetFromListMixin, gitlab.base.RESTManager

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

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

**archive** (\*args, \*\*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

**create\_fork\_relation** (\*args, \*\*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** (\*args, \*\*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

**repository\_archive** (\*args, \*\*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** (\*args, \*\*kwargs)

Return a blob 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 metadata

**Return type** `str`

**repository\_compare** (*\*args, \*\*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** (*\*args, \*\*kwargs*)

Return a list of contributors for the project.

**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 contributors

**Return type** *list*

**repository\_raw\_blob** (*\*args, \*\*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** (*\*args*, *\*\*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
- **\*\*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*

**share** (*\*args*, *\*\*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

**star** (*\*args*, *\*\*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

**trigger\_pipeline** (*\*args*, *\*\*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 commit SHA, a branch name, ...
- **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** (\*args, \*\*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

**unstar** (\*args, \*\*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** (\*args, \*\*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.GetFromListMixin`, `gitlab.mixins.CreateMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

**class** `gitlab.v4.objects.ProjectBoard` (*manager, attrs*)  
Bases: `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`

**class** `gitlab.v4.objects.ProjectBoardManager` (*gl, parent=None*)  
Bases: `gitlab.mixins.GetFromListMixin`, `gitlab.base.RESTManager`

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

**protect** (*\*args, \*\*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** (*\*args, \*\*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`

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

**cherry\_pick** (*\*args, \*\*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** (\*args, \*\*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*

**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`

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

Bases: `gitlab.mixins.RetrieveMixin`, `gitlab.mixins.CreateMixin`, `gitlab.base.RESTManager`

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

Bases: `gitlab.base.RESTObject`

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

Bases: `gitlab.mixins.GetFromListMixin`, `gitlab.mixins.CreateMixin`, `gitlab.base.RESTManager`

**create** (*data, \*\*kwargs*)

Create a new object.

**Parameters**

- **data** (*dict*) – Parameters to send to the server to create the resource
- **\*\*kwargs** – Extra data to send to the Gitlab 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.ProjectDeployment` (*manager, attrs*)

Bases: `gitlab.base.RESTObject`

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

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

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

**class** `gitlab.v4.objects.ProjectEnvironmentManager` (*gl, parent=None*)  
Bases: `gitlab.mixins.GetFromListMixin`, `gitlab.mixins.CreateMixin`, `gitlab.mixins.UpdateMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

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

**class** `gitlab.v4.objects.ProjectEventManager` (*gl, parent=None*)  
Bases: `gitlab.mixins.ListMixin`, `gitlab.base.RESTManager`

**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`

**create** (*\*args, \*\*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 Gitlab 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** (*\*args*, *\*\*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 Gitlab server (e.g. sudo)

**Raises**

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

**get** (*\*args*, *\*\*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 Gitlab 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** (*\*args*, *\*\*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 Gitlab 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`

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

Bases: `gitlab.base.RESTObject`

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

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

**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`

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

Bases: `gitlab.mixins.SubscribableMixin`, `gitlab.mixins.TODOMixin`, `gitlab.mixins.TimeTrackingMixin`, `gitlab.mixins.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

**move** (*\*args, \*\*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

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

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

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

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

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

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

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

Bases: `gitlab.base.RESTObject`

**artifacts** (*\*args, \*\*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** (*\*args*, *\*\*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

**erase** (*\*args*, *\*\*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** (*\*args*, *\*\*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** (*\*args*, *\*\*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** (*\*args*, *\*\*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** (\*args, \*\*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.ObjectDeleteMixin`, `gitlab.base.RESTObject`

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

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

**enable** (\*args, \*\*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** (\*args, \*\*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.GetFromListMixin, gitlab.mixins.CreateMixin, gitlab.
mixins.UpdateMixin, gitlab.mixins.DeleteMixin, gitlab.base.RESTManager

    delete (*args, **kwargs)
        Delete a Label on the server.

        Parameters
            • name – The name of the label
            • **kwargs – Extra options to send to the Gitlab server (e.g. sudo)

        Raises
            • GitlabAuthenticationError – If authentication is not correct.
            • GitlabDeleteError – If the server cannot perform the request.

class gitlab.v4.objects.ProjectManager (gl, parent=None)
    Bases: gitlab.mixins.CRUDMixin, gitlab.base.RESTManager

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

class gitlab.v4.objects.ProjectMergeRequest (manager, attrs)
    Bases: gitlab.mixins.SubscribableMixin, gitlab.mixins.TODOMixin, gitlab.
mixins.TimeTrackingMixin, gitlab.mixins.SaveMixin, gitlab.mixins.
ObjectDeleteMixin, gitlab.base.RESTObject

    cancel_merge_when_pipeline_succeeds (*args, **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 (*args, **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 (*args, **kwargs)
    List issues that will close on merge.”

        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 issues

**Return type** *RESTObjectList*

**commits** (*\*args, \*\*kwargs*)

List the merge request commits.

**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** The list of commits

**Return type** *RESTObjectList*

**merge** (*\*args, \*\*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

**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.ProjectMergeRequestManager` (*gl, parent=None*)

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

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

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

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

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

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

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

**issues** (*\*args, \*\*kwargs*)

List issues related to this milestone.

**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** The list of issues**Return type** *RESTObjectList***merge\_requests** (\*args, \*\*kwargs)

List the merge requests related to this milestone.

**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** The list of merge requests**Return type** *RESTObjectList*

**class** `gitlab.v4.objects.ProjectMilestoneManager` (*gl, parent=None*)  
 Bases: *gitlab.mixins.CRUDMixin, gitlab.base.RESTManager*

**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*

**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*

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

**cancel** (\*args, \*\*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** (\*args, \*\*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.ProjectPipelineManager` (*gl, parent=None*)  
 Bases: *gitlab.mixins.RetrieveMixin, gitlab.mixins.CreateMixin, gitlab.base.RESTManager*

**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.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`

**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`

**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** (*\*args*, *\*\*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 Gitlab 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={}, \*\*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 Gitlab 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.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

**content** (*\*args, \*\*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.ProjectSnippetManager` (*gl, parent=None*)

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

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

Bases: `gitlab.base.RESTObject`

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

Bases: `gitlab.mixins.RetrieveMixin`, `gitlab.mixins.CreateMixin`, `gitlab.base.RESTManager`

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

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

**set\_release\_description** (*\*args, \*\*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`

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

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

**take\_ownership** (*\*args, \*\*kwargs*)

Update the owner of a trigger.

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

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

**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`

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

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

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

Bases: `gitlab.mixins.RetrieveMixin`, `gitlab.mixins.UpdateMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

**all** (*\*args, \*\*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)*

**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** (*\*args, \*\*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** (*\*args, \*\*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** (*\*args, \*\*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** (*\*args, \*\*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.SaveMixin`, `gitlab.mixins.ObjectDeleteMixin`, `gitlab.base.RESTObject`

**content** (*\*args, \*\*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`

**public** (*\*args, \*\*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** (*\*args, \*\*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.GetFromListMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

**mark\_all\_as\_done** (*\*args, \*\*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** (*\*args, \*\*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** (*\*args, \*\*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.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`

**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.GetFromListMixin`, `gitlab.mixins.CreateMixin`, `gitlab.mixins.DeleteMixin`, `gitlab.base.RESTManager`

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

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

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

Bases: `gitlab.base.RESTObject`

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

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

## Module contents

# Submodules

## gitlab.base module

**class** `gitlab.base.BaseManager` (*gl*, *parent=None*, *args=[]*)

Bases: `object`

Base manager class for API operations.

Managers provide method to manage GitLab API objects, such as retrieval, listing, creation.

Inherited class must define the `obj_cls` attribute.

**obj\_cls**

*class* – class of objects wrapped by this manager.

**create** (*data*, *\*\*kwargs*)

Create a new object of class *obj\_cls*.

### Parameters

- **data** (*dict*) – The parameters to send to the GitLab server to create the object. Required and optional arguments are defined in the *requiredCreateAttrs* and *optionalCreateAttrs* of the *obj\_cls* class.
- **\*\*kwargs** – Additional arguments to send to GitLab.

**Returns** A newly create *obj\_cls* object.

**Return type** `object`

### Raises

- `NotImplementedError` – If objects cannot be created.
- `GitlabCreateError` – If the server fails to perform the request.

**delete** (*id*, *\*\*kwargs*)

Delete a GitLab object.

**Parameters** *id* – ID of the object to delete.

### Raises

- `NotImplementedError` – If objects cannot be deleted.
- `GitlabDeleteError` – If the server fails to perform the request.

**get** (*id=None*, *\*\*kwargs*)

Get a GitLab object.

### Parameters

- *id* – ID of the object to retrieve.
- **\*\*kwargs** – Additional arguments to send to GitLab.

**Returns** An object of class *obj\_cls*.

**Return type** `object`

### Raises

- `NotImplementedError` – If objects cannot be retrieved.
- `GitlabGetError` – If the server fails to perform the request.

**list** (*\*\*kwargs*)

Get a list of GitLab objects.

**Parameters** *\*\*kwargs* – Additional arguments to send to GitLab.

**Returns** A list of *obj\_cls* objects.

**Return type** *list*[object]

**Raises**

- `NotImplementedError` – If objects cannot be listed.
- `GitlabListError` – If the server fails to perform the request.

**obj\_cls** = None

**class** `gitlab.base.GitlabObject` (*gl, data=None, \*\*kwargs*)

Bases: `object`

Base class for all classes that interface with GitLab.

**save** (*\*\*kwargs*)

Send the modified object to the GitLab server. The following attributes are sent:

Available keys for *kwargs* are:

- `sudo` (string or int): run the request as another user (requires admin permissions)

**as\_dict** ()

Dump the object as a dict.

**canCreate** = True

Tells if GitLab-api allows creation of new objects.

**canDelete** = True

Tells if GitLab-api allows deleting object.

**canGet** = True

Tells if GitLab-api allows retrieving single objects.

**canList** = True

Tells if GitLab-api allows listing of objects.

**canUpdate** = True

Tells if GitLab-api allows updating object.

**classmethod** `create` (*gl, data, \*\*kwargs*)

Create an object.

**Parameters**

- **gl** (`gitlab.Gitlab`) – Gitlab object referencing the GitLab server.
- **data** (*dict*) – The data used to define the object.

**Returns** The new object.

**Return type** object

**Raises**

- `NotImplementedError` – If objects can't be created.

- `GitlabCreateError` – If the server cannot perform the request.

**delete** (*\*\*kwargs*)

**display** (*pretty*)

**classmethod get** (*gl, id, \*\*kwargs*)

Retrieve a single object.

**Parameters**

- **gl** (`gitlab.Gitlab`) – Gitlab object referencing the GitLab server.
- **id** (*int or str*) – ID of the object to retrieve.

**Returns** The found GitLab object.

**Return type** object

**Raises**

- `NotImplementedError` – If objects can't be retrieved.
- `GitlabGetError` – If the server cannot perform the request.

**getRequiresId = True**

Whether the object ID is required in the GET url.

**gitlab = None**

(`gitlab.Gitlab`) – Gitlab connection.

**idAttr = 'id'**

Name of the identifier of an object.

**json ()**

Dump the object as json.

**Returns** The json string.

**Return type** str

**classmethod list** (*gl, \*\*kwargs*)

Retrieve a list of objects from GitLab.

**Parameters**

- **gl** (`gitlab.Gitlab`) – Gitlab object referencing the GitLab server.
- **per\_page** (*int*) – Maximum number of items to return.
- **page** (*int*) – ID of the page to return when using pagination.

**Returns** A list of objects.

**Return type** *list*[object]

**Raises**

- `NotImplementedError` – If objects can't be listed.
- `GitlabListError` – If the server cannot perform the request.

**managers = []**

List of managers to create.

**optionalCreateAttrs = []**

Attributes that are optional when creating a new object.

**optionalGetAttrs** = []

Attributes that are optional when retrieving single object.

**optionalListAttrs** = []

Attributes that are optional when retrieving list of objects.

**optionalUpdateAttrs** = []

Attributes that are optional when updating an object.

**pretty\_print** (*depth=0*)

Print the object on the standard output (verbose).

**Parameters** *depth* (*int*) – Used internally for recursive call.

**requiredCreateAttrs** = []

Attributes that are required when creating a new object.

**requiredDeleteAttrs** = []

Attributes that are required when deleting object.

**requiredGetAttrs** = []

Attributes that are required when retrieving single object.

**requiredListAttrs** = []

Attributes that are required when retrieving list of objects.

**requiredUpdateAttrs** = []

Attributes that are required when updating an object.

**requiredUrlAttrs** = []

Attributes that are required for constructing url.

**save** (*\*\*kwargs*)

**shortPrintAttr** = None

Attribute to use as ID when displaying the object.

**short\_print** (*depth=0*)

Print the object on the standard output (verbose).

**Parameters** *depth* (*int*) – Used internally for recursive call.

**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

**next()**

**class** gitlab.base.**jsonEncoder** (*skipkeys=False, ensure\_ascii=True, check\_circular=True, allow\_nan=True, sort\_keys=False, indent=None, separators=None, encoding='utf-8', default=None*)

Bases: json.encoder.JSONEncoder

**default** (*obj*)

## 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*)

## gitlab.config module

**exception** gitlab.config.**ConfigError**

Bases: exceptions.Exception

**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*

## gitlab.const module

## gitlab.exceptions module

**exception** `gitlab.exceptions.GitlabAttachFileError` (`error_message=''`, `response_code=None`, `response_body=None`) re-re  
 Bases: `gitlab.exceptions.GitlabOperationError`

**exception** `gitlab.exceptions.GitlabAuthenticationError` (`error_message=''`, `response_code=None`, `response_body=None`) re-re  
 Bases: `gitlab.exceptions.GitlabError`

**exception** `gitlab.exceptions.GitlabBlockError` (`error_message=''`, `response_code=None`, `response_body=None`) re-  
 Bases: `gitlab.exceptions.GitlabOperationError`

**exception** `gitlab.exceptions.GitlabBuildCancelError` (`error_message=''`, `response_code=None`, `response_body=None`) re-re  
 Bases: `gitlab.exceptions.GitlabCancelError`

**exception** `gitlab.exceptions.GitlabBuildEraseError` (`error_message=''`, `response_code=None`, `response_body=None`) re-re  
 Bases: `gitlab.exceptions.GitlabRetryError`

**exception** `gitlab.exceptions.GitlabBuildPlayError` (`error_message=''`, `response_code=None`, `response_body=None`) re-  
 Bases: `gitlab.exceptions.GitlabRetryError`

**exception** `gitlab.exceptions.GitlabBuildRetryError` (`error_message=''`, `response_code=None`, `response_body=None`) re-re  
 Bases: `gitlab.exceptions.GitlabRetryError`

**exception** `gitlab.exceptions.GitlabCancelError` (`error_message=''`, `response_code=None`, `response_body=None`) re-  
 Bases: `gitlab.exceptions.GitlabOperationError`

**exception** `gitlab.exceptions.GitlabCherryPickError` (`error_message=''`, `response_code=None`, `response_body=None`) re-re  
 Bases: `gitlab.exceptions.GitlabOperationError`

**exception** `gitlab.exceptions.GitlabConnectionError` (`error_message=''`, `response_code=None`, `response_body=None`) re-re  
 Bases: `gitlab.exceptions.GitlabError`

**exception** `gitlab.exceptions.GitlabCreateError` (`error_message=''`, `response_code=None`, `response_body=None`) re-  
 Bases: `gitlab.exceptions.GitlabOperationError`

**exception** `gitlab.exceptions.GitlabDeleteError` (`error_message=''`, `response_code=None`, `response_body=None`) re-  
 Bases: `gitlab.exceptions.GitlabOperationError`

**exception** `gitlab.exceptions.GitlabError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `exceptions.Exception`

**exception** `gitlab.exceptions.GitlabGetError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `gitlab.exceptions.GitlabOperationError`

**exception** `gitlab.exceptions.GitlabHttpError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `gitlab.exceptions.GitlabError`

**exception** `gitlab.exceptions.GitlabJobCancelError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `gitlab.exceptions.GitlabCancelError`

**exception** `gitlab.exceptions.GitlabJobEraseError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `gitlab.exceptions.GitlabRetryError`

**exception** `gitlab.exceptions.GitlabJobPlayError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `gitlab.exceptions.GitlabRetryError`

**exception** `gitlab.exceptions.GitlabJobRetryError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `gitlab.exceptions.GitlabRetryError`

**exception** `gitlab.exceptions.GitlabListError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `gitlab.exceptions.GitlabOperationError`

**exception** `gitlab.exceptions.GitlabMRClosedError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `gitlab.exceptions.GitlabOperationError`

**exception** `gitlab.exceptions.GitlabMRForbiddenError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `gitlab.exceptions.GitlabOperationError`

**exception** `gitlab.exceptions.GitlabMROnBuildSuccessError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `gitlab.exceptions.GitlabOperationError`

**exception** `gitlab.exceptions.GitlabOperationError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `gitlab.exceptions.GitlabError`

**exception** `gitlab.exceptions.GitlabParsingError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `gitlab.exceptions.GitlabError`

**exception** `gitlab.exceptions.GitlabPipelineCancelError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `gitlab.exceptions.GitlabCancelError`

**exception** `gitlab.exceptions.GitlabPipelineRetryError` (*error\_message=''*, *response\_code=None*, *response\_body=None*)

Bases: `gitlab.exceptions.GitlabRetryError`



```
exception gitlab.exceptions.GitlabProjectDeployKeyError (error_message='',           re-  
                                                         sponse_code=None,           re-  
                                                         sponse_body=None)
```

Bases: `gitlab.exceptions.GitlabOperationError`

```
exception gitlab.exceptions.GitlabProtectError (error_message='',   response_code=None,  
                                                         response_body=None)
```

Bases: `gitlab.exceptions.GitlabOperationError`

```
exception gitlab.exceptions.GitlabRetryError (error_message='',   response_code=None, re-  
                                                         sponse_body=None)
```

Bases: `gitlab.exceptions.GitlabOperationError`

```
exception gitlab.exceptions.GitlabSubscribeError (error_message='', response_code=None,  
                                                         response_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='',   response_code=None,  
                                                         response_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, re-  
                                                         sponse_body=None)
```

Bases: `gitlab.exceptions.GitlabOperationError`

```
exception gitlab.exceptions.GitlabUploadError (error_message='', response_code=None, re-  
                                                         sponse_body=None)
```

Bases: `gitlab.exceptions.GitlabOperationError`

`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

`gitlab.exceptions.raise_error_from_response` (*response, error, expected\_code=200*)

Tries to parse gitlab error message from response and raises error.

