

NAME

GitLab::API::v4 – A complete GitLab API v4 client.

SYNOPSIS

```
use GitLab::API::v4;

my $api = GitLab::API::v4->new(
    url      => $v4_api_url,
    private_token => $token,
);

my $branches = $api->branches( $project_id );
```

DESCRIPTION

This module provides a one-to-one interface with the GitLab API v4. Much is not documented here as it would just be duplicating GitLab's own API Documentation <<http://doc.gitlab.com/ce/api/README.html>>.

Note that this distribution also includes the gitlab-api-v4 command-line interface (CLI).

Upgrading

If you are upgrading from GitLab::API::v3 make sure you read:

<https://docs.gitlab.com/ce/api/v3_to_v4.html>

Also, review the Changes file included in the distribution as it outlines the changes made to convert the v3 module to v4:

<<https://github.com/bluefeet/GitLab-API-v4/blob/master/Changes>>

Finally, be aware that many methods were added, removed, renamed, and/or altered. If you want to review exactly what was changed you can use GitHub's compare tool:

<<https://github.com/bluefeet/GitLab-API-v4/compare/72e384775c9570f60f8ef68dee3a1eecd347fb69...master>>

Or clone the repo and run this command:

```
git diff 72e384775c9570f60f8ef68dee3a1eecd347fb69..HEAD --
author/sections/
```

Credentials

Authentication credentials may be defined by setting either the "access_token" or "private_token" arguments.

If no credentials are supplied then the client will be anonymous and greatly limited in what it can do with the API.

Extra care has been taken to hide the token arguments behind closures. This way, if you dump your api object, your tokens won't accidentally leak into places you don't want them to.

Constants

The GitLab API, in rare cases, uses a hard-coded value to represent a state. To make life easier the GitLab::API::v4::Constants module exposes these states as named variables.

Exceptions

The API methods will all throw a useful exception if an unsuccessful response is received from the API. That is except for GET requests that return a 404 response – these will return undef for methods that return a value.

If you'd like to catch and handle these exceptions consider using Try::Tiny.

Logging

This module uses Log::Any and produces some debug messages here and there, but the most useful bits are the info messages produced just before each API call.

Project ID

Note that many API calls require a `$project_id`. This can be specified as a numeric project ID or, in many cases, maybe all cases, as a `NAMESPACE_PATH/PROJECT_PATH` string. The GitLab documentation on this point is vague.

REQUIRED ARGUMENTS**url**

The URL to your v4 API endpoint. Typically this will be something like `https://git.example.com/api/v4`.

OPTIONAL ARGUMENTS**access_token**

A GitLab API OAuth2 token. If set then “private_token” may not be set.

See <<https://docs.gitlab.com/ce/api/#oauth2-tokens>>.

private_token

A GitLab API personal token. If set then “access_token” may not be set.

See <<https://docs.gitlab.com/ce/api/#personal-access-tokens>>.

retries

The number of times the request should be retried in case it fails (5XX HTTP response code). Defaults to 0 (false), meaning that a failed request will not be retried.

sudo_user

The user to execute API calls as. You may find it more useful to use the “sudo” method instead.

See <<https://docs.gitlab.com/ce/api/#sudo>>.

rest_client

An instance of `GitLab::API::v4::RESTClient` (or whatever “rest_client_class” is set to). Typically you will not be setting this as it defaults to a new instance and customization should not be necessary.

rest_client_class

The class to use when constructing the “rest_client”. Defaults to `GitLab::API::v4::RESTClient`.

UTILITY METHODS**paginator**

```
my $paginator = $api->paginator( $method, @method_args );

my $members = $api->paginator('group_members', $group_id);
while (my $member = $members->next()) {
    ...
}

my $users_pager = $api->paginator('users');
while (my $users = $users_pager->next_page()) {
    ...
}

my $all_open_issues = $api->paginator(
    'issues',
    $project_id,
    { state=>'opened' },
)->all();
```

Given a method who supports the `page` and `per_page` parameters, and returns an array ref, this will return a `GitLab::API::v4::Paginator` object that will allow you to walk the records one page or one record at a time.

sudo

```
$api->sudo('fred')->create_issue(...);
```

Returns a new instance of GitLab::API::v4 with the “sudo_user” argument set.

See <<https://docs.gitlab.com/ce/api/#sudo>>.

API METHODS**Award Emoji**

See <https://docs.gitlab.com/ce/api/award_emoji.html>.

issue_award_emojis

```
my $award_emojis = $api->issue_award_emojis(
    $project_id,
    $issue_iid,
    \%params,
);
```

Sends a GET request to `projects/:project_id/issues/:issue_iid/award_emoji` and returns the decoded response content.

merge_request_award_emojis

```
my $award_emojis = $api->merge_request_award_emojis(
    $project_id,
    $merge_request_iid,
    \%params,
);
```

Sends a GET request to `projects/:project_id/merge_requests/:merge_request_iid/award_emoji` and returns the decoded response content.

snippet_award_emojis

```
my $award_emojis = $api->snippet_award_emojis(
    $project_id,
    $merge_request_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/merge_requests/:merge_request_id/award_emoji` and returns the decoded response content.

issue_award_emoji