Do nothing if the response status is the expected one.

If response status code is 401, raises instead GitlabAuthenticationError.

**Parameters**

- `response` – requests response object

- **error** – Error-class or dict {return-code => class} of possible error class to raise. Should be inherited from `GitLabError`

## gitlab.mixins module

**class** `gitlab.mixins.AccessRequestMixin`

Bases: `object`

**approve** (*\*args, \*\*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.CRUDMixin`

Bases: `gitlab.mixins.GetMixin`, `gitlab.mixins.ListMixin`, `gitlab.mixins.CreateMixin`, `gitlab.mixins.UpdateMixin`, `gitlab.mixins.DeleteMixin`

**class** `gitlab.mixins.CreateMixin`

Bases: `object`

**create** (*\*args, \*\*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 Gitlab 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** (\*args, \*\*kwargs)

Delete an object on the server.

**Parameters**

- **id** – ID of the object to delete
- **\*\*kwargs** – Extra options to send to the Gitlab server (e.g. sudo)

**Raises**

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

**class** `gitlab.mixins.GetFromListMixin`

Bases: `gitlab.mixins.ListMixin`

**get** (id, \*\*kwargs)

Retrieve a single object.

**Parameters**

- **id** (*int or str*) – ID of the object to retrieve
- **\*\*kwargs** – Extra options to send to the Gitlab 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.GetMixin`

Bases: object

**get** (\*args, \*\*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 Gitlab 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** (\*args, \*\*kwargs)

Retrieve a single object.

**Parameters** **\*\*kwargs** – Extra options to send to the Gitlab 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** (*\*args, \*\*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 Gitlab 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.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.SubscribableMixin`

Bases: `object`

**subscribe** (*\*args, \*\*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** (*\*args, \*\*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** (*\*args, \*\*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** (*\*args, \*\*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** (*\*args, \*\*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** (*\*args*, *\*\*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** (*\*args*, *\*\*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** (*\*args*, *\*\*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** (*\*args*, *\*\*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 Gitlab 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

## gitlab.utils module

`gitlab.utils.response_content` (*response, streamed, action, chunk\_size*)

## Module contents

Wrapper for the GitLab API.

```
class gitlab.Gitlab(url, private_token=None, email=None, password=None, ssl_verify=True,
                  http_username=None, http_password=None, timeout=None, api_version='3',
                  session=None)
```

Bases: object

Represents a GitLab server connection.

### Parameters

- **url** (*str*) – The URL of the GitLab server.
- **private\_token** (*str*) – The user private token
- **email** (*str*) – The user email or login.
- **password** (*str*) – The user password (associated with email).
- **ssl\_verify** (*bool*) – Whether SSL certificates should be validated.
- **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 (3 or 4)

### broadcastmessages

<class 'gitlab.v3.objects.BroadcastMessageManager'> manager for <class 'gitlab.v3.objects.BroadcastMessage'> objects.

### deploykeys

<class 'gitlab.v3.objects.DeployKeyManager'> manager for <class 'gitlab.v3.objects.DeployKey'> objects.

### gitignores

<class 'gitlab.v3.objects.GitignoreManager'> manager for <class 'gitlab.v3.objects.Gitignore'> objects.

### gitlabciyaml

<class 'gitlab.v3.objects.GitlabciyamlManager'> manager for <class 'gitlab.v3.objects.Gitlabciyaml'> objects.

**group\_accessrequests**

<class 'gitlab.v3.objects.GroupAccessRequestManager'> manager for <class 'gitlab.v3.objects.GroupAccessRequest'> objects.

**group\_issues**

<class 'gitlab.v3.objects.GroupIssueManager'> manager for <class 'gitlab.v3.objects.GroupIssue'> objects.

**group\_members**

<class 'gitlab.v3.objects.GroupMemberManager'> manager for <class 'gitlab.v3.objects.GroupMember'> objects.

**group\_notificationsettings**

<class 'gitlab.v3.objects.GroupNotificationSettingsManager'> manager for <class 'gitlab.v3.objects.GroupNotificationSettings'> objects.

**group\_project\_accessrequests**

<class 'gitlab.v3.objects.ProjectAccessRequestManager'> manager for <class 'gitlab.v3.objects.ProjectAccessRequest'> objects.

**group\_project\_board\_lists**

<class 'gitlab.v3.objects.ProjectBoardListManager'> manager for <class 'gitlab.v3.objects.ProjectBoardList'> objects.

**group\_project\_boards**

<class 'gitlab.v3.objects.ProjectBoardManager'> manager for <class 'gitlab.v3.objects.ProjectBoard'> objects.

**group\_project\_branches**

<class 'gitlab.v3.objects.ProjectBranchManager'> manager for <class 'gitlab.v3.objects.ProjectBranch'> objects.

**group\_project\_builds**

<class 'gitlab.v3.objects.ProjectBuildManager'> manager for <class 'gitlab.v3.objects.ProjectBuild'> objects.

**group\_project\_commits**

<class 'gitlab.v3.objects.ProjectCommitManager'> manager for <class 'gitlab.v3.objects.ProjectCommit'> objects.

**group\_project\_deployments**

<class 'gitlab.v3.objects.ProjectDeploymentManager'> manager for <class 'gitlab.v3.objects.ProjectDeployment'> objects.

**group\_project\_environments**

<class 'gitlab.v3.objects.ProjectEnvironmentManager'> manager for <class 'gitlab.v3.objects.ProjectEnvironment'> objects.

**group\_project\_events**

<class 'gitlab.v3.objects.ProjectEventManager'> manager for <class 'gitlab.v3.objects.ProjectEvent'> objects.

**group\_project\_files**

<class 'gitlab.v3.objects.ProjectFileManager'> manager for <class 'gitlab.v3.objects.ProjectFile'> objects.

**group\_project\_forks**

<class 'gitlab.v3.objects.ProjectForkManager'> manager for <class 'gitlab.v3.objects.ProjectFork'> objects.



**group\_project\_hooks**

<class 'gitlab.v3.objects.ProjectHookManager'> manager for <class 'gitlab.v3.objects.ProjectHook'> objects.

**group\_project\_issues**

<class 'gitlab.v3.objects.ProjectIssueManager'> manager for <class 'gitlab.v3.objects.ProjectIssue'> objects.

**group\_project\_keys**

<class 'gitlab.v3.objects.ProjectKeyManager'> manager for <class 'gitlab.v3.objects.ProjectKey'> objects.

**group\_project\_labels**

<class 'gitlab.v3.objects.ProjectLabelManager'> manager for <class 'gitlab.v3.objects.ProjectLabel'> objects.

**group\_project\_members**

<class 'gitlab.v3.objects.ProjectMemberManager'> manager for <class 'gitlab.v3.objects.ProjectMember'> objects.

**group\_project\_mergerequests**

<class 'gitlab.v3.objects.ProjectMergeRequestManager'> manager for <class 'gitlab.v3.objects.ProjectMergeRequest'> objects.

**group\_project\_milestones**

<class 'gitlab.v3.objects.ProjectMilestoneManager'> manager for <class 'gitlab.v3.objects.ProjectMilestone'> objects.

**group\_project\_notes**

<class 'gitlab.v3.objects.ProjectNoteManager'> manager for <class 'gitlab.v3.objects.ProjectNote'> objects.

**group\_project\_notificationsettings**

<class 'gitlab.v3.objects.ProjectNotificationSettingsManager'> manager for <class 'gitlab.v3.objects.ProjectNotificationSettings'> objects.

**group\_project\_pipelines**

<class 'gitlab.v3.objects.ProjectPipelineManager'> manager for <class 'gitlab.v3.objects.ProjectPipeline'> objects.

**group\_project\_runners**

<class 'gitlab.v3.objects.ProjectRunnerManager'> manager for <class 'gitlab.v3.objects.ProjectRunner'> objects.

**group\_project\_services**

<class 'gitlab.v3.objects.ProjectServiceManager'> manager for <class 'gitlab.v3.objects.ProjectService'> objects.

**group\_project\_snippets**

<class 'gitlab.v3.objects.ProjectSnippetManager'> manager for <class 'gitlab.v3.objects.ProjectSnippet'> objects.

**group\_project\_tags**

<class 'gitlab.v3.objects.ProjectTagManager'> manager for <class 'gitlab.v3.objects.ProjectTag'> objects.

**group\_project\_triggers**

<class 'gitlab.v3.objects.ProjectTriggerManager'> manager for <class 'gitlab.v3.objects.ProjectTrigger'> objects.

**group\_project\_variables**

<class 'gitlab.v3.objects.ProjectVariableManager'> manager for <class 'gitlab.v3.objects.ProjectVariable'> objects.

**group\_projects**

<class 'gitlab.v3.objects.GroupProjectManager'> manager for <class 'gitlab.v3.objects.GroupProject'> objects.

**groups**

<class 'gitlab.v3.objects.GroupManager'> manager for <class 'gitlab.v3.objects.Group'> objects.

**hooks**

<class 'gitlab.v3.objects.HookManager'> manager for <class 'gitlab.v3.objects.Hook'> objects.

**issues**

<class 'gitlab.v3.objects.IssueManager'> manager for <class 'gitlab.v3.objects.Issue'> objects.

**keys**

<class 'gitlab.v3.objects.KeyManager'> manager for <class 'gitlab.v3.objects.Key'> objects.

**licenses**

<class 'gitlab.v3.objects.LicenseManager'> manager for <class 'gitlab.v3.objects.License'> objects.

**namespaces**

<class 'gitlab.v3.objects.NamespaceManager'> manager for <class 'gitlab.v3.objects.Namespace'> objects.

**notificationsettings**

<class 'gitlab.v3.objects.NotificationSettingsManager'> manager for <class 'gitlab.v3.objects.NotificationSettings'> objects.

**project\_accessrequests**

<class 'gitlab.v3.objects.ProjectAccessRequestManager'> manager for <class 'gitlab.v3.objects.ProjectAccessRequest'> objects.

**project\_board\_lists**

<class 'gitlab.v3.objects.ProjectBoardListManager'> manager for <class 'gitlab.v3.objects.ProjectBoardList'> objects.

**project\_boards**

<class 'gitlab.v3.objects.ProjectBoardManager'> manager for <class 'gitlab.v3.objects.ProjectBoard'> objects.

**project\_branches**

<class 'gitlab.v3.objects.ProjectBranchManager'> manager for <class 'gitlab.v3.objects.ProjectBranch'> objects.

**project\_builds**

<class 'gitlab.v3.objects.ProjectBuildManager'> manager for <class 'gitlab.v3.objects.ProjectBuild'> objects.

**project\_commit\_comments**

<class 'gitlab.v3.objects.ProjectCommitCommentManager'> manager for <class 'gitlab.v3.objects.ProjectCommitComment'> objects.

**project\_commit\_statuses**

<class 'gitlab.v3.objects.ProjectCommitStatusManager'> manager for <class 'gitlab.v3.objects.ProjectCommitStatus'> objects.

**project\_commits**

<class 'gitlab.v3.objects.ProjectCommitManager'> manager for <class 'gitlab.v3.objects.ProjectCommit'> objects.

**project\_deployments**

<class 'gitlab.v3.objects.ProjectDeploymentManager'> manager for <class 'gitlab.v3.objects.ProjectDeployment'> objects.

**project\_environments**

<class 'gitlab.v3.objects.ProjectEnvironmentManager'> manager for <class 'gitlab.v3.objects.ProjectEnvironment'> objects.

**project\_events**

<class 'gitlab.v3.objects.ProjectEventManager'> manager for <class 'gitlab.v3.objects.ProjectEvent'> objects.

**project\_files**

<class 'gitlab.v3.objects.ProjectFileManager'> manager for <class 'gitlab.v3.objects.ProjectFile'> objects.

**project\_forks**

<class 'gitlab.v3.objects.ProjectForkManager'> manager for <class 'gitlab.v3.objects.ProjectFork'> objects.

**project\_hooks**

<class 'gitlab.v3.objects.ProjectHookManager'> manager for <class 'gitlab.v3.objects.ProjectHook'> objects.

**project\_issue\_notes**

<class 'gitlab.v3.objects.ProjectIssueNoteManager'> manager for <class 'gitlab.v3.objects.ProjectIssueNote'> objects.

**project\_issues**

<class 'gitlab.v3.objects.ProjectIssueManager'> manager for <class 'gitlab.v3.objects.ProjectIssue'> objects.

**project\_keys**

<class 'gitlab.v3.objects.ProjectKeyManager'> manager for <class 'gitlab.v3.objects.ProjectKey'> objects.

**project\_labels**

<class 'gitlab.v3.objects.ProjectLabelManager'> manager for <class 'gitlab.v3.objects.ProjectLabel'> objects.

**project\_members**

<class 'gitlab.v3.objects.ProjectMemberManager'> manager for <class 'gitlab.v3.objects.ProjectMember'> objects.

**project\_mergerequest\_diffs**

<class 'gitlab.v3.objects.ProjectMergeRequestDiffManager'> manager for <class 'gitlab.v3.objects.ProjectMergeRequestDiff'> objects.

**project\_mergerequest\_notes**

<class 'gitlab.v3.objects.ProjectMergeRequestNoteManager'> manager for <class 'gitlab.v3.objects.ProjectMergeRequestNote'> objects.

**project\_mergerequests**

<class 'gitlab.v3.objects.ProjectMergeRequestManager'> manager for <class 'gitlab.v3.objects.ProjectMergeRequest'> objects.

**project\_milestones**

<class 'gitlab.v3.objects.ProjectMilestoneManager'> manager for <class 'gitlab.v3.objects.ProjectMilestone'> objects.

**project\_notes**

<class 'gitlab.v3.objects.ProjectNoteManager'> manager for <class 'gitlab.v3.objects.ProjectNote'> objects.

**project\_notificationsettings**

<class 'gitlab.v3.objects.ProjectNotificationSettingsManager'> manager for <class 'gitlab.v3.objects.ProjectNotificationSettings'> objects.

**project\_pipelines**

<class 'gitlab.v3.objects.ProjectPipelineManager'> manager for <class 'gitlab.v3.objects.ProjectPipeline'> objects.

**project\_runners**

<class 'gitlab.v3.objects.ProjectRunnerManager'> manager for <class 'gitlab.v3.objects.ProjectRunner'> objects.

**project\_services**

<class 'gitlab.v3.objects.ProjectServiceManager'> manager for <class 'gitlab.v3.objects.ProjectService'> objects.

**project\_snippet\_notes**

<class 'gitlab.v3.objects.ProjectSnippetNoteManager'> manager for <class 'gitlab.v3.objects.ProjectSnippetNote'> objects.

**project\_snippets**

<class 'gitlab.v3.objects.ProjectSnippetManager'> manager for <class 'gitlab.v3.objects.ProjectSnippet'> objects.

**project\_tags**

<class 'gitlab.v3.objects.ProjectTagManager'> manager for <class 'gitlab.v3.objects.ProjectTag'> objects.

**project\_triggers**

<class 'gitlab.v3.objects.ProjectTriggerManager'> manager for <class 'gitlab.v3.objects.ProjectTrigger'> objects.

**project\_variables**

<class 'gitlab.v3.objects.ProjectVariableManager'> manager for <class 'gitlab.v3.objects.ProjectVariable'> objects.

**projects**

<class 'gitlab.v3.objects.ProjectManager'> manager for <class 'gitlab.v3.objects.Project'> objects.

**runners**

<class 'gitlab.v3.objects.RunnerManager'> manager for <class 'gitlab.v3.objects.Runner'> objects.

**settings**

<class 'gitlab.v3.objects.ApplicationSettingsManager'> manager for <class 'gitlab.v3.objects.ApplicationSettings'> objects.

**snippets**

<class 'gitlab.v3.objects.SnippetManager'> manager for <class 'gitlab.v3.objects.Snippet'> objects.

**team\_members**

<class 'gitlab.v3.objects.TeamMemberManager'> manager for <class 'gitlab.v3.objects.TeamMember'> objects.

**team\_projects**

<class 'gitlab.v3.objects.TeamProjectManager'> manager for <class 'gitlab.v3.objects.TeamProject'> objects.

**teams**

<class 'gitlab.v3.objects.TeamManager'> manager for <class 'gitlab.v3.objects.Team'> objects.

**todos**

<class 'gitlab.v3.objects.TODOManager'> manager for <class 'gitlab.v3.objects.TODO'> objects.

**user\_emails**

<class 'gitlab.v3.objects.UserEmailManager'> manager for <class 'gitlab.v3.objects.UserEmail'> objects.

**user\_keys**

<class 'gitlab.v3.objects.UserKeyManager'> manager for <class 'gitlab.v3.objects.UserKey'> objects.

**user\_projects**

<class 'gitlab.v3.objects.UserProjectManager'> manager for <class 'gitlab.v3.objects.UserProject'> objects.

**users**

<class 'gitlab.v3.objects.UserManager'> manager for <class 'gitlab.v3.objects.User'> objects.

**api\_version****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.

**create(obj, \*\*kwargs)**

Create an object on the GitLab server.

The object class and attributes define the request to be made on the GitLab server.

**Parameters**

- **obj** (*object*) – The object to create.
- **\*\*kwargs** – Additional arguments to send to GitLab.

**Returns**

A json representation of the object as returned by the GitLab server

**Return type** str**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabCreateError` – If the server fails to perform the request.

**credentials\_auth()**

Performs an authentication using email/password.

**delete(obj, id=None, \*\*kwargs)**

Delete an object on the GitLab server.

**Parameters**

- **obj** (*object or id*) – The object, or the class of the object to delete. If it is the class, the id of the object must be specified as the *id* arguments.
- **id** – ID of the object to remove. Required if *obj* is a class.
- **\*\*kwargs** – Additional arguments to send to GitLab.

**Returns** True if the operation succeeds.

**Return type** bool

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabDeleteError` – If the server fails to perform the request.

**email = None**

The user email

**enable\_debug()**

**static 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** (*obj\_class, id=None, \*\*kwargs*)

Request a GitLab resources.

**Parameters**

- **obj\_class** (*object*) – The class of resource to request.
- **id** – The object ID.
- **\*\*kwargs** – Additional arguments to send to GitLab.

**Returns** An object of class *obj\_class*.

**Return type** *obj\_class*

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabGetError` – If the server fails to perform the request.

**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 data to make the query (e.g. sudo, per\_page, page)

**Returns** The requests object.

**Raises** `GitlabHttpError` – When the return code is not 2xx

**http\_get** (*path, query\_data={}, streamed=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
- **\*\*kwargs** – Extra data to make the query (e.g. `sudo`, `per_page`, `page`)

**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*={}, *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 data to make the query (e.g. `sudo`, `per_page`, `page`, `all`)

**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*={}, *post\_data*={}, *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)
- **\*\*kwargs** – Extra data to make the query (e.g. `sudo`, `per_page`, `page`)

**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*={}, *post\_data*={}, *\*\*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)
- **\*\*kwargs** – Extra data to make the query (e.g. `sudo`, `per_page`, `page`)

**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*={}, *post\_data*={}, *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
- **\*\*kwargs** – Extra data to make the query (e.g. sudo, per\_page, page)

**Returns** A requests result object.

**Raises** `GitlabHttpError` – When the return code is not 2xx

**list** (*obj\_class*, *\*\*kwargs*)  
Request the listing of GitLab resources.

**Parameters**

- **obj\_class** (*object*) – The class of resource to request.
- **\*\*kwargs** – Additional arguments to send to GitLab.

**Returns** A list of objects of class *obj\_class*.

**Return type** *list*(*obj\_class*)

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabListError` – If the server fails to perform the request.

**password = None**

The user password (associated with email)

**session = None**

Create a session object for requests

**set\_credentials** (*email*, *password*)

Sets the email/login and password for authentication.

**Parameters**

- **email** (*str*) – The user email or login.
- **password** (*str*) – The user password.

**set\_token** (*token*)

Sets the private token for authentication.

**Parameters** **token** (*str*) – The private token.



**set\_url** (*url*)

Updates the GitLab URL.

**Parameters** **url** (*str*) – Base URL of the GitLab server.

**ssl\_verify** = None

Whether SSL certificates should be validated

**timeout** = None

Timeout to use for requests to gitlab server

**token\_auth** ()

Performs an authentication using the private token.

**update** (*obj*, *\*\*kwargs*)

Update an object on the GitLab server.

The object class and attributes define the request to be made on the GitLab server.

**Parameters**

- **obj** (*object*) – The object to create.
- **\*\*kwargs** – Additional arguments to send to GitLab.

**Returns**

A json representation of the object as returned by the GitLab server

**Return type** str

**Raises**

- `GitlabConnectionError` – If the server cannot be reached.
- `GitlabUpdateError` – If the server fails to perform the request.

**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, or ('unknown', 'unknown') if the server doesn't support this API call (gitlab < 8.13.0)

**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.

**next** ()



This page describes important changes between python-gitlab releases.

### 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.

### 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.

## 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](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](http://python-gitlab.readthedocs.io/en/stable/gl_objects/deploy_keys.html)

Related issue: <https://github.com/python-gitlab/python-gitlab/issues/212>

### 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()

### 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

## 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()

## 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

## Version 0.21.1 - 2017-05-25

- Fix the manager name for jobs in the Project class
- Fix the docs

## 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/corresponding/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

## 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)

## 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

## 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

## 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*

## 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()

## 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

## 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

## 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

## 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()

## 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

## Version 0.12.1 - 2016-02-03

- Fix a broken upload to pypi

## 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

## 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

## 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

## 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)

## Version 0.9.2 - 2015-07-11

- CLI: fix the update and delete subcommands (#62)

## Version 0.9.1 - 2015-05-15

- Fix the setup.py script

## 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`

## 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

## 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)

## 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

## 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

## 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

## Version 0.3 - 2013-08-27

- Use PRIVATE-TOKEN header for passing the auth token
- provide a AUTHORS file
- cli: support ssl\_verify config option
- Add ssl\_verify option to Gitlab object. Defaults to True
- Correct url for merge requests API.



## Version 0.2 - 2013-08-08

- provide a pip requirements.txt
- drop some debug statements

## Version 0.1 - 2013-07-08

- Initial release



## CHAPTER 9

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



**g**

[gitlab](#), 177  
[gitlab.base](#), 164  
[gitlab.cli](#), 168  
[gitlab.config](#), 168  
[gitlab.const](#), 169  
[gitlab.exceptions](#), 169  
[gitlab.mixins](#), 172  
[gitlab.utils](#), 177  
[gitlab.v3](#), 141  
[gitlab.v3.objects](#), 65  
[gitlab.v4](#), 164  
[gitlab.v4.objects](#), 141



**A**

AccessRequestMixin (class in gitlab.mixins), 172  
 accessrequests (gitlab.v3.objects.Group attribute), 71  
 accessrequests (gitlab.v3.objects.Project attribute), 82  
 add\_spent\_time() (gitlab.mixins.TimeTrackingMixin method), 175  
 add\_spent\_time() (gitlab.v3.objects.ProjectIssue method), 103  
 add\_spent\_time() (gitlab.v3.objects.ProjectMergeRequest method), 113  
 all() (gitlab.v3.objects.ProjectManager method), 110  
 all() (gitlab.v3.objects.RunnerManager method), 129  
 all() (gitlab.v4.objects.RunnerManager method), 160  
 api\_version (gitlab.Gitlab attribute), 183  
 ApplicationSettings (class in gitlab.v3.objects), 65  
 ApplicationSettings (class in gitlab.v4.objects), 141  
 ApplicationSettingsManager (class in gitlab.v3.objects), 66  
 ApplicationSettingsManager (class in gitlab.v4.objects), 141  
 approve() (gitlab.mixins.AccessRequestMixin method), 172  
 approve() (gitlab.v3.objects.GroupAccessRequest method), 72  
 approve() (gitlab.v3.objects.ProjectAccessRequest method), 88  
 archive() (gitlab.v3.objects.Project method), 84  
 archive() (gitlab.v4.objects.Project method), 143  
 archive\_() (gitlab.v3.objects.Project method), 84  
 artifacts() (gitlab.v3.objects.ProjectBuild method), 92  
 artifacts() (gitlab.v4.objects.ProjectJob method), 152  
 as\_dict() (gitlab.base.GitlabObject method), 165  
 attributes (gitlab.base.RESTObject attribute), 167  
 auth() (gitlab.Gitlab method), 183  
 available() (gitlab.v3.objects.ProjectServiceManager method), 122  
 available() (gitlab.v4.objects.ProjectServiceManager method), 158

**B**

BaseManager (class in gitlab.base), 164  
 blob() (gitlab.v3.objects.ProjectCommit method), 94  
 block() (gitlab.v3.objects.User method), 136  
 block() (gitlab.v4.objects.User method), 163  
 board\_lists (gitlab.v3.objects.Project attribute), 82  
 boards (gitlab.v3.objects.Project attribute), 82  
 branches (gitlab.v3.objects.Project attribute), 82  
 BroadcastMessage (class in gitlab.v3.objects), 66  
 BroadcastMessage (class in gitlab.v4.objects), 141  
 BroadcastMessageManager (class in gitlab.v3.objects), 67  
 BroadcastMessageManager (class in gitlab.v4.objects), 141  
 broadcastmessages (gitlab.Gitlab attribute), 177  
 builds (gitlab.v3.objects.Project attribute), 82  
 builds() (gitlab.v3.objects.ProjectCommit method), 94

**C**

cancel() (gitlab.v3.objects.ProjectBuild method), 92  
 cancel() (gitlab.v3.objects.ProjectPipeline method), 120  
 cancel() (gitlab.v4.objects.ProjectJob method), 153  
 cancel() (gitlab.v4.objects.ProjectPipeline method), 157  
 cancel\_merge\_when\_build\_succeeds() (gitlab.v3.objects.ProjectMergeRequest method), 113  
 cancel\_merge\_when\_pipeline\_succeeds() (gitlab.v4.objects.ProjectMergeRequest method), 155  
 canCreate (gitlab.base.GitlabObject attribute), 165  
 canCreate (gitlab.v3.objects.ApplicationSettings attribute), 66  
 canCreate (gitlab.v3.objects.CurrentUser attribute), 67  
 canCreate (gitlab.v3.objects.DeployKey attribute), 69  
 canCreate (gitlab.v3.objects.Gitignore attribute), 70  
 canCreate (gitlab.v3.objects.Gitlabciyaml attribute), 71  
 canCreate (gitlab.v3.objects.GroupIssue attribute), 73  
 canCreate (gitlab.v3.objects.GroupProject attribute), 77  
 canCreate (gitlab.v3.objects.Issue attribute), 79

- canCreate (gitlab.v3.objects.Key attribute), 79
- canCreate (gitlab.v3.objects.License attribute), 80
- canCreate (gitlab.v3.objects.Namespace attribute), 81
- canCreate (gitlab.v3.objects.NotificationSettings attribute), 82
- canCreate (gitlab.v3.objects.ProjectBoard attribute), 89
- canCreate (gitlab.v3.objects.ProjectBuild attribute), 92
- canCreate (gitlab.v3.objects.ProjectDeployment attribute), 97
- canCreate (gitlab.v3.objects.ProjectEvent attribute), 99
- canCreate (gitlab.v3.objects.ProjectMergeRequestDiff attribute), 114
- canCreate (gitlab.v3.objects.ProjectService attribute), 122
- canCreate (gitlab.v3.objects.Runner attribute), 129
- canCreate (gitlab.v3.objects.TODO attribute), 134
- canDelete (gitlab.base.GitlabObject attribute), 165
- canDelete (gitlab.v3.objects.ApplicationSettings attribute), 66
- canDelete (gitlab.v3.objects.CurrentUser attribute), 67
- canDelete (gitlab.v3.objects.DeployKey attribute), 69
- canDelete (gitlab.v3.objects.Gitignore attribute), 70
- canDelete (gitlab.v3.objects.Gitlabciyaml attribute), 71
- canDelete (gitlab.v3.objects.GroupIssue attribute), 73
- canDelete (gitlab.v3.objects.GroupProject attribute), 77
- canDelete (gitlab.v3.objects.Issue attribute), 79
- canDelete (gitlab.v3.objects.Key attribute), 79
- canDelete (gitlab.v3.objects.License attribute), 80
- canDelete (gitlab.v3.objects.Namespace attribute), 81
- canDelete (gitlab.v3.objects.NotificationSettings attribute), 82
- canDelete (gitlab.v3.objects.ProjectBoard attribute), 89
- canDelete (gitlab.v3.objects.ProjectBuild attribute), 92
- canDelete (gitlab.v3.objects.ProjectCommit attribute), 94
- canDelete (gitlab.v3.objects.ProjectCommitComment attribute), 94
- canDelete (gitlab.v3.objects.ProjectCommitStatus attribute), 96
- canDelete (gitlab.v3.objects.ProjectDeployment attribute), 97
- canDelete (gitlab.v3.objects.ProjectEvent attribute), 99
- canDelete (gitlab.v3.objects.ProjectFork attribute), 101
- canDelete (gitlab.v3.objects.ProjectIssueNote attribute), 106
- canDelete (gitlab.v3.objects.ProjectMergeRequestDiff attribute), 114
- canDelete (gitlab.v3.objects.ProjectMilestone attribute), 117
- canDelete (gitlab.v3.objects.ProjectNote attribute), 119
- canDelete (gitlab.v3.objects.ProjectPipeline attribute), 120
- canDelete (gitlab.v3.objects.ProjectSnippetNote attribute), 124
- canDelete (gitlab.v3.objects.ProjectTagRelease attribute), 126
- canDelete (gitlab.v3.objects.UserProject attribute), 140
- canGet (gitlab.base.GitlabObject attribute), 165
- canGet (gitlab.v3.objects.DeployKey attribute), 69
- canGet (gitlab.v3.objects.GroupAccessRequest attribute), 72
- canGet (gitlab.v3.objects.GroupIssue attribute), 73
- canGet (gitlab.v3.objects.GroupMember attribute), 75
- canGet (gitlab.v3.objects.GroupProject attribute), 77
- canGet (gitlab.v3.objects.Issue attribute), 79
- canGet (gitlab.v3.objects.Key attribute), 79
- canGet (gitlab.v3.objects.Namespace attribute), 81
- canGet (gitlab.v3.objects.ProjectAccessRequest attribute), 89
- canGet (gitlab.v3.objects.ProjectBoard attribute), 89
- canGet (gitlab.v3.objects.ProjectCommitComment attribute), 94
- canGet (gitlab.v3.objects.ProjectEnvironment attribute), 98
- canGet (gitlab.v3.objects.ProjectEvent attribute), 99
- canGet (gitlab.v3.objects.ProjectFork attribute), 101
- canGet (gitlab.v3.objects.ProjectLabel attribute), 108
- canGet (gitlab.v3.objects.ProjectTag attribute), 125
- canGet (gitlab.v3.objects.TODO attribute), 134
- canGet (gitlab.v3.objects.UserKey attribute), 137
- canGet (gitlab.v3.objects.UserProject attribute), 140
- canList (gitlab.base.GitlabObject attribute), 165
- canList (gitlab.v3.objects.ApplicationSettings attribute), 66
- canList (gitlab.v3.objects.CurrentUser attribute), 68
- canList (gitlab.v3.objects.NotificationSettings attribute), 82
- canList (gitlab.v3.objects.ProjectFile attribute), 100
- canList (gitlab.v3.objects.ProjectFork attribute), 101
- canList (gitlab.v3.objects.ProjectService attribute), 122
- canList (gitlab.v3.objects.ProjectTagRelease attribute), 126
- canList (gitlab.v3.objects.UserProject attribute), 140
- canUpdate (gitlab.base.GitlabObject attribute), 165
- canUpdate (gitlab.v3.objects.CurrentUser attribute), 68
- canUpdate (gitlab.v3.objects.CurrentUserEmail attribute), 68
- canUpdate (gitlab.v3.objects.CurrentUserKey attribute), 68
- canUpdate (gitlab.v3.objects.DeployKey attribute), 69
- canUpdate (gitlab.v3.objects.Gitignore attribute), 70
- canUpdate (gitlab.v3.objects.Gitlabciyaml attribute), 71
- canUpdate (gitlab.v3.objects.GroupAccessRequest attribute), 72
- canUpdate (gitlab.v3.objects.GroupIssue attribute), 73
- canUpdate (gitlab.v3.objects.GroupProject attribute), 77
- canUpdate (gitlab.v3.objects.Hook attribute), 78
- canUpdate (gitlab.v3.objects.Issue attribute), 79
- canUpdate (gitlab.v3.objects.Key attribute), 79
- canUpdate (gitlab.v3.objects.License attribute), 80