```
my $award_emoji = $api->issue_award_emoji(
    $project_id,
    $issue_iid,
    $award_id,
);
```

Sends a GET request to `projects/:project_id/issues/:issue_iid/award_emoji/:award_id` and returns the decoded response content.

merge_request_award_emoji

```
my $award_emoji = $api->merge_request_award_emoji(
    $project_id,
    $merge_request_iid,
    $award_id,
);
```

Sends a GET request to

projects/:project_id/merge_requests/:merge_request_iid/award_emoji/:award_id and returns the decoded response content.

snippet_award_emoji

```
my $award_emoji = $api->snippet_award_emoji(
    $project_id,
    $snippet_id,
    $award_id,
);
```

Sends a GET request to projects/:project_id/snippets/:snippet_id/award_emoji/:award_id and returns the decoded response content.

create_issue_award_emoji

```
my $award_emoji = $api->create_issue_award_emoji(
    $project_id,
    $issue_iid,
    \%params,
);
```

Sends a POST request to projects/:project_id/issues/:issue_iid/award_emoji and returns the decoded response content.

create_merge_request_award_emoji

```
my $award_emoji = $api->create_merge_request_award_emoji(
    $project_id,
    $merge_request_iid,
    \%params,
);
```

Sends a POST request to projects/:project_id/merge_requests/:merge_request_iid/award_emoji and returns the decoded response content.

create_snippet_award_emoji

```
my $award_emoji = $api->create_snippet_award_emoji(
    $project_id,
    $snippet_id,
);
```

Sends a POST request to projects/:project_id/snippets/:snippet_id/award_emoji and returns the decoded response content.

delete_issue_award_emoji

```
my $award_emoji = $api->delete_issue_award_emoji(
    $project_id,
    $issue_id,
    $award_id,
);
```

Sends a DELETE request to projects/:project_id/issues/:issue_id/award_emoji/:award_id and returns the decoded response content.

delete_merge_request_award_emoji

```
my $award_emoji = $api->delete_merge_request_emoji(
    $project_id,
    $merge_request_id,
    $award_id,
);
```

Sends a DELETE request to
 projects/:project_id/merge_requests/:merge_request_id/award_emoji/:award_id
 and returns the decoded response content.

delete_snippet_emoji

```
my $award_emoji = $api->delete_snippet_emoji(
    $project_id,
    $snippet_id,
    $award_id,
);
```

Sends a DELETE request to
 projects/:project_id/snippets/:snippet_id/award_emoji/:award_id and
 returns the decoded response content.

issue_note_award_emojis

```
my $award_emojis = $api->issue_note_award_emojis(
    $project_id,
    $issue_iid,
    $note_id,
);
```

Sends a GET request to
 projects/:project_id/issues/:issue_iid/notes/:note_id/award_emoji and
 returns the decoded response content.

issue_note_award_emoji

```
my $award_emoji = $api->issue_note_award_emoji(
    $project_id,
    $issue_iid,
    $note_id,
    $award_id,
);
```

Sends a GET request to
 projects/:project_id/issues/:issue_iid/notes/:note_id/award_emoji/:award_id
 and returns the decoded response content.

create_issue_note_award_emoji

```
my $award_emoji = $api->create_issue_note_award_emoji(
    $project_id,
    $issue_iid,
    $note_id,
    \%params,
);
```

Sends a POST request to
 projects/:project_id/issues/:issue_iid/notes/:note_id/award_emoji and
 returns the decoded response content.

delete_issue_note_award_emoji

```
my $award_emoji = $api->delete_issue_note_award_emoji(
    $project_id,
    $issue_iid,
    $note_id,
    $award_id,
);
```

Sends a DELETE request to
 projects/:project_id/issues/:issue_iid/notes/:note_id/award_emoji/:award_id
 and returns the decoded response content.

merge_request_note_award_emojis

```
my $award_emojis = $api->merge_request_note_award_emojis(
    $project_id,
    $merge_request_iid,
    $note_id,
);
```

Sends a GET request to
 projects/:project_id/merge_requests/:merge_request_iid/notes/:note_id/award_emojis/:award_id
 and returns the decoded response content.

merge_request_note_award_emoji

```
my $award_emoji = $api->merge_request_note_award_emoji(
    $project_id,
    $merge_request_iid,
    $note_id,
    $award_id,
);
```

Sends a GET request to
 projects/:project_id/merge_requests/:merge_request_iid/notes/:note_id/award_emoji/:award_id
 and returns the decoded response content.

create_merge_request_note_award_emoji

```
my $award_emoji = $api->create_merge_request_note_award_emoji(
    $project_id,
    $merge_request_iid,
    $note_id,
    \%params,
);
```

Sends a POST request to
 projects/:project_id/merge_requests/:merge_request_iid/notes/:note_id/award_emojis/:award_id
 and returns the decoded response content.

delete_merge_request_note_award_emoji

```
my $award_emoji = $api->delete_merge_request_note_award_emoji(
    $project_id,
    $merge_request_iid,
    $note_id,
    $award_id,
);
```

Sends a DELETE request to
 projects/:project_id/merge_requests/:merge_request_iid/notes/:note_id/award_emoji/:award_id
 and returns the decoded response content.

Branches

See <<https://docs.gitlab.com/ce/api/branches.html>>.

branches