- canUpdate (gitlab.v3.objects.Namespace attribute), 81
- canUpdate (gitlab.v3.objects.ProjectAccessRequest attribute), 89
- canUpdate (gitlab.v3.objects.ProjectBoard attribute), 89
- canUpdate (gitlab.v3.objects.ProjectBranch attribute), 91
- canUpdate (gitlab.v3.objects.ProjectBuild attribute), 92
- canUpdate (gitlab.v3.objects.ProjectCommit attribute), 94
- canUpdate (gitlab.v3.objects.ProjectCommitComment attribute), 95
- canUpdate (gitlab.v3.objects.ProjectCommitStatus attribute), 96
- canUpdate (gitlab.v3.objects.ProjectDeployment attribute), 97
- canUpdate (gitlab.v3.objects.ProjectEvent attribute), 99
- canUpdate (gitlab.v3.objects.ProjectFork attribute), 101
- canUpdate (gitlab.v3.objects.ProjectKey attribute), 106
- canUpdate (gitlab.v3.objects.ProjectMergeRequestDiff attribute), 114
- canUpdate (gitlab.v3.objects.ProjectNote attribute), 119
- canUpdate (gitlab.v3.objects.ProjectPipeline attribute), 120
- canUpdate (gitlab.v3.objects.ProjectRunner attribute), 121
- canUpdate (gitlab.v3.objects.ProjectSnippetNote attribute), 124
- canUpdate (gitlab.v3.objects.ProjectTag attribute), 125
- canUpdate (gitlab.v3.objects.ProjectTrigger attribute), 127
- canUpdate (gitlab.v3.objects.Team attribute), 132
- canUpdate (gitlab.v3.objects.TeamMember attribute), 132
- canUpdate (gitlab.v3.objects.TeamProject attribute), 133
- canUpdate (gitlab.v3.objects.TODO attribute), 134
- canUpdate (gitlab.v3.objects.UserEmail attribute), 136
- canUpdate (gitlab.v3.objects.UserKey attribute), 137
- canUpdate (gitlab.v3.objects.UserProject attribute), 140
- changes() (gitlab.v3.objects.ProjectMergeRequest method), 113
- changes() (gitlab.v4.objects.ProjectMergeRequest method), 155
- cherry\_pick() (gitlab.v3.objects.ProjectCommit method), 94
- cherry\_pick() (gitlab.v4.objects.ProjectCommit method), 148
- closes\_issues() (gitlab.v3.objects.ProjectMergeRequest method), 113
- closes\_issues() (gitlab.v4.objects.ProjectMergeRequest method), 155
- cls\_to\_what() (in module gitlab.cli), 168
- commits (gitlab.v3.objects.Project attribute), 82
- commits() (gitlab.v3.objects.ProjectMergeRequest method), 113
- commits() (gitlab.v4.objects.ProjectMergeRequest method), 156
- compound\_metrics() (gitlab.v3.objects.SidekiqManager method), 130
- compound\_metrics() (gitlab.v4.objects.SidekiqManager method), 161
- ConfigError, 168
- content() (gitlab.v3.objects.ProjectSnippet method), 123
- content() (gitlab.v4.objects.ProjectSnippet method), 159
- content() (gitlab.v4.objects.Snippet method), 162
- create() (gitlab.base.BaseManager method), 164
- create() (gitlab.base.GitlabObject class method), 165
- create() (gitlab.Gitlab method), 183
- create() (gitlab.mixins.CreateMixin method), 172
- create() (gitlab.v3.objects.BroadcastMessageManager method), 67
- create() (gitlab.v3.objects.CurrentUserEmailManager method), 68
- create() (gitlab.v3.objects.CurrentUserKeyManager method), 69
- create() (gitlab.v3.objects.GroupAccessRequestManager method), 73
- create() (gitlab.v3.objects.GroupManager method), 74
- create() (gitlab.v3.objects.GroupMemberManager method), 76
- create() (gitlab.v3.objects.HookManager method), 78
- create() (gitlab.v3.objects.ProjectAccessRequestManager method), 89
- create() (gitlab.v3.objects.ProjectBoardListManager method), 90
- create() (gitlab.v3.objects.ProjectBranchManager method), 92
- create() (gitlab.v3.objects.ProjectCommitCommentManager method), 95
- create() (gitlab.v3.objects.ProjectCommitManager method), 96
- create() (gitlab.v3.objects.ProjectCommitStatusManager method), 97
- create() (gitlab.v3.objects.ProjectEnvironmentManager method), 98
- create() (gitlab.v3.objects.ProjectFileManager method), 100
- create() (gitlab.v3.objects.ProjectForkManager method), 101
- create() (gitlab.v3.objects.ProjectHookManager method), 102
- create() (gitlab.v3.objects.ProjectIssueManager method), 105
- create() (gitlab.v3.objects.ProjectIssueNoteManager method), 106
- create() (gitlab.v3.objects.ProjectKeyManager method), 107
- create() (gitlab.v3.objects.ProjectLabelManager method), 108
- create() (gitlab.v3.objects.ProjectManager method), 109

- create() (gitlab.v3.objects.ProjectMemberManager method), 112
- create() (gitlab.v3.objects.ProjectMergeRequestManager method), 115
- create() (gitlab.v3.objects.ProjectMergeRequestNoteManager method), 117
- create() (gitlab.v3.objects.ProjectMilestoneManager method), 118
- create() (gitlab.v3.objects.ProjectNoteManager method), 119
- create() (gitlab.v3.objects.ProjectPipelineManager method), 121
- create() (gitlab.v3.objects.ProjectRunnerManager method), 122
- create() (gitlab.v3.objects.ProjectSnippetManager method), 124
- create() (gitlab.v3.objects.ProjectSnippetNoteManager method), 125
- create() (gitlab.v3.objects.ProjectTagManager method), 126
- create() (gitlab.v3.objects.ProjectTriggerManager method), 127
- create() (gitlab.v3.objects.ProjectVariableManager method), 128
- create() (gitlab.v3.objects.SnippetManager method), 131
- create() (gitlab.v3.objects.TeamManager method), 132
- create() (gitlab.v3.objects.TeamMemberManager method), 133
- create() (gitlab.v3.objects.TeamProjectManager method), 134
- create() (gitlab.v3.objects.UserEmailManager method), 137
- create() (gitlab.v3.objects.UserKeyManager method), 138
- create() (gitlab.v3.objects.UserManager method), 139
- create() (gitlab.v3.objects.UserProjectManager method), 140
- create() (gitlab.v4.objects.ProjectCommitStatusManager method), 149
- create() (gitlab.v4.objects.ProjectFileManager method), 150
- create() (gitlab.v4.objects.ProjectPipelineManager method), 157
- create\_fork\_relation() (gitlab.v3.objects.Project method), 84
- create\_fork\_relation() (gitlab.v4.objects.Project method), 144
- CreateMixin (class in gitlab.mixins), 172
- credentials\_auth() (gitlab.Gitlab method), 183
- CRUDMixin (class in gitlab.mixins), 172
- CurrentUser (class in gitlab.v3.objects), 67
- CurrentUser (class in gitlab.v4.objects), 141
- CurrentUserEmail (class in gitlab.v3.objects), 68
- CurrentUserEmail (class in gitlab.v4.objects), 141
- CurrentUserEmailManager (class in gitlab.v3.objects), 68
- CurrentUserEmailManager (class in gitlab.v4.objects), 141
- CurrentUserKey (class in gitlab.v3.objects), 68
- CurrentUserKey (class in gitlab.v4.objects), 141
- CurrentUserKeyManager (class in gitlab.v3.objects), 69
- CurrentUserKeyManager (class in gitlab.v4.objects), 141
- CurrentUserManager (class in gitlab.v4.objects), 141
- ## D
- decode() (gitlab.v3.objects.ProjectFile method), 100
- decode() (gitlab.v4.objects.ProjectFile method), 150
- default() (gitlab.base.jsonEncoder method), 168
- delete() (gitlab.base.BaseManager method), 164
- delete() (gitlab.base.GitlabObject method), 166
- delete() (gitlab.Gitlab method), 183
- delete() (gitlab.mixins.DeleteMixin method), 172
- delete() (gitlab.mixins.ObjectDeleteMixin method), 174
- delete() (gitlab.v3.objects.BroadcastMessageManager method), 67
- delete() (gitlab.v3.objects.CurrentUserEmailManager method), 68
- delete() (gitlab.v3.objects.CurrentUserKeyManager method), 69
- delete() (gitlab.v3.objects.GroupAccessRequestManager method), 73
- delete() (gitlab.v3.objects.GroupManager method), 75
- delete() (gitlab.v3.objects.GroupMemberManager method), 76
- delete() (gitlab.v3.objects.HookManager method), 78
- delete() (gitlab.v3.objects.ProjectAccessRequestManager method), 89
- delete() (gitlab.v3.objects.ProjectBoardListManager method), 90
- delete() (gitlab.v3.objects.ProjectBranchManager method), 92
- delete() (gitlab.v3.objects.ProjectEnvironmentManager method), 99
- delete() (gitlab.v3.objects.ProjectFileManager method), 101
- delete() (gitlab.v3.objects.ProjectHookManager method), 103
- delete() (gitlab.v3.objects.ProjectIssueManager method), 105
- delete() (gitlab.v3.objects.ProjectKeyManager method), 107
- delete() (gitlab.v3.objects.ProjectLabelManager method), 109
- delete() (gitlab.v3.objects.ProjectManager method), 110
- delete() (gitlab.v3.objects.ProjectMemberManager method), 112
- delete() (gitlab.v3.objects.ProjectMergeRequestManager method), 116
- delete() (gitlab.v3.objects.ProjectMergeRequestNoteManager method), 117

- delete() (gitlab.v3.objects.ProjectRunnerManager method), 122
- delete() (gitlab.v3.objects.ProjectServiceManager method), 122
- delete() (gitlab.v3.objects.ProjectSnippetManager method), 124
- delete() (gitlab.v3.objects.ProjectTagManager method), 126
- delete() (gitlab.v3.objects.ProjectTriggerManager method), 127
- delete() (gitlab.v3.objects.ProjectVariableManager method), 128
- delete() (gitlab.v3.objects.RunnerManager method), 129
- delete() (gitlab.v3.objects.SnippetManager method), 131
- delete() (gitlab.v3.objects.TeamManager method), 132
- delete() (gitlab.v3.objects.TeamMemberManager method), 133
- delete() (gitlab.v3.objects.TeamProjectManager method), 134
- delete() (gitlab.v3.objects.TODOManager method), 135
- delete() (gitlab.v3.objects.UserEmailManager method), 137
- delete() (gitlab.v3.objects.UserKeyManager method), 138
- delete() (gitlab.v3.objects.UserManager method), 139
- delete() (gitlab.v4.objects.ProjectFile method), 150
- delete() (gitlab.v4.objects.ProjectFileManager method), 151
- delete() (gitlab.v4.objects.ProjectLabelManager method), 155
- delete\_all() (gitlab.v3.objects.TODOManager method), 135
- delete\_fork\_relation() (gitlab.v3.objects.Project method), 85
- delete\_fork\_relation() (gitlab.v4.objects.Project method), 144
- DeleteMixin (class in gitlab.mixins), 172
- DeployKey (class in gitlab.v3.objects), 69
- DeployKey (class in gitlab.v4.objects), 142
- DeployKeyManager (class in gitlab.v3.objects), 69
- DeployKeyManager (class in gitlab.v4.objects), 142
- deploykeys (gitlab.Gitlab attribute), 177
- deployments (gitlab.v3.objects.Project attribute), 83
- description\_attr (gitlab.v3.objects.ProjectIssue attribute), 103
- description\_attr (gitlab.v3.objects.ProjectIssueNote attribute), 106
- DEVELOPER\_ACCESS (gitlab.v3.objects.Group attribute), 72
- die() (in module gitlab.cli), 168
- diff() (gitlab.v3.objects.ProjectCommit method), 94
- diff() (gitlab.v4.objects.ProjectCommit method), 149
- diffs (gitlab.v3.objects.ProjectMergeRequest attribute), 112
- disable() (gitlab.v3.objects.ProjectKeyManager method), 107
- display() (gitlab.base.GitlabObject method), 166
- Dockerfile (class in gitlab.v4.objects), 142
- DockerfileManager (class in gitlab.v4.objects), 142
- ## E
- email (gitlab.Gitlab attribute), 184
- emails (gitlab.v3.objects.User attribute), 135
- enable() (gitlab.v3.objects.ProjectKeyManager method), 107
- enable() (gitlab.v4.objects.ProjectKeyManager method), 154
- enable\_debug() (gitlab.Gitlab method), 184
- environments (gitlab.v3.objects.Project attribute), 83
- erase() (gitlab.v3.objects.ProjectBuild method), 92
- erase() (gitlab.v4.objects.ProjectJob method), 153
- events (gitlab.v3.objects.Project attribute), 83
- ## F
- files (gitlab.v3.objects.Project attribute), 83
- forks (gitlab.v3.objects.Project attribute), 83
- from\_config() (gitlab.Gitlab static method), 184
- ## G
- get() (gitlab.base.BaseManager method), 164
- get() (gitlab.base.GitlabObject class method), 166
- get() (gitlab.Gitlab method), 184
- get() (gitlab.mixins.GetFromListMixin method), 173
- get() (gitlab.mixins.GetMixin method), 173
- get() (gitlab.mixins.GetWithoutIdMixin method), 173
- get() (gitlab.v3.objects.ApplicationSettingsManager method), 66
- get() (gitlab.v3.objects.BroadcastMessageManager method), 67
- get() (gitlab.v3.objects.CurrentUserEmailManager method), 68
- get() (gitlab.v3.objects.CurrentUserKeyManager method), 69
- get() (gitlab.v3.objects.DeployKeyManager method), 70
- get() (gitlab.v3.objects.GitignoreManager method), 70
- get() (gitlab.v3.objects.GitlabciyamlManager method), 71
- get() (gitlab.v3.objects.GroupAccessRequestManager method), 73
- get() (gitlab.v3.objects.GroupIssueManager method), 74
- get() (gitlab.v3.objects.GroupManager method), 74
- get() (gitlab.v3.objects.GroupMemberManager method), 76
- get() (gitlab.v3.objects.GroupNotificationSettingsManager method), 77
- get() (gitlab.v3.objects.GroupProjectManager method), 78
- get() (gitlab.v3.objects.HookManager method), 78
- get() (gitlab.v3.objects.IssueManager method), 79
- get() (gitlab.v3.objects.KeyManager method), 80
- get() (gitlab.v3.objects.LicenseManager method), 80