```
my $branches = $api->branches (
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/repository/branches` and returns the decoded response content.

branch

```
my $branch = $api->branch (
    $project_id,
    $branch_name,
);
```

Sends a GET request to `projects/:project_id/repository/branches/:branch_name` and returns the decoded response content.

create_branch

```
my $branch = $api->create_branch (
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/repository/branches` and returns the decoded response content.

delete_branch

```
$api->delete_branch (
    $project_id,
    $branch_name,
);
```

Sends a DELETE request to `projects/:project_id/repository/branches/:branch_name`.

delete_merged_branches

```
$api->delete_merged_branches (
    $project_id,
);
```

Sends a DELETE request to `projects/:project_id/repository/merged_branches`.

Broadcast Messages

See <https://docs.gitlab.com/ce/api/broadcast_messages.html>.

broadcast_messages

```
my $messages = $api->broadcast_messages (
    \%params,
);
```

Sends a GET request to `broadcast_messages` and returns the decoded response content.

broadcast_message

```
my $message = $api->broadcast_message (
    $message_id,
);
```

Sends a GET request to `broadcast_messages/:message_id` and returns the decoded response

content.

create_broadcast_message

```
my $message = $api->create_broadcast_message(
    \%params,
);
```

Sends a POST request to broadcast_messages and returns the decoded response content.

edit_broadcast_message

```
my $message = $api->edit_broadcast_message(
    $message_id,
    \%params,
);
```

Sends a PUT request to broadcast_messages/:message_id and returns the decoded response content.

delete_broadcast_message

```
$api->delete_broadcast_message(
    $message_id,
);
```

Sends a DELETE request to broadcast_messages/:message_id.

Project-level Variables

See <https://docs.gitlab.com/ce/api/project_level_variables.html>.

project_variables

```
my $variables = $api->project_variables(
    $project_id,
    \%params,
);
```

Sends a GET request to projects/:project_id/variables and returns the decoded response content.

project_variable

```
my $variable = $api->project_variable(
    $project_id,
    $variable_key,
);
```

Sends a GET request to projects/:project_id/variables/:variable_key and returns the decoded response content.

create_project_variable

```
my $variable = $api->create_project_variable(
    $project_id,
    \%params,
);
```

Sends a POST request to projects/:project_id/variables and returns the decoded response content.

edit_project_variable

```
my $variable = $api->edit_project_variable(
    $project_id,
    $variable_key,
    \%params,
);
```

Sends a PUT request to projects/:project_id/variables/:variable_key and returns

the decoded response content.

```
delete_project_variable
    $api->delete_project_variable(
        $project_id,
        $variable_key,
    );
```

Sends a DELETE request to `projects/:project_id/variables/:variable_key`.

Group-level Variables

See <https://docs.gitlab.com/ce/api/group_level_variables.html>.

```
group_variables
    my $variables = $api->group_variables(
        $group_id,
        \%params,
    );
```

Sends a GET request to `groups/:group_id/variables` and returns the decoded response content.

```
group_variable
    my $variable = $api->group_variable(
        $group_id,
        $variable_key,
    );
```

Sends a GET request to `groups/:group_id/variables/:variable_key` and returns the decoded response content.

```
create_group_variable
    my $variable = $api->create_group_variable(
        $group_id,
        \%params,
    );
```

Sends a POST request to `groups/:group_id/variables` and returns the decoded response content.

```
edit_group_variable
    my $variable = $api->edit_group_variable(
        $group_id,
        $variable_key,
        \%params,
    );
```

Sends a PUT request to `groups/:group_id/variables/:variable_key` and returns the decoded response content.

```
delete_group_variable
    $api->delete_group_variable(
        $group_id,
        $variable_key,
    );
```

Sends a DELETE request to `groups/:group_id/variables/:variable_key`.

Snippets

See <<https://docs.gitlab.com/ce/api/snippets.html>>.

snippets

```
my $snippets = $api->snippets();
```

Sends a GET request to `snippets` and returns the decoded response content.

snippet

```
my $snippet = $api->snippet(
    $snippet_id,
);
```

Sends a GET request to `snippets/:snippet_id` and returns the decoded response content.

create_snippet

```
my $snippet = $api->create_snippet(
    \%params,
);
```

Sends a POST request to `snippets` and returns the decoded response content.

edit_snippet

```
my $snippet = $api->edit_snippet(
    $snippet_id,
    \%params,
);
```

Sends a PUT request to `snippets/:snippet_id` and returns the decoded response content.

delete_snippet

```
$api->delete_snippet(
    $snippet_id,
);
```

Sends a DELETE request to `snippets/:snippet_id`.

public_snippets

```
my $snippets = $api->public_snippets(
    \%params,
);
```

Sends a GET request to `snippets/public` and returns the decoded response content.

snippet_user_agent_detail

```
my $user_agent = $api->snippet_user_agent_detail(
    $snippet_id,
);
```

Sends a GET request to `snippets/:snippet_id/user_agent_detail` and returns the decoded response content.

Commits

See <<https://docs.gitlab.com/ce/api/commits.html>>.

commits

```
my $commits = $api->commits(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/repository/commits` and returns the decoded response content.

create_commit

```
my $commit = $api->create_commit(
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/repository/commits` and returns the decoded response content.

`commit`

```
my $commit = $api->commit(
    $project_id,
    $commit_sha,
);
```

Sends a GET request to `projects/:project_id/repository/commits/:commit_sha` and returns the decoded response content.

`commit_refs`

```
my $refs = $api->commit_refs(
    $project_id,
    $commit_sha,
    \%params,
);
```

Sends a GET request to `projects/:project_id/repository/commits/:commit_sha/refs` and returns the decoded response content.

`cherry_pick_commit`

```
my $commit = $api->cherry_pick_commit(
    $project_id,
    $commit_sha,
    \%params,
);
```

Sends a POST request to `projects/:project_id/repository/commits/:commit_sha/cherry_pick` and returns the decoded response content.

`commit_diff`

```
my $diff = $api->commit_diff(
    $project_id,
    $commit_sha,
    \%params,
);
```

Sends a GET request to `projects/:project_id/repository/commits/:commit_sha/diff` and returns the decoded response content.

`commit_comments`

```
my $comments = $api->commit_comments(
    $project_id,
    $commit_sha,
    \%params,
);
```

Sends a GET request to `projects/:project_id/repository/commits/:commit_sha/comments` and returns the decoded response content.

create_commit_comment

```
$api->create_commit_comment (
    $project_id,
    $commit_sha,
    \%params,
);
```

Sends a POST request to projects/:project_id/repository/commits/:commit_sha/comments.

commit_statuses

```
my $build_statuses = $api->commit_statuses (
    $project_id,
    $commit_sha,
    \%params,
);
```

Sends a GET request to projects/:project_id/repository/commits/:commit_sha/statuses and returns the decoded response content.

create_commit_status

```
my $build_status = $api->create_commit_status (
    $project_id,
    $commit_sha,
    \%params,
);
```

Sends a POST request to projects/:project_id/statuses/:commit_sha and returns the decoded response content.

Custom Attributes

See <https://docs.gitlab.com/ce/api/custom_attributes.html>.

custom_user_attributes

```
my $attributes = $api->custom_user_attributes (
    $user_id,
);
```

Sends a GET request to users/:user_id/custom_attributes and returns the decoded response content.

custom_group_attributes

```
my $attributes = $api->custom_group_attributes (
    $group_id,
);
```

Sends a GET request to groups/:group_id/custom_attributes and returns the decoded response content.

custom_project_attributes

```
my $attributes = $api->custom_project_attributes (
    $project_id,
);
```

Sends a GET request to projects/:project_id/custom_attributes and returns the decoded response content.

custom_user_attribute

```
my $attribute = $api->custom_user_attribute(
    $user_id,
    $attribute_key,
);
```

Sends a GET request to `users/:user_id/custom_attributes/:attribute_key` and returns the decoded response content.

`custom_group_attribute`

```
my $attribute = $api->custom_group_attribute(
    $group_id,
    $attribute_key,
);
```

Sends a GET request to `groups/:group_id/custom_attributes/:attribute_key` and returns the decoded response content.

`custom_project_attribute`

```
my $attribute = $api->custom_project_attribute(
    $project_id,
    $attribute_key,
);
```

Sends a GET request to `projects/:project_id/custom_attributes/:attribute_key` and returns the decoded response content.

`set_custom_user_attribute`

```
my $attribute = $api->set_custom_user_attribute(
    $user_id,
    $attribute_key,
    \%params,
);
```

Sends a PUT request to `users/:user_id/custom_attributes/:attribute_key` and returns the decoded response content.

`set_custom_group_attribute`

```
my $attribute = $api->set_custom_group_attribute(
    $group_id,
    $attribute_key,
    \%params,
);
```

Sends a PUT request to `groups/:group_id/custom_attributes/:attribute_key` and returns the decoded response content.

`set_custom_project_attribute`

```
my $attribute = $api->set_custom_project_attribute(
    $project_id,
    $attribute_key,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/custom_attributes/:attribute_key` and returns the decoded response content.

`delete_custom_user_attribute`

```
$api->delete_custom_user_attribute(
    $user_id,
    $attribute_key,
);
```

Sends a DELETE request to `users/:user_id/custom_attributes/:attribute_key`.

```
delete_custom_group_attribute
    $api->delete_custom_group_attribute(
        $group_id,
        $attribute_key,
    );
```

Sends a DELETE request to `groups/:group_id/custom_attributes/:attribute_key`.

```
delete_custom_project_attribute
    $api->delete_custom_project_attribute(
        $project_id,
        $attribute_key,
    );
```

Sends a DELETE request to `projects/:project_id/custom_attributes/:attribute_key`.

Deployments

See <https://docs.gitlab.com/ce/api/deployments.html>.

```
deployments
    my $deployments = $api->deployments(
        $project_id,
        \%params,
    );
```

Sends a GET request to `projects/:project_id/deployments` and returns the decoded response content.

```
deployment
    my $deployment = $api->deployment(
        $project_id,
        $deployment_id,
    );
```

Sends a GET request to `projects/:project_id/deployments/:deployment_id` and returns the decoded response content.