- get() (gitlab.v3.objects.NamespaceManager method), 81
- get() (gitlab.v3.objects.NotificationSettingsManager method), 82
- get() (gitlab.v3.objects.ProjectAccessRequestManager method), 89
- get() (gitlab.v3.objects.ProjectBoardListManager method), 90
- get() (gitlab.v3.objects.ProjectBoardManager method), 91
- get() (gitlab.v3.objects.ProjectBranchManager method), 91
- get() (gitlab.v3.objects.ProjectBuildManager method), 93
- get() (gitlab.v3.objects.ProjectCommitManager method), 96
- get() (gitlab.v3.objects.ProjectCommitStatusManager method), 96
- get() (gitlab.v3.objects.ProjectDeploymentManager method), 98
- get() (gitlab.v3.objects.ProjectEnvironmentManager method), 98
- get() (gitlab.v3.objects.ProjectEventManager method), 99
- get() (gitlab.v3.objects.ProjectFileManager method), 100
- get() (gitlab.v3.objects.ProjectHookManager method), 102
- get() (gitlab.v3.objects.ProjectIssueManager method), 105
- get() (gitlab.v3.objects.ProjectIssueNoteManager method), 106
- get() (gitlab.v3.objects.ProjectKeyManager method), 107
- get() (gitlab.v3.objects.ProjectLabelManager method), 108
- get() (gitlab.v3.objects.ProjectManager method), 109
- get() (gitlab.v3.objects.ProjectMemberManager method), 112
- get() (gitlab.v3.objects.ProjectMergeRequestDiffManager method), 115
- get() (gitlab.v3.objects.ProjectMergeRequestManager method), 115
- get() (gitlab.v3.objects.ProjectMergeRequestNoteManager method), 117
- get() (gitlab.v3.objects.ProjectMilestoneManager method), 118
- get() (gitlab.v3.objects.ProjectNoteManager method), 119
- get() (gitlab.v3.objects.ProjectNotificationSettingsManager method), 120
- get() (gitlab.v3.objects.ProjectPipelineManager method), 121
- get() (gitlab.v3.objects.ProjectRunnerManager method), 121
- get() (gitlab.v3.objects.ProjectServiceManager method), 122
- get() (gitlab.v3.objects.ProjectSnippetManager method), 124
- get() (gitlab.v3.objects.ProjectSnippetNoteManager method), 125
- get() (gitlab.v3.objects.ProjectTagManager method), 126
- get() (gitlab.v3.objects.ProjectTriggerManager method), 127
- get() (gitlab.v3.objects.ProjectVariableManager method), 128
- get() (gitlab.v3.objects.RunnerManager method), 129
- get() (gitlab.v3.objects.SnippetManager method), 131
- get() (gitlab.v3.objects.TeamManager method), 132
- get() (gitlab.v3.objects.TeamMemberManager method), 133
- get() (gitlab.v3.objects.TeamProjectManager method), 134
- get() (gitlab.v3.objects.TODOManager method), 135
- get() (gitlab.v3.objects.UserEmailManager method), 137
- get() (gitlab.v3.objects.UserKeyManager method), 138
- get() (gitlab.v3.objects.UserManager method), 138
- get() (gitlab.v4.objects.ProjectFileManager method), 151
- get() (gitlab.v4.objects.ProjectServiceManager method), 158
- get\_by\_username() (gitlab.v3.objects.UserManager method), 139
- get\_create\_attrs() (gitlab.mixins.CreateMixin method), 172
- get\_id() (gitlab.base.RESTObject method), 168
- get\_update\_attrs() (gitlab.mixins.UpdateMixin method), 176
- GetFromListMixin (class in gitlab.mixins), 173
- GetMixin (class in gitlab.mixins), 173
- getRequiresId (gitlab.base.GitlabObject attribute), 166
- getRequiresId (gitlab.v3.objects.ApplicationSettings attribute), 66
- getRequiresId (gitlab.v3.objects.NotificationSettings attribute), 82
- getRequiresId (gitlab.v3.objects.ProjectFile attribute), 100
- getRequiresId (gitlab.v3.objects.ProjectService attribute), 122
- GetWithoutIdMixin (class in gitlab.mixins), 173
- Gitignore (class in gitlab.v3.objects), 70
- Gitignore (class in gitlab.v4.objects), 142
- GitignoreManager (class in gitlab.v3.objects), 70
- GitignoreManager (class in gitlab.v4.objects), 142
- gitignores (gitlab.Gitlab attribute), 177
- Gitlab (class in gitlab), 177
- gitlab (gitlab.base.GitlabObject attribute), 166
- gitlab (module), 177
- gitlab.base (module), 164
- gitlab.cli (module), 168
- gitlab.config (module), 168
- gitlab.const (module), 169
- gitlab.exceptions (module), 169
- gitlab.mixins (module), 172

- gitlab.utils (module), 177
- gitlab.v3 (module), 141
- gitlab.v3.objects (module), 65
- gitlab.v4 (module), 164
- gitlab.v4.objects (module), 141
- GitlabAttachFileError, 169
- GitlabAuthenticationError, 169
- GitlabBlockError, 169
- GitlabBuildCancelError, 169
- GitlabBuildEraseError, 169
- GitlabBuildPlayError, 169
- GitlabBuildRetryError, 169
- GitlabCancelError, 169
- GitlabCherryPickError, 169
- Gitlabciyaml (class in gitlab.v3.objects), 71
- Gitlabciyaml (class in gitlab.v4.objects), 142
- GitlabciyamlManager (class in gitlab.v3.objects), 71
- GitlabciyamlManager (class in gitlab.v4.objects), 142
- gitlabciyaml (gitlab.Gitlab attribute), 177
- GitlabConfigParser (class in gitlab.config), 168
- GitlabConnectionError, 169
- GitlabCreateError, 169
- GitlabDataError, 168
- GitlabDeleteError, 169
- GitlabError, 169
- GitlabGetError, 170
- GitlabHttpError, 170
- GitlabIDError, 168
- GitlabJobCancelError, 170
- GitlabJobEraseError, 170
- GitlabJobPlayError, 170
- GitlabJobRetryError, 170
- GitlabList (class in gitlab), 187
- GitlabListError, 170
- GitlabMRClosedError, 170
- GitlabMRForbiddenError, 170
- GitlabMROnBuildSuccessError, 170
- GitlabObject (class in gitlab.base), 165
- GitlabOperationError, 170
- GitlabParsingError, 170
- GitlabPipelineCancelError, 170
- GitlabPipelineRetryError, 170
- GitlabProjectDeployKeyError, 170
- GitlabProtectError, 171
- GitlabRetryError, 171
- GitlabSubscribeError, 171
- GitlabTimeTrackingError, 171
- GitlabTodoError, 171
- GitlabTransferProjectError, 171
- GitlabUnblockError, 171
- GitlabUnsubscribeError, 171
- GitlabUpdateError, 171
- GitlabUploadError, 171
- Group (class in gitlab.v3.objects), 71
- Group (class in gitlab.v4.objects), 142
- group\_accessrequests (gitlab.Gitlab attribute), 177
- group\_issues (gitlab.Gitlab attribute), 178
- group\_members (gitlab.Gitlab attribute), 178
- group\_notificationsettings (gitlab.Gitlab attribute), 178
- group\_project\_accessrequests (gitlab.Gitlab attribute), 178
- group\_project\_board\_lists (gitlab.Gitlab attribute), 178
- group\_project\_boards (gitlab.Gitlab attribute), 178
- group\_project\_branches (gitlab.Gitlab attribute), 178
- group\_project\_builds (gitlab.Gitlab attribute), 178
- group\_project\_commits (gitlab.Gitlab attribute), 178
- group\_project\_deployments (gitlab.Gitlab attribute), 178
- group\_project\_environments (gitlab.Gitlab attribute), 178
- group\_project\_events (gitlab.Gitlab attribute), 178
- group\_project\_files (gitlab.Gitlab attribute), 178
- group\_project\_forks (gitlab.Gitlab attribute), 178
- group\_project\_hooks (gitlab.Gitlab attribute), 178
- group\_project\_issues (gitlab.Gitlab attribute), 179
- group\_project\_keys (gitlab.Gitlab attribute), 179
- group\_project\_labels (gitlab.Gitlab attribute), 179
- group\_project\_members (gitlab.Gitlab attribute), 179
- group\_project\_mergerequests (gitlab.Gitlab attribute), 179
- group\_project\_milestones (gitlab.Gitlab attribute), 179
- group\_project\_notes (gitlab.Gitlab attribute), 179
- group\_project\_notificationsettings (gitlab.Gitlab attribute), 179
- group\_project\_pipelines (gitlab.Gitlab attribute), 179
- group\_project\_runners (gitlab.Gitlab attribute), 179
- group\_project\_services (gitlab.Gitlab attribute), 179
- group\_project\_snippets (gitlab.Gitlab attribute), 179
- group\_project\_tags (gitlab.Gitlab attribute), 179
- group\_project\_triggers (gitlab.Gitlab attribute), 179
- group\_project\_variables (gitlab.Gitlab attribute), 179
- group\_projects (gitlab.Gitlab attribute), 180
- GroupAccessRequest (class in gitlab.v3.objects), 72
- GroupAccessRequest (class in gitlab.v4.objects), 142
- GroupAccessRequestManager (class in gitlab.v3.objects), 72
- GroupAccessRequestManager (class in gitlab.v4.objects), 142
- GroupIssue (class in gitlab.v3.objects), 73
- GroupIssue (class in gitlab.v4.objects), 142
- GroupIssueManager (class in gitlab.v3.objects), 73
- GroupIssueManager (class in gitlab.v4.objects), 142
- GroupManager (class in gitlab.v3.objects), 74
- GroupManager (class in gitlab.v4.objects), 142
- GroupMember (class in gitlab.v3.objects), 75
- GroupMember (class in gitlab.v4.objects), 142
- GroupMemberManager (class in gitlab.v3.objects), 75
- GroupMemberManager (class in gitlab.v4.objects), 143
- GroupNotificationSettings (class in gitlab.v3.objects), 76