Deploy Keys

See https://docs.gitlab.com/ce/api/deploy_keys.html.

```
all_deploy_keys
    my $keys = $api->all_deploy_keys(
        \%params,
    );
```

Sends a GET request to `deploy_keys` and returns the decoded response content.

```
deploy_keys
    my $keys = $api->deploy_keys(
        $project_id,
        \%params,
    );
```

Sends a GET request to `projects/:project_id/deploy_keys` and returns the decoded response content.

deploy_key

```
my $key = $api->deploy_key(  
    $project_id,  
    $key_id,  
);
```

Sends a GET request to `projects/:project_id/deploy_keys/:key_id` and returns the decoded response content.

create_deploy_key

```
my $key = $api->create_deploy_key(  
    $project_id,  
    \%params,  
);
```

Sends a POST request to `projects/:project_id/deploy_keys` and returns the decoded response content.

delete_deploy_key

```
$api->delete_deploy_key(  
    $project_id,  
    $key_id,  
);
```

Sends a DELETE request to `projects/:project_id/deploy_keys/:key_id`.

enable_deploy_key

```
my $key = $api->enable_deploy_key(  
    $project_id,  
    $key_id,  
);
```

Sends a POST request to `projects/:project_id/deploy_keys/:key_id/enable` and returns the decoded response content.

Environments

See [<https://docs.gitlab.com/ce/api/environments.html>](https://docs.gitlab.com/ce/api/environments.html).

environments

```
my $environments = $api->environments(  
    $project_id,  
    \%params,  
);
```

Sends a GET request to `projects/:project_id/environments` and returns the decoded response content.

create_environment

```
my $environment = $api->create_environment(  
    $project_id,  
    \%params,  
);
```

Sends a POST request to `projects/:project_id/environments` and returns the decoded response content.

edit_environment

```
my $environment = $api->edit_environment(
    $project_id,
    $environments_id,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/environments/:environments_id` and returns the decoded response content.

delete_environment

```
$api->delete_environment(
    $project_id,
    $environment_id,
);
```

Sends a DELETE request to `projects/:project_id/environments/:environment_id`.

stop_environment

```
my $environment = $api->stop_environment(
    $project_id,
    $environment_id,
);
```

Sends a POST request to `projects/:project_id/environments/:environment_id/stop` and returns the decoded response content.

Events

See <https://docs.gitlab.com/ce/api/events.html>.

all_events

```
my $events = $api->all_events(
    \%params,
);
```

Sends a GET request to `events` and returns the decoded response content.

user_events

```
my $events = $api->user_events(
    $user_id,
    \%params,
);
```

Sends a GET request to `users/:user_id/events` and returns the decoded response content.

project_events

```
my $events = $api->project_events(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/events` and returns the decoded response content.

Feature flags

See <https://docs.gitlab.com/ce/api/features.html>.

features

```
my $features = $api->features();
```

Sends a GET request to `features` and returns the decoded response content.


```
set_feature
    my $feature = $api->set_feature(
        $name,
        \%params,
    );
```

Sends a POST request to `features/:name` and returns the decoded response content.

Gitignores

See <<https://docs.gitlab.com/ce/api/templates/gitignores.html>>.

```
gitignores_templates
    my $templates = $api->gitignores_templates(
        \%params,
    );
```

Sends a GET request to `templates/gitignores` and returns the decoded response content.

```
gitignores_template
    my $template = $api->gitignores_template(
        $template_key,
    );
```

Sends a GET request to `templates/gitignores/:template_key` and returns the decoded response content.

GitLab CI YMLs

See <https://docs.gitlab.com/ce/api/templates/gitlab_ci_ymls.html>.

```
gitlab_ci_ymls_templates
    my $templates = $api->gitlab_ci_ymls_templates(
        \%params,
    );
```

Sends a GET request to `templates/gitlab_ci_ymls` and returns the decoded response content.

```
gitlab_ci_ymls_template
    my $template = $api->gitlab_ci_ymls_template(
        $template_key,
    );
```

Sends a GET request to `templates/gitlab_ci_ymls/:template_key` and returns the decoded response content.

Groups

See <<https://docs.gitlab.com/ce/api/groups.html>>.

```
groups
    my $groups = $api->groups(
        \%params,
    );
```

Sends a GET request to `groups` and returns the decoded response content.

```
group_subgroups
    my $subgroups = $api->group_subgroups(
        $group_id,
        \%params,
    );
```

Sends a GET request to `groups/:group_id/subgroups` and returns the decoded response content.

group_projects

```
my $projects = $api->group_projects(  
    $group_id,  
    \%params,  
);
```

Sends a GET request to `groups/:group_id/projects` and returns the decoded response content.

group

```
my $group = $api->group(  
    $group_id,  
    \%params,  
);
```

Sends a GET request to `groups/:group_id` and returns the decoded response content.

create_group

```
$api->create_group(  
    \%params,  
);
```

Sends a POST request to `groups`.