GroupNotificationSettings (class in gitlab.v4.objects), 143

GroupNotificationSettingsManager (class in gitlab.v3.objects), 77

GroupNotificationSettingsManager (class in gitlab.v4.objects), 143

GroupProject (class in gitlab.v3.objects), 77

GroupProject (class in gitlab.v4.objects), 143

GroupProjectManager (class in gitlab.v3.objects), 77

GroupProjectManager (class in gitlab.v4.objects), 143

groups (gitlab.Gitlab attribute), 180

GroupVariable (class in gitlab.v4.objects), 143

GroupVariableManager (class in gitlab.v4.objects), 143

GUEST\_ACCESS (gitlab.v3.objects.Group attribute), 72

## H

headers (gitlab.Gitlab attribute), 184

Hook (class in gitlab.v3.objects), 78

Hook (class in gitlab.v4.objects), 143

HookManager (class in gitlab.v3.objects), 78

HookManager (class in gitlab.v4.objects), 143

hooks (gitlab.Gitlab attribute), 180

hooks (gitlab.v3.objects.Project attribute), 83

http\_delete() (gitlab.Gitlab method), 184

http\_get() (gitlab.Gitlab method), 184

http\_list() (gitlab.Gitlab method), 185

http\_post() (gitlab.Gitlab method), 185

http\_put() (gitlab.Gitlab method), 185

http\_request() (gitlab.Gitlab method), 186

## I

idAttr (gitlab.base.GitlabObject attribute), 166

idAttr (gitlab.v3.objects.Gitignore attribute), 70

idAttr (gitlab.v3.objects.Gitlabciyaml attribute), 71

idAttr (gitlab.v3.objects.License attribute), 80

idAttr (gitlab.v3.objects.ProjectBranch attribute), 91

idAttr (gitlab.v3.objects.ProjectLabel attribute), 108

idAttr (gitlab.v3.objects.ProjectTag attribute), 125

idAttr (gitlab.v3.objects.ProjectTrigger attribute), 127

idAttr (gitlab.v3.objects.ProjectVariable attribute), 128

Issue (class in gitlab.v3.objects), 79

Issue (class in gitlab.v4.objects), 143

IssueManager (class in gitlab.v3.objects), 79

IssueManager (class in gitlab.v4.objects), 143

issues (gitlab.Gitlab attribute), 180

issues (gitlab.v3.objects.Group attribute), 71

issues (gitlab.v3.objects.Project attribute), 83

issues() (gitlab.v3.objects.ProjectMilestone method), 117

issues() (gitlab.v4.objects.ProjectMilestone method), 156

## J

job\_stats() (gitlab.v3.objects.SidekiqManager method), 130

job\_stats() (gitlab.v4.objects.SidekiqManager method), 161

json() (gitlab.base.GitlabObject method), 166

jsonEncoder (class in gitlab.base), 168

## K

keep\_artifacts() (gitlab.v3.objects.ProjectBuild method), 92

keep\_artifacts() (gitlab.v4.objects.ProjectJob method), 153

Key (class in gitlab.v3.objects), 79

KeyManager (class in gitlab.v3.objects), 79

keys (gitlab.Gitlab attribute), 180

keys (gitlab.v3.objects.Project attribute), 83

keys (gitlab.v3.objects.User attribute), 135

## L

labels (gitlab.v3.objects.Project attribute), 83

License (class in gitlab.v3.objects), 80

License (class in gitlab.v4.objects), 143

LicenseManager (class in gitlab.v3.objects), 80

LicenseManager (class in gitlab.v4.objects), 143

licenses (gitlab.Gitlab attribute), 180

list() (gitlab.base.BaseManager method), 165

list() (gitlab.base.GitlabObject class method), 166

list() (gitlab.Gitlab method), 186

list() (gitlab.mixins.ListMixin method), 174

list() (gitlab.v3.objects.BroadcastMessageManager method), 67

list() (gitlab.v3.objects.CurrentUserEmailManager method), 68

list() (gitlab.v3.objects.CurrentUserKeyManager method), 69

list() (gitlab.v3.objects.DeployKeyManager method), 70

list() (gitlab.v3.objects.GitignoreManager method), 70

list() (gitlab.v3.objects.GitlabciyamlManager method), 71

list() (gitlab.v3.objects.GroupAccessRequestManager method), 73

list() (gitlab.v3.objects.GroupIssueManager method), 73

list() (gitlab.v3.objects.GroupManager method), 74

list() (gitlab.v3.objects.GroupMemberManager method), 75

list() (gitlab.v3.objects.GroupProjectManager method), 77

list() (gitlab.v3.objects.HookManager method), 78

list() (gitlab.v3.objects.IssueManager method), 79

list() (gitlab.v3.objects.KeyManager method), 80

list() (gitlab.v3.objects.LicenseManager method), 80

list() (gitlab.v3.objects.NamespaceManager method), 81

list() (gitlab.v3.objects.ProjectAccessRequestManager method), 89

list() (gitlab.v3.objects.ProjectBoardListManager method), 90

- list() (gitlab.v3.objects.ProjectBoardManager method), 91
  - list() (gitlab.v3.objects.ProjectBranchManager method), 91
  - list() (gitlab.v3.objects.ProjectBuildManager method), 93
  - list() (gitlab.v3.objects.ProjectCommitCommentManager method), 95
  - list() (gitlab.v3.objects.ProjectCommitManager method), 95
  - list() (gitlab.v3.objects.ProjectCommitStatusManager method), 96
  - list() (gitlab.v3.objects.ProjectDeploymentManager method), 97
  - list() (gitlab.v3.objects.ProjectEnvironmentManager method), 98
  - list() (gitlab.v3.objects.ProjectEventManager method), 99
  - list() (gitlab.v3.objects.ProjectHookManager method), 102
  - list() (gitlab.v3.objects.ProjectIssueManager method), 104
  - list() (gitlab.v3.objects.ProjectIssueNoteManager method), 106
  - list() (gitlab.v3.objects.ProjectKeyManager method), 107
  - list() (gitlab.v3.objects.ProjectLabelManager method), 108
  - list() (gitlab.v3.objects.ProjectManager method), 109
  - list() (gitlab.v3.objects.ProjectMemberManager method), 111
  - list() (gitlab.v3.objects.ProjectMergeRequestDiffManager method), 115
  - list() (gitlab.v3.objects.ProjectMergeRequestManager method), 115
  - list() (gitlab.v3.objects.ProjectMergeRequestNoteManager method), 116
  - list() (gitlab.v3.objects.ProjectMilestoneManager method), 118
  - list() (gitlab.v3.objects.ProjectNoteManager method), 119
  - list() (gitlab.v3.objects.ProjectPipelineManager method), 121
  - list() (gitlab.v3.objects.ProjectRunnerManager method), 121
  - list() (gitlab.v3.objects.ProjectSnippetManager method), 124
  - list() (gitlab.v3.objects.ProjectSnippetNoteManager method), 125
  - list() (gitlab.v3.objects.ProjectTagManager method), 126
  - list() (gitlab.v3.objects.ProjectTriggerManager method), 127
  - list() (gitlab.v3.objects.ProjectVariableManager method), 128
  - list() (gitlab.v3.objects.RunnerManager method), 129
  - list() (gitlab.v3.objects.SnippetManager method), 131
  - list() (gitlab.v3.objects.TeamManager method), 132
  - list() (gitlab.v3.objects.TeamMemberManager method), 133
  - list() (gitlab.v3.objects.TeamProjectManager method), 134
  - list() (gitlab.v3.objects.TODOManager method), 134
  - list() (gitlab.v3.objects.UserEmailManager method), 137
  - list() (gitlab.v3.objects.UserKeyManager method), 137
  - list() (gitlab.v3.objects.UserManager method), 138
  - ListMixin (class in gitlab.mixins), 174
- ## M
- main() (in module gitlab.cli), 168
  - managers (gitlab.base.GitlabObject attribute), 166
  - managers (gitlab.v3.objects.CurrentUser attribute), 68
  - managers (gitlab.v3.objects.Group attribute), 72
  - managers (gitlab.v3.objects.Project attribute), 85
  - managers (gitlab.v3.objects.ProjectBoard attribute), 89
  - managers (gitlab.v3.objects.ProjectCommit attribute), 94
  - managers (gitlab.v3.objects.ProjectIssue attribute), 103
  - managers (gitlab.v3.objects.ProjectMergeRequest attribute), 113
  - managers (gitlab.v3.objects.ProjectSnippet attribute), 123
  - managers (gitlab.v3.objects.Team attribute), 132
  - managers (gitlab.v3.objects.User attribute), 136
  - mark\_all\_as\_done() (gitlab.v4.objects.TODOManager method), 162
  - mark\_as\_done() (gitlab.v4.objects.TODO method), 162
  - MASTER\_ACCESS (gitlab.v3.objects.Group attribute), 72
  - members (gitlab.v3.objects.Group attribute), 71
  - members (gitlab.v3.objects.Project attribute), 83
  - merge() (gitlab.v3.objects.ProjectMergeRequest method), 113
  - merge() (gitlab.v4.objects.ProjectMergeRequest method), 156
  - merge\_requests() (gitlab.v3.objects.ProjectMilestone method), 117
  - merge\_requests() (gitlab.v4.objects.ProjectMilestone method), 157
  - mergerequests (gitlab.v3.objects.Project attribute), 83
  - milestones (gitlab.v3.objects.Project attribute), 83
  - move() (gitlab.v3.objects.ProjectIssue method), 103
  - move() (gitlab.v4.objects.ProjectIssue method), 152
- ## N
- Namespace (class in gitlab.v3.objects), 81
  - Namespace (class in gitlab.v4.objects), 143
  - NamespaceManager (class in gitlab.v3.objects), 81
  - NamespaceManager (class in gitlab.v4.objects), 143
  - namespaces (gitlab.Gitlab attribute), 180
  - next() (gitlab.base.RESTObjectList method), 168
  - next() (gitlab.GitlabList method), 187
  - notes (gitlab.v3.objects.Project attribute), 83
  - notes (gitlab.v3.objects.ProjectIssue attribute), 103

- notes (gitlab.v3.objects.ProjectMergeRequest attribute), 112
- notes (gitlab.v3.objects.ProjectSnippet attribute), 123
- NotificationSettings (class in gitlab.v3.objects), 81
- NotificationSettings (class in gitlab.v4.objects), 143
- notificationsettings (gitlab.Gitlab attribute), 180
- notificationsettings (gitlab.v3.objects.Group attribute), 71
- notificationsettings (gitlab.v3.objects.Project attribute), 83
- NotificationSettingsManager (class in gitlab.v3.objects), 82
- NotificationSettingsManager (class in gitlab.v4.objects), 143
- NoUpdateMixin (class in gitlab.mixins), 174
- ## O
- obj\_cls (gitlab.base.BaseManager attribute), 164, 165
- obj\_cls (gitlab.v3.objects.ApplicationSettingsManager attribute), 66
- obj\_cls (gitlab.v3.objects.BroadcastMessageManager attribute), 67
- obj\_cls (gitlab.v3.objects.CurrentUserEmailManager attribute), 68
- obj\_cls (gitlab.v3.objects.CurrentUserKeyManager attribute), 69
- obj\_cls (gitlab.v3.objects.DeployKeyManager attribute), 70
- obj\_cls (gitlab.v3.objects.GitignoreManager attribute), 70
- obj\_cls (gitlab.v3.objects.GitlabciyamlManager attribute), 71
- obj\_cls (gitlab.v3.objects.GroupAccessRequestManager attribute), 73
- obj\_cls (gitlab.v3.objects.GroupIssueManager attribute), 74
- obj\_cls (gitlab.v3.objects.GroupManager attribute), 75
- obj\_cls (gitlab.v3.objects.GroupMemberManager attribute), 76
- obj\_cls (gitlab.v3.objects.GroupNotificationSettingsManager attribute), 77
- obj\_cls (gitlab.v3.objects.GroupProjectManager attribute), 78
- obj\_cls (gitlab.v3.objects.HookManager attribute), 78
- obj\_cls (gitlab.v3.objects.IssueManager attribute), 79
- obj\_cls (gitlab.v3.objects.KeyManager attribute), 80
- obj\_cls (gitlab.v3.objects.LicenseManager attribute), 81
- obj\_cls (gitlab.v3.objects.NamespaceManager attribute), 81
- obj\_cls (gitlab.v3.objects.NotificationSettingsManager attribute), 82
- obj\_cls (gitlab.v3.objects.ProjectAccessRequestManager attribute), 89
- obj\_cls (gitlab.v3.objects.ProjectBoardListManager attribute), 90
- obj\_cls (gitlab.v3.objects.ProjectBoardManager attribute), 91
- obj\_cls (gitlab.v3.objects.ProjectBranchManager attribute), 92
- obj\_cls (gitlab.v3.objects.ProjectBuildManager attribute), 93
- obj\_cls (gitlab.v3.objects.ProjectCommitCommentManager attribute), 95
- obj\_cls (gitlab.v3.objects.ProjectCommitManager attribute), 96
- obj\_cls (gitlab.v3.objects.ProjectCommitStatusManager attribute), 97
- obj\_cls (gitlab.v3.objects.ProjectDeploymentManager attribute), 98
- obj\_cls (gitlab.v3.objects.ProjectEnvironmentManager attribute), 99
- obj\_cls (gitlab.v3.objects.ProjectEventManager attribute), 99
- obj\_cls (gitlab.v3.objects.ProjectFileManager attribute), 101
- obj\_cls (gitlab.v3.objects.ProjectForkManager attribute), 101
- obj\_cls (gitlab.v3.objects.ProjectHookManager attribute), 103
- obj\_cls (gitlab.v3.objects.ProjectIssueManager attribute), 105
- obj\_cls (gitlab.v3.objects.ProjectIssueNoteManager attribute), 106
- obj\_cls (gitlab.v3.objects.ProjectKeyManager attribute), 107
- obj\_cls (gitlab.v3.objects.ProjectLabelManager attribute), 109
- obj\_cls (gitlab.v3.objects.ProjectManager attribute), 110
- obj\_cls (gitlab.v3.objects.ProjectMemberManager attribute), 112
- obj\_cls (gitlab.v3.objects.ProjectMergeRequestDiffManager attribute), 115
- obj\_cls (gitlab.v3.objects.ProjectMergeRequestManager attribute), 116
- obj\_cls (gitlab.v3.objects.ProjectMergeRequestNoteManager attribute), 117
- obj\_cls (gitlab.v3.objects.ProjectMilestoneManager attribute), 118
- obj\_cls (gitlab.v3.objects.ProjectNoteManager attribute), 119
- obj\_cls (gitlab.v3.objects.ProjectNotificationSettingsManager attribute), 120
- obj\_cls (gitlab.v3.objects.ProjectPipelineManager attribute), 121
- obj\_cls (gitlab.v3.objects.ProjectRunnerManager attribute), 122
- obj\_cls (gitlab.v3.objects.ProjectServiceManager attribute), 123
- obj\_cls (gitlab.v3.objects.ProjectSnippetManager attribute), 123



- tribute), 124
- obj\_cls (gitlab.v3.objects.ProjectSnippetNoteManager attribute), 125
- obj\_cls (gitlab.v3.objects.ProjectTagManager attribute), 126
- obj\_cls (gitlab.v3.objects.ProjectTriggerManager attribute), 127
- obj\_cls (gitlab.v3.objects.ProjectVariableManager attribute), 128
- obj\_cls (gitlab.v3.objects.RunnerManager attribute), 130
- obj\_cls (gitlab.v3.objects.SnippetManager attribute), 131
- obj\_cls (gitlab.v3.objects.TeamManager attribute), 132
- obj\_cls (gitlab.v3.objects.TeamMemberManager attribute), 133
- obj\_cls (gitlab.v3.objects.TeamProjectManager attribute), 134
- obj\_cls (gitlab.v3.objects.TODOManager attribute), 135
- obj\_cls (gitlab.v3.objects.UserEmailManager attribute), 137
- obj\_cls (gitlab.v3.objects.UserKeyManager attribute), 138
- obj\_cls (gitlab.v3.objects.UserManager attribute), 140
- obj\_cls (gitlab.v3.objects.UserProjectManager attribute), 141
- ObjectDeleteMixin (class in gitlab.mixins), 174
- on\_http\_error() (in module gitlab.exceptions), 171
- optionalCreateAttrs (gitlab.base.GitlabObject attribute), 166
- optionalCreateAttrs (gitlab.v3.objects.BroadcastMessage attribute), 66
- optionalCreateAttrs (gitlab.v3.objects.Group attribute), 72
- optionalCreateAttrs (gitlab.v3.objects.GroupMember attribute), 75
- optionalCreateAttrs (gitlab.v3.objects.Project attribute), 85
- optionalCreateAttrs (gitlab.v3.objects.ProjectCommit attribute), 94
- optionalCreateAttrs (gitlab.v3.objects.ProjectCommitComment attribute), 95
- optionalCreateAttrs (gitlab.v3.objects.ProjectCommitStatus attribute), 96
- optionalCreateAttrs (gitlab.v3.objects.ProjectEnvironment attribute), 98
- optionalCreateAttrs (gitlab.v3.objects.ProjectFile attribute), 100
- optionalCreateAttrs (gitlab.v3.objects.ProjectFork attribute), 101
- optionalCreateAttrs (gitlab.v3.objects.ProjectHook attribute), 102
- optionalCreateAttrs (gitlab.v3.objects.ProjectIssue attribute), 103
- optionalCreateAttrs (gitlab.v3.objects.ProjectIssueNote attribute), 106
- optionalCreateAttrs (gitlab.v3.objects.ProjectLabel attribute), 108
- optionalCreateAttrs (gitlab.v3.objects.ProjectMember attribute), 111
- optionalCreateAttrs (gitlab.v3.objects.ProjectMergeRequest attribute), 114
- optionalCreateAttrs (gitlab.v3.objects.ProjectMilestone attribute), 118
- optionalCreateAttrs (gitlab.v3.objects.ProjectSnippet attribute), 123
- optionalCreateAttrs (gitlab.v3.objects.ProjectTag attribute), 125
- optionalCreateAttrs (gitlab.v3.objects.Snippet attribute), 130
- optionalCreateAttrs (gitlab.v3.objects.User attribute), 136
- optionalCreateAttrs (gitlab.v3.objects.UserProject attribute), 140
- optionalGetAttrs (gitlab.base.GitlabObject attribute), 166
- optionalGetAttrs (gitlab.v3.objects.License attribute), 80
- optionalGetAttrs (gitlab.v3.objects.ProjectCommitStatus attribute), 96
- optionalListAttrs (gitlab.base.GitlabObject attribute), 167
- optionalListAttrs (gitlab.v3.objects.GroupIssue attribute), 73
- optionalListAttrs (gitlab.v3.objects.GroupProject attribute), 77
- optionalListAttrs (gitlab.v3.objects.Issue attribute), 79
- optionalListAttrs (gitlab.v3.objects.License attribute), 80
- optionalListAttrs (gitlab.v3.objects.Namespace attribute), 81
- optionalListAttrs (gitlab.v3.objects.Project attribute), 85
- optionalListAttrs (gitlab.v3.objects.ProjectIssue attribute), 103
- optionalListAttrs (gitlab.v3.objects.ProjectMergeRequest attribute), 114
- optionalListAttrs (gitlab.v3.objects.ProjectMilestone attribute), 118
- optionalListAttrs (gitlab.v3.objects.Runner attribute), 129
- optionalListAttrs (gitlab.v3.objects.TODO attribute), 134
- optionalUpdateAttrs (gitlab.base.GitlabObject attribute), 167
- optionalUpdateAttrs (gitlab.v3.objects.ApplicationSettings attribute), 66
- optionalUpdateAttrs (gitlab.v3.objects.BroadcastMessage attribute), 67
- optionalUpdateAttrs (gitlab.v3.objects.Group attribute), 72
- optionalUpdateAttrs (gitlab.v3.objects.ProjectIssue attribute), 103

- lab.v3.objects.NotificationSettings attribute), 82
  - optionalUpdateAttrs (gitlab.v3.objects.Project attribute), 85
  - optionalUpdateAttrs (gitlab.v3.objects.ProjectEnvironment attribute), 98
  - optionalUpdateAttrs (gitlab.v3.objects.ProjectIssue attribute), 103
  - optionalUpdateAttrs (gitlab.v3.objects.ProjectLabel attribute), 108
  - optionalUpdateAttrs (gitlab.v3.objects.ProjectMergeRequest attribute), 114
  - optionalUpdateAttrs (gitlab.v3.objects.ProjectMilestone attribute), 118
  - optionalUpdateAttrs (gitlab.v3.objects.ProjectSnippet attribute), 123
  - optionalUpdateAttrs (gitlab.v3.objects.Runner attribute), 129
  - optionalUpdateAttrs (gitlab.v3.objects.Snippet attribute), 130
  - optionalUpdateAttrs (gitlab.v3.objects.User attribute), 136
  - owned() (gitlab.v3.objects.ProjectManager method), 110
  - OWNER\_ACCESS (gitlab.v3.objects.Group attribute), 72
- P**
- parent\_attrs (gitlab.base.RESTManager attribute), 167
  - password (gitlab.Gitlab attribute), 186
  - path (gitlab.base.RESTManager attribute), 167
  - pipelines (gitlab.v3.objects.Project attribute), 83
  - play() (gitlab.v3.objects.ProjectBuild method), 93
  - play() (gitlab.v4.objects.ProjectJob method), 153
  - pretty\_print() (gitlab.base.GitlabObject method), 167
  - process\_metrics() (gitlab.v3.objects.SidekiqManager method), 130
  - process\_metrics() (gitlab.v4.objects.SidekiqManager method), 161
  - Project (class in gitlab.v3.objects), 82
  - Project (class in gitlab.v4.objects), 143
  - project\_accessrequests (gitlab.Gitlab attribute), 180
  - project\_board\_lists (gitlab.Gitlab attribute), 180
  - project\_boards (gitlab.Gitlab attribute), 180
  - project\_branches (gitlab.Gitlab attribute), 180
  - project\_builds (gitlab.Gitlab attribute), 180
  - project\_commit\_comments (gitlab.Gitlab attribute), 180
  - project\_commit\_statuses (gitlab.Gitlab attribute), 180
  - project\_commits (gitlab.Gitlab attribute), 180
  - project\_deployments (gitlab.Gitlab attribute), 180
  - project\_environments (gitlab.Gitlab attribute), 181
  - project\_events (gitlab.Gitlab attribute), 181
  - project\_files (gitlab.Gitlab attribute), 181
  - project\_forks (gitlab.Gitlab attribute), 181
  - project\_hooks (gitlab.Gitlab attribute), 181
  - project\_id\_attr (gitlab.v3.objects.ProjectIssue attribute), 104
  - project\_id\_attr (gitlab.v3.objects.ProjectIssueNote attribute), 106
  - project\_issue\_notes (gitlab.Gitlab attribute), 181
  - project\_issues (gitlab.Gitlab attribute), 181
  - project\_keys (gitlab.Gitlab attribute), 181
  - project\_labels (gitlab.Gitlab attribute), 181
  - project\_members (gitlab.Gitlab attribute), 181
  - project\_mergerequest\_diffs (gitlab.Gitlab attribute), 181
  - project\_mergerequest\_notes (gitlab.Gitlab attribute), 181
  - project\_mergerequests (gitlab.Gitlab attribute), 181
  - project\_milestones (gitlab.Gitlab attribute), 181
  - project\_notes (gitlab.Gitlab attribute), 181
  - project\_notificationsettings (gitlab.Gitlab attribute), 182
  - project\_pipelines (gitlab.Gitlab attribute), 182
  - project\_runners (gitlab.Gitlab attribute), 182
  - project\_services (gitlab.Gitlab attribute), 182
  - project\_snippet\_notes (gitlab.Gitlab attribute), 182
  - project\_snippets (gitlab.Gitlab attribute), 182
  - project\_tags (gitlab.Gitlab attribute), 182
  - project\_triggers (gitlab.Gitlab attribute), 182
  - project\_variables (gitlab.Gitlab attribute), 182
  - ProjectAccessRequest (class in gitlab.v3.objects), 88
  - ProjectAccessRequest (class in gitlab.v4.objects), 147
  - ProjectAccessRequestManager (class in gitlab.v3.objects), 89
  - ProjectAccessRequestManager (class in gitlab.v4.objects), 147
  - ProjectBoard (class in gitlab.v3.objects), 89
  - ProjectBoard (class in gitlab.v4.objects), 148
  - ProjectBoardList (class in gitlab.v3.objects), 89
  - ProjectBoardList (class in gitlab.v4.objects), 148
  - ProjectBoardListManager (class in gitlab.v3.objects), 90
  - ProjectBoardListManager (class in gitlab.v4.objects), 148
  - ProjectBoardManager (class in gitlab.v3.objects), 90
  - ProjectBoardManager (class in gitlab.v4.objects), 148
  - ProjectBranch (class in gitlab.v3.objects), 91
  - ProjectBranch (class in gitlab.v4.objects), 148
  - ProjectBranchManager (class in gitlab.v3.objects), 91
  - ProjectBranchManager (class in gitlab.v4.objects), 148
  - ProjectBuild (class in gitlab.v3.objects), 92
  - ProjectBuildManager (class in gitlab.v3.objects), 93
  - ProjectCommit (class in gitlab.v3.objects), 93
  - ProjectCommit (class in gitlab.v4.objects), 148
  - ProjectCommitComment (class in gitlab.v3.objects), 94
  - ProjectCommitComment (class in gitlab.v4.objects), 149
  - ProjectCommitCommentManager (class in gitlab.v3.objects), 95
  - ProjectCommitCommentManager (class in gitlab.v4.objects), 149
  - ProjectCommitManager (class in gitlab.v3.objects), 95