transfer_project_to_group

```
$api->transfer_project_to_group(  
    $group_id,  
    $project_id,  
);
```

Sends a POST request to `groups/:group_id/projects/:project_id`.

edit_group

```
my $group = $api->edit_group(  
    $group_id,  
    \%params,  
);
```

Sends a PUT request to `groups/:group_id` and returns the decoded response content.

delete_group

```
$api->delete_group(  
    $group_id,  
);
```

Sends a DELETE request to `groups/:group_id`.

sync_group_with_ldap

```
$api->sync_group_with_ldap(  
    $group_id,  
);
```

Sends a POST request to `groups/:group_id/ldap_sync`.

create_ldap_group_link

```
$api->create_ldap_group_link(  
    $group_id,  
    \%params,  
);
```

Sends a POST request to `groups/:group_id/ldap_group_links`.

```
delete_ldap_group_link
    $api->delete_ldap_group_link(
        $group_id,
        $cn,
    );
```

Sends a DELETE request to `groups/:group_id/ldap_group_links/:cn`.

```
delete_ldap_provider_group_link
    $api->delete_ldap_provider_group_link(
        $group_id,
        $provider,
        $cn,
    );
```

Sends a DELETE request to `groups/:group_id/ldap_group_links/:provider/:cn`.

Group access requests

See https://docs.gitlab.com/ce/api/access_requests.html.

```
group_access_requests
    my $requests = $api->group_access_requests(
        $group_id,
        \%params,
    );
```

Sends a GET request to `groups/:group_id/access_requests` and returns the decoded response content.

```
request_group_access
    my $request = $api->request_group_access(
        $group_id,
    );
```

Sends a POST request to `groups/:group_id/access_requests` and returns the decoded response content.

```
approve_group_access
    my $request = $api->approve_group_access(
        $group_id,
        $user_id,
    );
```

Sends a PUT request to `groups/:group_id/access_requests/:user_id/approve` and returns the decoded response content.

```
deny_group_access
    $api->deny_group_access(
        $group_id,
        $user_id,
    );
```

Sends a DELETE request to `groups/:group_id/access_requests/:user_id`.

Group badges

See https://docs.gitlab.com/ce/api/group_badges.html.

```
group_badges
    my $badges = $api->group_badges(
        $group_id,
    );
```

Sends a GET request to `groups/:group_id/badges` and returns the decoded response content.

group_badge

```
my $badge = $api->group_badge (
    $group_id,
    $badge_id,
);
```

Sends a GET request to `groups/:group_id/badges/:badge_id` and returns the decoded response content.

create_group_badge

```
my $badge = $api->create_group_badge (
    $group_id,
    \%params,
);
```

Sends a POST request to `groups/:group_id/badges` and returns the decoded response content.

edit_group_badge

```
my $badge = $api->edit_group_badge (
    $group_id,
    $badge_id,
    \%params,
);
```

Sends a PUT request to `groups/:group_id/badges/:badge_id` and returns the decoded response content.

delete_group_badge

```
$api->delete_group_badge (
    $group_id,
    $badge_id,
);
```

Sends a DELETE request to `groups/:group_id/badges/:badge_id`.

preview_group_badge

```
my $preview = $api->preview_group_badge (
    $group_id,
    \%params,
);
```

Sends a GET request to `groups/:group_id/badges/render` and returns the decoded response content.

Group members

See <<https://docs.gitlab.com/ce/api/members.html>>.

group_members

```
my $members = $api->group_members (
    $group_id,
    \%params,
);
```

Sends a GET request to `groups/:group_id/members` and returns the decoded response content.

all_group_members

```
my $members = $api->all_group_members (
    $group_id,
    \%params,
);
```

Sends a GET request to `groups/:group_id/members/all` and returns the decoded response

content.

group_member

```
my $member = $api->group_member(
    $project_id,
    $user_id,
);
```

Sends a GET request to `groups/:project_id/members/:user_id` and returns the decoded response content.

add_group_member

```
my $member = $api->add_group_member(
    $group_id,
    \%params,
);
```

Sends a POST request to `groups/:group_id/members` and returns the decoded response content.

update_group_member

```
my $member = $api->update_group_member(
    $group_id,
    $user_id,
    \%params,
);
```

Sends a PUT request to `groups/:group_id/members/:user_id` and returns the decoded response content.

remove_group_member

```
$api->remove_group_member(
    $group_id,
    $user_id,
);
```

Sends a DELETE request to `groups/:group_id/members/:user_id`.

Issues

See <<https://docs.gitlab.com/ce/api/issues.html>>.

global_issues

```
my $issues = $api->global_issues(
    \%params,
);
```

Sends a GET request to `issues` and returns the decoded response content.

group_issues

```
my $issues = $api->group_issues(
    $group_id,
    \%params,
);
```

Sends a GET request to `groups/:group_id/issues` and returns the decoded response content.

issues

```
my $issues = $api->issues(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/issues` and returns the decoded response

content.

issue

```
my $issue = $api->issue(
    $project_id,
    $issue_iid,
);
```

Sends a GET request to `projects/:project_id/issues/:issue_iid` and returns the decoded response content.

create_issue

```
my $issue = $api->create_issue(
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/issues` and returns the decoded response content.

edit_issue

```
my $issue = $api->edit_issue(
    $project_id,
    $issue_iid,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/issues/:issue_iid` and returns the decoded response content.

delete_issue

```
$api->delete_issue(
    $project_id,
    $issue_iid,
);
```

Sends a DELETE request to `projects/:project_id/issues/:issue_iid`.

move_issue

```
my $issue = $api->move_issue(
    $project_id,
    $issue_iid,
    \%params,
);
```

Sends a POST request to `projects/:project_id/issues/:issue_iid/move` and returns the decoded response content.

subscribe_to_issue

```
my $issue = $api->subscribe_to_issue(
    $project_id,
    $issue_iid,
);
```

Sends a POST request to `projects/:project_id/issues/:issue_iid/subscribe` and returns the decoded response content.

unsubscribe_from_issue

```
my $issue = $api->unsubscribe_from_issue(
    $project_id,
    $issue_iid,
);
```

Sends a POST request to `projects/:project_id/issues/:issue_iid/unsubscribe` and returns the decoded response content.

`create_issue_todo`

```
my $todo = $api->create_issue_todo(
    $project_id,
    $issue_iid,
);
```

Sends a POST request to `projects/:project_id/issues/:issue_iid/todo` and returns the decoded response content.

`set_issue_time_estimate`

```
my $tracking = $api->set_issue_time_estimate(
    $project_id,
    $issue_iid,
    \%params,
);
```

Sends a POST request to `projects/:project_id/issues/:issue_iid/time_estimate` and returns the decoded response content.

`reset_issue_time_estimate`

```
my $tracking = $api->reset_issue_time_estimate(
    $project_id,
    $issue_iid,
);
```

Sends a POST request to `projects/:project_id/issues/:issue_iid/reset_time_estimate` and returns the decoded response content.

`add_issue_spent_time`

```
my $tracking = $api->add_issue_spent_time(
    $project_id,
    $issue_iid,
    \%params,
);
```

Sends a POST request to `projects/:project_id/issues/:issue_iid/add_spent_time` and returns the decoded response content.

`reset_issue_spent_time`

```
my $tracking = $api->reset_issue_spent_time(
    $project_id,
    $issue_iid,
);
```

Sends a POST request to `projects/:project_id/issues/:issue_iid/reset_spent_time` and returns the decoded response content.

issue_time_stats

```
my $tracking = $api->issue_time_stats(
    $project_id,
    $issue_iid,
);
```

Sends a GET request to `projects/:project_id/issues/:issue_iid/time_stats` and returns the decoded response content.

issue_closed_by

```
my $merge_requests = $api->issue_closed_by(
    $project_id,
    $issue_iid,
);
```

Sends a GET request to `projects/:project_id/issues/:issue_iid/closed_by` and returns the decoded response content.

issue_user_agent_detail

```
my $user_agent = $api->issue_user_agent_detail(
    $project_id,
    $issue_iid,
);
```

Sends a GET request to `projects/:project_id/issues/:issue_iid/user_agent_detail` and returns the decoded response content.

Issue Boards

See <https://docs.gitlab.com/ce/api/boards.html>.

project_boards

```
my $boards = $api->project_boards(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/boards` and returns the decoded response content.

project_board_lists

```
my $lists = $api->project_board_lists(
    $project_id,
    $board_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/boards/:board_id/lists` and returns the decoded response content.

project_board_list

```
my $list = $api->project_board_list(
    $project_id,
    $board_id,
    $list_id,
);
```

Sends a GET request to `projects/:project_id/boards/:board_id/lists/:list_id` and returns the decoded response content.

create_project_board_list

```
my $list = $api->create_project_board_list(
    $project_id,
    $board_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/boards/:board_id/lists` and returns the decoded response content.

edit_project_board_list

```
my $list = $api->edit_project_board_list(
    $project_id,
    $board_id,
    $list_id,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/boards/:board_id/lists/:list_id` and returns the decoded response content.

delete_project_board_list

```
$api->delete_project_board_list(
    $project_id,
    $board_id,
    $list_id,
);
```

Sends a DELETE request to `projects/:project_id/boards/:board_id/lists/:list_id`.

Group Issue Boards

See <https://docs.gitlab.com/ce/api/group_boards.html>.

group_boards

```
my $boards = $api->group_boards(
    $group_id,
);
```

Sends a GET request to `groups/:group_id/boards` and returns the decoded response content.

group_board

```
my $board = $api->group_board(
    $group_id,
    $board_id,
);
```

Sends a GET request to `groups/:group_id/boards/:board_id` and returns the decoded response content.

group_board_lists

```
my $lists = $api->group_board_lists(
    $group_id,
    $board_id,
);
```

Sends a GET request to `groups/:group_id/boards/:board_id/lists` and returns the decoded response content.

group_board_list

```
my $list = $api->group_board_list(
    $group_id,
    $board_id,
    $list_id,
);
```

Sends a GET request to `groups/:group_id/boards/:board_id/lists/:list_id` and returns the decoded response content.

create_group_board_list