- ProjectCommitManager (class in gitlab.v4.objects), 149
- ProjectCommitStatus (class in gitlab.v3.objects), 96
- ProjectCommitStatus (class in gitlab.v4.objects), 149
- ProjectCommitStatusManager (class in gitlab.v3.objects), 96
- ProjectCommitStatusManager (class in gitlab.v4.objects), 149
- ProjectDeployment (class in gitlab.v3.objects), 97
- ProjectDeployment (class in gitlab.v4.objects), 149
- ProjectDeploymentManager (class in gitlab.v3.objects), 97
- ProjectDeploymentManager (class in gitlab.v4.objects), 149
- ProjectEnvironment (class in gitlab.v3.objects), 98
- ProjectEnvironment (class in gitlab.v4.objects), 149
- ProjectEnvironmentManager (class in gitlab.v3.objects), 98
- ProjectEnvironmentManager (class in gitlab.v4.objects), 150
- ProjectEvent (class in gitlab.v3.objects), 99
- ProjectEvent (class in gitlab.v4.objects), 150
- ProjectEventManager (class in gitlab.v3.objects), 99
- ProjectEventManager (class in gitlab.v4.objects), 150
- ProjectFile (class in gitlab.v3.objects), 99
- ProjectFile (class in gitlab.v4.objects), 150
- ProjectFileManager (class in gitlab.v3.objects), 100
- ProjectFileManager (class in gitlab.v4.objects), 150
- ProjectFork (class in gitlab.v3.objects), 101
- ProjectFork (class in gitlab.v4.objects), 152
- ProjectForkManager (class in gitlab.v3.objects), 101
- ProjectForkManager (class in gitlab.v4.objects), 152
- ProjectHook (class in gitlab.v3.objects), 101
- ProjectHook (class in gitlab.v4.objects), 152
- ProjectHookManager (class in gitlab.v3.objects), 102
- ProjectHookManager (class in gitlab.v4.objects), 152
- ProjectIssue (class in gitlab.v3.objects), 103
- ProjectIssue (class in gitlab.v4.objects), 152
- ProjectIssueManager (class in gitlab.v3.objects), 104
- ProjectIssueManager (class in gitlab.v4.objects), 152
- ProjectIssueNote (class in gitlab.v3.objects), 105
- ProjectIssueNote (class in gitlab.v4.objects), 152
- ProjectIssueNoteManager (class in gitlab.v3.objects), 106
- ProjectIssueNoteManager (class in gitlab.v4.objects), 152
- ProjectJob (class in gitlab.v4.objects), 152
- ProjectJobManager (class in gitlab.v4.objects), 154
- ProjectKey (class in gitlab.v3.objects), 106
- ProjectKey (class in gitlab.v4.objects), 154
- ProjectKeyManager (class in gitlab.v3.objects), 106
- ProjectKeyManager (class in gitlab.v4.objects), 154
- ProjectLabel (class in gitlab.v3.objects), 107
- ProjectLabel (class in gitlab.v4.objects), 154
- ProjectLabelManager (class in gitlab.v3.objects), 108
- ProjectLabelManager (class in gitlab.v4.objects), 154
- ProjectManager (class in gitlab.v3.objects), 109
- ProjectManager (class in gitlab.v4.objects), 155
- ProjectMember (class in gitlab.v3.objects), 111
- ProjectMember (class in gitlab.v4.objects), 155
- ProjectMemberManager (class in gitlab.v3.objects), 111
- ProjectMemberManager (class in gitlab.v4.objects), 155
- ProjectMergeRequest (class in gitlab.v3.objects), 112
- ProjectMergeRequest (class in gitlab.v4.objects), 155
- ProjectMergeRequestDiff (class in gitlab.v3.objects), 114
- ProjectMergeRequestDiff (class in gitlab.v4.objects), 156
- ProjectMergeRequestDiffManager (class in gitlab.v3.objects), 115
- ProjectMergeRequestDiffManager (class in gitlab.v4.objects), 156
- ProjectMergeRequestManager (class in gitlab.v3.objects), 115
- ProjectMergeRequestManager (class in gitlab.v4.objects), 156
- ProjectMergeRequestNote (class in gitlab.v3.objects), 116
- ProjectMergeRequestNote (class in gitlab.v4.objects), 156
- ProjectMergeRequestNoteManager (class in gitlab.v3.objects), 116
- ProjectMergeRequestNoteManager (class in gitlab.v4.objects), 156
- ProjectMilestone (class in gitlab.v3.objects), 117
- ProjectMilestone (class in gitlab.v4.objects), 156
- ProjectMilestoneManager (class in gitlab.v3.objects), 118
- ProjectMilestoneManager (class in gitlab.v4.objects), 157
- ProjectNote (class in gitlab.v3.objects), 119
- ProjectNote (class in gitlab.v4.objects), 157
- ProjectNoteManager (class in gitlab.v3.objects), 119
- ProjectNoteManager (class in gitlab.v4.objects), 157
- ProjectNotificationSettings (class in gitlab.v3.objects), 119
- ProjectNotificationSettings (class in gitlab.v4.objects), 157
- ProjectNotificationSettingsManager (class in gitlab.v3.objects), 120
- ProjectNotificationSettingsManager (class in gitlab.v4.objects), 157
- ProjectPipeline (class in gitlab.v3.objects), 120
- ProjectPipeline (class in gitlab.v4.objects), 157
- ProjectPipelineManager (class in gitlab.v3.objects), 121
- ProjectPipelineManager (class in gitlab.v4.objects), 157
- ProjectProtectedBranch (class in gitlab.v4.objects), 158
- ProjectProtectedBranchManager (class in gitlab.v4.objects), 158
- ProjectRunner (class in gitlab.v3.objects), 121
- ProjectRunner (class in gitlab.v4.objects), 158
- ProjectRunnerManager (class in gitlab.v3.objects), 121
- ProjectRunnerManager (class in gitlab.v4.objects), 158
- projects (gitlab.Gitlab attribute), 182
- projects (gitlab.v3.objects.Group attribute), 71

projects (gitlab.v3.objects.User attribute), 135  
 ProjectService (class in gitlab.v3.objects), 122  
 ProjectService (class in gitlab.v4.objects), 158  
 ProjectServiceManager (class in gitlab.v3.objects), 122  
 ProjectServiceManager (class in gitlab.v4.objects), 158  
 ProjectSnippet (class in gitlab.v3.objects), 123  
 ProjectSnippet (class in gitlab.v4.objects), 159  
 ProjectSnippetManager (class in gitlab.v3.objects), 123  
 ProjectSnippetManager (class in gitlab.v4.objects), 159  
 ProjectSnippetNote (class in gitlab.v3.objects), 124  
 ProjectSnippetNote (class in gitlab.v4.objects), 159  
 ProjectSnippetNoteManager (class in gitlab.v3.objects), 124  
 ProjectSnippetNoteManager (class in gitlab.v4.objects), 159  
 ProjectTag (class in gitlab.v3.objects), 125  
 ProjectTag (class in gitlab.v4.objects), 159  
 ProjectTagManager (class in gitlab.v3.objects), 126  
 ProjectTagManager (class in gitlab.v4.objects), 160  
 ProjectTagRelease (class in gitlab.v3.objects), 126  
 ProjectTrigger (class in gitlab.v3.objects), 127  
 ProjectTrigger (class in gitlab.v4.objects), 160  
 ProjectTriggerManager (class in gitlab.v3.objects), 127  
 ProjectTriggerManager (class in gitlab.v4.objects), 160  
 ProjectVariable (class in gitlab.v3.objects), 127  
 ProjectVariable (class in gitlab.v4.objects), 160  
 ProjectVariableManager (class in gitlab.v3.objects), 128  
 ProjectVariableManager (class in gitlab.v4.objects), 160  
 protect() (gitlab.v3.objects.ProjectBranch method), 91  
 protect() (gitlab.v4.objects.ProjectBranch method), 148  
 public() (gitlab.v3.objects.SnippetManager method), 131  
 public() (gitlab.v4.objects.SnippetManager method), 162

## Q

queue\_metrics() (gitlab.v3.objects.SidekiqManager method), 130  
 queue\_metrics() (gitlab.v4.objects.SidekiqManager method), 161

## R

raise\_error\_from\_response() (in module gitlab.exceptions), 171  
 raw() (gitlab.v3.objects.Snippet method), 130  
 raw() (gitlab.v4.objects.ProjectFileManager method), 151  
 register\_custom\_action() (in module gitlab.cli), 168  
 REPORTER\_ACCESS (gitlab.v3.objects.Group attribute), 72  
 repository\_archive() (gitlab.v3.objects.Project method), 85  
 repository\_archive() (gitlab.v4.objects.Project method), 144  
 repository\_blob() (gitlab.v3.objects.Project method), 85  
 repository\_blob() (gitlab.v4.objects.Project method), 144

repository\_compare() (gitlab.v3.objects.Project method), 86  
 repository\_compare() (gitlab.v4.objects.Project method), 145  
 repository\_contributors() (gitlab.v3.objects.Project method), 86  
 repository\_contributors() (gitlab.v4.objects.Project method), 145  
 repository\_raw\_blob() (gitlab.v3.objects.Project method), 86  
 repository\_raw\_blob() (gitlab.v4.objects.Project method), 145  
 repository\_tree() (gitlab.v3.objects.Project method), 86  
 repository\_tree() (gitlab.v4.objects.Project method), 146  
 requiredCreateAttrs (gitlab.base.GitlabObject attribute), 167  
 requiredCreateAttrs (gitlab.v3.objects.BroadcastMessage attribute), 67  
 requiredCreateAttrs (gitlab.v3.objects.CurrentUserEmail attribute), 68  
 requiredCreateAttrs (gitlab.v3.objects.CurrentUserKey attribute), 69  
 requiredCreateAttrs (gitlab.v3.objects.Group attribute), 72  
 requiredCreateAttrs (gitlab.v3.objects.GroupMember attribute), 75  
 requiredCreateAttrs (gitlab.v3.objects.Hook attribute), 78  
 requiredCreateAttrs (gitlab.v3.objects.Project attribute), 87  
 requiredCreateAttrs (gitlab.v3.objects.ProjectBoardList attribute), 90  
 requiredCreateAttrs (gitlab.v3.objects.ProjectBranch attribute), 91  
 requiredCreateAttrs (gitlab.v3.objects.ProjectCommit attribute), 94  
 requiredCreateAttrs (gitlab.v3.objects.ProjectCommitComment attribute), 95  
 requiredCreateAttrs (gitlab.v3.objects.ProjectCommitStatus attribute), 96  
 requiredCreateAttrs (gitlab.v3.objects.ProjectEnvironment attribute), 98  
 requiredCreateAttrs (gitlab.v3.objects.ProjectFile attribute), 100  
 requiredCreateAttrs (gitlab.v3.objects.ProjectHook attribute), 102  
 requiredCreateAttrs (gitlab.v3.objects.ProjectIssue attribute), 104  
 requiredCreateAttrs (gitlab.v3.objects.ProjectIssueNote attribute), 106  
 requiredCreateAttrs (gitlab.v3.objects.ProjectKey attribute), 106

- requiredCreateAttrs (gitlab.v3.objects.ProjectLabel attribute), 108
- requiredCreateAttrs (gitlab.v3.objects.ProjectMember attribute), 111
- requiredCreateAttrs (gitlab.v3.objects.ProjectMergeRequest attribute), 114
- requiredCreateAttrs (gitlab.v3.objects.ProjectMergeRequestNote attribute), 116
- requiredCreateAttrs (gitlab.v3.objects.ProjectMilestone attribute), 118
- requiredCreateAttrs (gitlab.v3.objects.ProjectNote attribute), 119
- requiredCreateAttrs (gitlab.v3.objects.ProjectPipeline attribute), 120
- requiredCreateAttrs (gitlab.v3.objects.ProjectRunner attribute), 121
- requiredCreateAttrs (gitlab.v3.objects.ProjectSnippet attribute), 123
- requiredCreateAttrs (gitlab.v3.objects.ProjectSnippetNote attribute), 124
- requiredCreateAttrs (gitlab.v3.objects.ProjectTag attribute), 125
- requiredCreateAttrs (gitlab.v3.objects.ProjectTagRelease attribute), 127
- requiredCreateAttrs (gitlab.v3.objects.ProjectVariable attribute), 128
- requiredCreateAttrs (gitlab.v3.objects.Snippet attribute), 131
- requiredCreateAttrs (gitlab.v3.objects.Team attribute), 132
- requiredCreateAttrs (gitlab.v3.objects.TeamMember attribute), 133
- requiredCreateAttrs (gitlab.v3.objects.TeamProject attribute), 133
- requiredCreateAttrs (gitlab.v3.objects.User attribute), 136
- requiredCreateAttrs (gitlab.v3.objects.UserEmail attribute), 136
- requiredCreateAttrs (gitlab.v3.objects.UserKey attribute), 137
- requiredCreateAttrs (gitlab.v3.objects.UserProject attribute), 140
- requiredDeleteAttrs (gitlab.base.GitlabObject attribute), 167
- requiredDeleteAttrs (gitlab.v3.objects.ProjectFile attribute), 100
- requiredDeleteAttrs (gitlab.v3.objects.ProjectLabel attribute), 108
- requiredGetAttrs (gitlab.base.GitlabObject attribute), 167
- requiredGetAttrs (gitlab.v3.objects.ProjectFile attribute), 100
- requiredListAttrs (gitlab.base.GitlabObject attribute), 167
- requiredUpdateAttrs (gitlab.base.GitlabObject attribute), 167
- requiredUpdateAttrs (gitlab.v3.objects.BroadcastMessage attribute), 67
- requiredUpdateAttrs (gitlab.v3.objects.GroupMember attribute), 75
- requiredUpdateAttrs (gitlab.v3.objects.ProjectBoardList attribute), 90
- requiredUpdateAttrs (gitlab.v3.objects.ProjectLabel attribute), 108
- requiredUpdateAttrs (gitlab.v3.objects.ProjectMember attribute), 111
- requiredUpdateAttrs (gitlab.v3.objects.User attribute), 136
- requiredUrlAttrs (gitlab.base.GitlabObject attribute), 167
- requiredUrlAttrs (gitlab.v3.objects.GroupIssue attribute), 73
- requiredUrlAttrs (gitlab.v3.objects.GroupMember attribute), 75
- requiredUrlAttrs (gitlab.v3.objects.GroupNotificationSettings attribute), 77
- requiredUrlAttrs (gitlab.v3.objects.ProjectBoard attribute), 89
- requiredUrlAttrs (gitlab.v3.objects.ProjectBoardList attribute), 90
- requiredUrlAttrs (gitlab.v3.objects.ProjectBranch attribute), 91
- requiredUrlAttrs (gitlab.v3.objects.ProjectBuild attribute), 93
- requiredUrlAttrs (gitlab.v3.objects.ProjectCommit attribute), 94
- requiredUrlAttrs (gitlab.v3.objects.ProjectCommitComment attribute), 95
- requiredUrlAttrs (gitlab.v3.objects.ProjectCommitStatus attribute), 96
- requiredUrlAttrs (gitlab.v3.objects.ProjectEnvironment attribute), 98
- requiredUrlAttrs (gitlab.v3.objects.ProjectEvent attribute), 99
- requiredUrlAttrs (gitlab.v3.objects.ProjectFile attribute), 100
- requiredUrlAttrs (gitlab.v3.objects.ProjectFork attribute), 101
- requiredUrlAttrs (gitlab.v3.objects.ProjectHook attribute), 102
- requiredUrlAttrs (gitlab.v3.objects.ProjectIssue attribute), 104
- requiredUrlAttrs (gitlab.v3.objects.ProjectIssueNote attribute), 106
- requiredUrlAttrs (gitlab.v3.objects.ProjectKey attribute), 106
- requiredUrlAttrs (gitlab.v3.objects.ProjectLabel attribute), 108

- requiredUrlAttrs (gitlab.v3.objects.ProjectMember attribute), 111
  - requiredUrlAttrs (gitlab.v3.objects.ProjectMergeRequest attribute), 114
  - requiredUrlAttrs (gitlab.v3.objects.ProjectMergeRequestDiff attribute), 115
  - requiredUrlAttrs (gitlab.v3.objects.ProjectMergeRequestNote attribute), 116
  - requiredUrlAttrs (gitlab.v3.objects.ProjectMilestone attribute), 118
  - requiredUrlAttrs (gitlab.v3.objects.ProjectNote attribute), 119
  - requiredUrlAttrs (gitlab.v3.objects.ProjectNotificationSettings attribute), 120
  - requiredUrlAttrs (gitlab.v3.objects.ProjectPipeline attribute), 120
  - requiredUrlAttrs (gitlab.v3.objects.ProjectService attribute), 122
  - requiredUrlAttrs (gitlab.v3.objects.ProjectSnippet attribute), 123
  - requiredUrlAttrs (gitlab.v3.objects.ProjectSnippetNote attribute), 124
  - requiredUrlAttrs (gitlab.v3.objects.ProjectTag attribute), 125
  - requiredUrlAttrs (gitlab.v3.objects.ProjectTagRelease attribute), 127
  - requiredUrlAttrs (gitlab.v3.objects.ProjectTrigger attribute), 127
  - requiredUrlAttrs (gitlab.v3.objects.ProjectVariable attribute), 128
  - requiredUrlAttrs (gitlab.v3.objects.TeamMember attribute), 133
  - requiredUrlAttrs (gitlab.v3.objects.TeamProject attribute), 133
  - requiredUrlAttrs (gitlab.v3.objects.UserEmail attribute), 136
  - requiredUrlAttrs (gitlab.v3.objects.UserKey attribute), 137
  - requiredUrlAttrs (gitlab.v3.objects.UserProject attribute), 140
  - reset\_spent\_time() (gitlab.mixins.TimeTrackingMixin method), 175
  - reset\_spent\_time() (gitlab.v3.objects.ProjectIssue method), 104
  - reset\_spent\_time() (gitlab.v3.objects.ProjectMergeRequest method), 114
  - reset\_time\_estimate() (gitlab.mixins.TimeTrackingMixin method), 175
  - reset\_time\_estimate() (gitlab.v3.objects.ProjectIssue method), 104
  - reset\_time\_estimate() (gitlab.v3.objects.ProjectMergeRequest method), 114
  - response\_content() (in module gitlab.utils), 177
  - RESTManager (class in gitlab.base), 167
  - RESTObject (class in gitlab.base), 167
  - RESTObjectList (class in gitlab.base), 168
  - RetrieveMixin (class in gitlab.mixins), 174
  - retry() (gitlab.v3.objects.ProjectBuild method), 93
  - retry() (gitlab.v3.objects.ProjectPipeline method), 120
  - retry() (gitlab.v4.objects.ProjectJob method), 153
  - retry() (gitlab.v4.objects.ProjectPipeline method), 157
  - Runner (class in gitlab.v3.objects), 128
  - Runner (class in gitlab.v4.objects), 160
  - RunnerManager (class in gitlab.v3.objects), 129
  - RunnerManager (class in gitlab.v4.objects), 160
  - runners (gitlab.Gitlab attribute), 182
  - runners (gitlab.v3.objects.Project attribute), 83
- ## S
- save() (gitlab.base.GitlabObject method), 165, 167
  - save() (gitlab.mixins.SaveMixin method), 174
  - save() (gitlab.v3.objects.ApplicationSettings method), 65
  - save() (gitlab.v3.objects.BroadcastMessage method), 66
  - save() (gitlab.v3.objects.Group method), 71
  - save() (gitlab.v3.objects.GroupMember method), 75
  - save() (gitlab.v3.objects.GroupNotificationSettings method), 76
  - save() (gitlab.v3.objects.NotificationSettings method), 81
  - save() (gitlab.v3.objects.Project method), 84
  - save() (gitlab.v3.objects.ProjectBoardList method), 90
  - save() (gitlab.v3.objects.ProjectEnvironment method), 98
  - save() (gitlab.v3.objects.ProjectFile method), 99
  - save() (gitlab.v3.objects.ProjectHook method), 101
  - save() (gitlab.v3.objects.ProjectIssue method), 103
  - save() (gitlab.v3.objects.ProjectIssueNote method), 105
  - save() (gitlab.v3.objects.ProjectLabel method), 107
  - save() (gitlab.v3.objects.ProjectMember method), 111
  - save() (gitlab.v3.objects.ProjectMergeRequest method), 112
  - save() (gitlab.v3.objects.ProjectMergeRequestNote method), 116
  - save() (gitlab.v3.objects.ProjectMilestone method), 117
  - save() (gitlab.v3.objects.ProjectNotificationSettings method), 119
  - save() (gitlab.v3.objects.ProjectService method), 122
  - save() (gitlab.v3.objects.ProjectSnippet method), 123
  - save() (gitlab.v3.objects.ProjectTagRelease method), 126
  - save() (gitlab.v3.objects.ProjectVariable method), 127
  - save() (gitlab.v3.objects.Runner method), 129
  - save() (gitlab.v3.objects.Snippet method), 130
  - save() (gitlab.v3.objects.User method), 135
  - save() (gitlab.v4.objects.ProjectFile method), 150
  - save() (gitlab.v4.objects.ProjectLabel method), 154
  - SaveMixin (class in gitlab.mixins), 174
  - search() (gitlab.v3.objects.GroupManager method), 75
  - search() (gitlab.v3.objects.ProjectManager method), 111

- search() (gitlab.v3.objects.UserManager method), 140
- services (gitlab.v3.objects.Project attribute), 83
- session (gitlab.Gitlab attribute), 186
- set\_credentials() (gitlab.Gitlab method), 186
- set\_release\_description() (gitlab.v3.objects.ProjectTag method), 125
- set\_release\_description() (gitlab.v4.objects.ProjectTag method), 159
- set\_token() (gitlab.Gitlab method), 186
- set\_url() (gitlab.Gitlab method), 186
- settings (gitlab.Gitlab attribute), 182
- share() (gitlab.v3.objects.Project method), 87
- share() (gitlab.v4.objects.Project method), 146
- short\_print() (gitlab.base.GitlabObject method), 167
- shortPrintAttr (gitlab.base.GitlabObject attribute), 167
- shortPrintAttr (gitlab.v3.objects.CurrentUser attribute), 68
- shortPrintAttr (gitlab.v3.objects.CurrentUserEmail attribute), 68
- shortPrintAttr (gitlab.v3.objects.CurrentUserKey attribute), 69
- shortPrintAttr (gitlab.v3.objects.Group attribute), 72
- shortPrintAttr (gitlab.v3.objects.GroupMember attribute), 75
- shortPrintAttr (gitlab.v3.objects.Hook attribute), 78
- shortPrintAttr (gitlab.v3.objects.Issue attribute), 79
- shortPrintAttr (gitlab.v3.objects.Project attribute), 87
- shortPrintAttr (gitlab.v3.objects.ProjectCommit attribute), 94
- shortPrintAttr (gitlab.v3.objects.ProjectEvent attribute), 99
- shortPrintAttr (gitlab.v3.objects.ProjectFile attribute), 100
- shortPrintAttr (gitlab.v3.objects.ProjectHook attribute), 102
- shortPrintAttr (gitlab.v3.objects.ProjectIssue attribute), 104
- shortPrintAttr (gitlab.v3.objects.ProjectMember attribute), 111
- shortPrintAttr (gitlab.v3.objects.ProjectMilestone attribute), 118
- shortPrintAttr (gitlab.v3.objects.ProjectSnippet attribute), 123
- shortPrintAttr (gitlab.v3.objects.ProjectTag attribute), 125
- shortPrintAttr (gitlab.v3.objects.ProjectTagRelease attribute), 127
- shortPrintAttr (gitlab.v3.objects.Snippet attribute), 131
- shortPrintAttr (gitlab.v3.objects.Team attribute), 132
- shortPrintAttr (gitlab.v3.objects.TeamMember attribute), 133
- shortPrintAttr (gitlab.v3.objects.TeamProject attribute), 133
- shortPrintAttr (gitlab.v3.objects.User attribute), 136
- shortPrintAttr (gitlab.v3.objects.UserEmail attribute), 136
- SidekiqManager (class in gitlab.v3.objects), 130
- SidekiqManager (class in gitlab.v4.objects), 161
- Snippet (class in gitlab.v3.objects), 130
- Snippet (class in gitlab.v4.objects), 161
- SnippetManager (class in gitlab.v3.objects), 131
- SnippetManager (class in gitlab.v4.objects), 162
- snippets (gitlab.Gitlab attribute), 182
- snippets (gitlab.v3.objects.Project attribute), 83
- ssl\_verify (gitlab.Gitlab attribute), 187
- star() (gitlab.v3.objects.Project method), 87
- star() (gitlab.v4.objects.Project method), 146
- starred() (gitlab.v3.objects.ProjectManager method), 111
- SubscribableMixin (class in gitlab.mixins), 175
- subscribe() (gitlab.mixins.SubscribableMixin method), 175
- subscribe() (gitlab.v3.objects.ProjectIssue method), 104
- subscribe() (gitlab.v3.objects.ProjectLabel method), 108
- subscribe() (gitlab.v3.objects.ProjectMergeRequest method), 114
- ## T
- tags (gitlab.v3.objects.Project attribute), 83
- take\_ownership() (gitlab.v4.objects.ProjectTrigger method), 160
- Team (class in gitlab.v3.objects), 132
- team\_members (gitlab.Gitlab attribute), 182
- team\_projects (gitlab.Gitlab attribute), 182
- TeamManager (class in gitlab.v3.objects), 132
- TeamMember (class in gitlab.v3.objects), 132
- TeamMemberManager (class in gitlab.v3.objects), 133
- TeamProject (class in gitlab.v3.objects), 133
- TeamProjectManager (class in gitlab.v3.objects), 133
- teams (gitlab.Gitlab attribute), 182
- time\_estimate() (gitlab.mixins.TimeTrackingMixin method), 176
- time\_estimate() (gitlab.v3.objects.ProjectIssue method), 104
- time\_estimate() (gitlab.v3.objects.ProjectMergeRequest method), 114
- time\_stats() (gitlab.mixins.TimeTrackingMixin method), 176
- time\_stats() (gitlab.v3.objects.ProjectIssue method), 104
- time\_stats() (gitlab.v3.objects.ProjectMergeRequest method), 114
- timeout (gitlab.Gitlab attribute), 187
- TimeTrackingMixin (class in gitlab.mixins), 175
- Todo (class in gitlab.v3.objects), 134
- Todo (class in gitlab.v4.objects), 162
- todo() (gitlab.mixins.TODOMixin method), 176
- todo() (gitlab.v3.objects.ProjectIssue method), 104
- todo() (gitlab.v3.objects.ProjectMergeRequest method), 114
- TodoManager (class in gitlab.v3.objects), 134

TodoManager (class in gitlab.v4.objects), 162  
TodoMixin (class in gitlab.mixins), 176  
todos (gitlab.Gitlab attribute), 182  
token\_auth() (gitlab.Gitlab method), 187  
trace() (gitlab.v3.objects.ProjectBuild method), 93  
trace() (gitlab.v4.objects.ProjectJob method), 154  
transfer\_project() (gitlab.v3.objects.Group method), 72  
transfer\_project() (gitlab.v4.objects.Group method), 142  
trigger\_build() (gitlab.v3.objects.Project method), 87  
trigger\_pipeline() (gitlab.v4.objects.Project method), 146  
triggers (gitlab.v3.objects.Project attribute), 83

## U

unarchive() (gitlab.v3.objects.Project method), 87  
unarchive() (gitlab.v4.objects.Project method), 147  
unarchive\_() (gitlab.v3.objects.Project method), 88  
unblock() (gitlab.v3.objects.User method), 136  
unblock() (gitlab.v4.objects.User method), 163  
unprotect() (gitlab.v3.objects.ProjectBranch method), 91  
unprotect() (gitlab.v4.objects.ProjectBranch method), 148  
unstar() (gitlab.v3.objects.Project method), 88  
unstar() (gitlab.v4.objects.Project method), 147  
unsubscribe() (gitlab.mixins.SubscribableMixin method), 175  
unsubscribe() (gitlab.v3.objects.ProjectIssue method), 104  
unsubscribe() (gitlab.v3.objects.ProjectLabel method), 108  
unsubscribe() (gitlab.v3.objects.ProjectMergeRequest method), 114  
update() (gitlab.Gitlab method), 187  
update() (gitlab.mixins.UpdateMixin method), 176  
update() (gitlab.v4.objects.ProjectServiceManager method), 159  
UpdateMixin (class in gitlab.mixins), 176  
upload() (gitlab.v3.objects.Project method), 88  
upload() (gitlab.v4.objects.Project method), 147  
User (class in gitlab.v3.objects), 135  
User (class in gitlab.v4.objects), 163  
user\_emails (gitlab.Gitlab attribute), 183  
user\_keys (gitlab.Gitlab attribute), 183  
user\_projects (gitlab.Gitlab attribute), 183  
UserEmail (class in gitlab.v3.objects), 136  
UserEmail (class in gitlab.v4.objects), 163  
UserEmailManager (class in gitlab.v3.objects), 136  
UserEmailManager (class in gitlab.v4.objects), 163  
UserKey (class in gitlab.v3.objects), 137  
UserKey (class in gitlab.v4.objects), 163  
UserKeyManager (class in gitlab.v3.objects), 137  
UserKeyManager (class in gitlab.v4.objects), 163  
UserManager (class in gitlab.v3.objects), 138  
UserManager (class in gitlab.v4.objects), 163  
UserProject (class in gitlab.v3.objects), 140  
UserProject (class in gitlab.v4.objects), 163

UserProjectManager (class in gitlab.v3.objects), 140  
UserProjectManager (class in gitlab.v4.objects), 163  
users (gitlab.Gitlab attribute), 183

## V

variables (gitlab.v3.objects.Project attribute), 83  
version() (gitlab.Gitlab method), 187  
VISIBILITY\_INTERNAL (gitlab.v3.objects.Group attribute), 72  
VISIBILITY\_INTERNAL (gitlab.v3.objects.Project attribute), 84  
VISIBILITY\_PRIVATE (gitlab.v3.objects.Group attribute), 72  
VISIBILITY\_PRIVATE (gitlab.v3.objects.Project attribute), 84  
VISIBILITY\_PUBLIC (gitlab.v3.objects.Group attribute), 72  
VISIBILITY\_PUBLIC (gitlab.v3.objects.Project attribute), 84

## W

what\_to\_cls() (in module gitlab.cli), 168