```
my $list = $api->create_group_board_list(
    $group_id,
    $board_id,
    \%params,
);
```

Sends a POST request to `groups/:group_id/boards/:board_id/lists` and returns the decoded response content.

edit_group_board_list

```
my $list = $api->edit_group_board_list(
    $group_id,
    $board_id,
    $list_id,
    \%params,
);
```

Sends a PUT request to `groups/:group_id/boards/:board_id/lists/:list_id` and returns the decoded response content.

delete_group_board_list

```
$api->delete_group_board_list(
    $group_id,
    $board_id,
    $list_id,
);
```

Sends a DELETE request to `groups/:group_id/boards/:board_id/lists/:list_id`.

Jobs

See <<https://docs.gitlab.com/ce/api/jobs.html>>.

jobs

```
my $jobs = $api->jobs(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/jobs` and returns the decoded response content.

pipeline_jobs

```
my $jobs = $api->pipeline_jobs(
    $project_id,
    $pipeline_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/pipelines/:pipeline_id/jobs` and returns the decoded response content.

job

```
my $job = $api->job(
    $project_id,
    $job_id,
);
```

Sends a GET request to `projects/:project_id/jobs/:job_id` and returns the decoded response content.

job_artifacts

```
my $artifacts = $api->job_artifacts(
    $project_id,
    $job_id,
);
```

Sends a GET request to `projects/:project_id/jobs/:job_id/artifacts` and returns the decoded response content.

job_artifacts_archive

```
my $archive = $api->job_artifacts_archive(
    $project_id,
    $ref_name,
    \%params,
);
```

Sends a GET request to `projects/:project_id/jobs/artifacts/:ref_name/download` and returns the decoded response content.

job_artifacts_file

```
my $file = $api->job_artifacts_file(
    $project_id,
    $job_id,
    $artifact_path,
);
```

Sends a GET request to `projects/:project_id/jobs/:job_id/artifacts/:artifact_path` and returns the decoded response content.

job_trace_file

```
my $file = $api->job_trace_file(
    $project_id,
    $job_id,
);
```

Sends a GET request to `projects/:project_id/jobs/:job_id/trace` and returns the decoded response content.

cancel_job

```
my $job = $api->cancel_job(
    $project_id,
    $job_id,
);
```

Sends a POST request to `projects/:project_id/jobs/:job_id/cancel` and returns the decoded response content.

retry_job

```
my $job = $api->retry_job(
    $project_id,
    $job_id,
);
```

Sends a POST request to `projects/:project_id/jobs/:job_id/retry` and returns the decoded response content.

erase_job

```
my $job = $api->erase_job(
    $project_id,
    $job_id,
);
```

Sends a POST request to `projects/:project_id/jobs/:job_id/erase` and returns the decoded response content.

keep_job_artifacts

```
my $job = $api->keep_job_artifacts(
    $project_id,
    $job_id,
);
```

Sends a POST request to `projects/:project_id/jobs/:job_id/artifacts/keep` and returns the decoded response content.

play_job

```
my $job = $api->play_job(
    $project_id,
    $job_id,
);
```

Sends a POST request to `projects/:project_id/jobs/:job_id/play` and returns the decoded response content.

Keys

See <<https://docs.gitlab.com/ce/api/keys.html>>.

key

```
my $key = $api->key(
    $key_id,
);
```

Sends a GET request to `keys/:key_id` and returns the decoded response content.

Labels

See <<https://docs.gitlab.com/ce/api/labels.html>>.

labels

```
my $labels = $api->labels(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/labels` and returns the decoded response content.

create_label

```
my $label = $api->create_label(
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/labels` and returns the decoded response content.

`delete_label`

```
$api->delete_label(
    $project_id,
    \%params,
);
```

Sends a DELETE request to `projects/:project_id/labels`.

`edit_label`

```
my $label = $api->edit_label(
    $project_id,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/labels` and returns the decoded response content.

`subscribe_to_label`

```
my $label = $api->subscribe_to_label(
    $project_id,
    $label_id,
);
```

Sends a POST request to `projects/:project_id/labels/:label_id/subscribe` and returns the decoded response content.

`unsubscribe_from_label`

```
$api->unsubscribe_from_label(
    $project_id,
    $label_id,
);
```

Sends a POST request to `projects/:project_id/labels/:label_id/unsubscribe`.

Markdown

See <<https://docs.gitlab.com/ce/api/markdown.html>>.

`markdown`

```
my $html = $api->markdown(
    \%params,
);
```

Sends a POST request to `markdown` and returns the decoded response content.

Merge requests

See <https://docs.gitlab.com/ce/api/merge_requests.html>.

`global_merge_requests`

```
my $merge_requests = $api->global_merge_requests(
    \%params,
);
```

Sends a GET request to `merge_requests` and returns the decoded response content.

merge_requests

```
my $merge_requests = $api->merge_requests(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/merge_requests` and returns the decoded response content.

merge_request

```
my $merge_request = $api->merge_request(
    $project_id,
    $merge_request_iid,
);
```

Sends a GET request to `projects/:project_id/merge_requests/:merge_request_iid` and returns the decoded response content.

merge_request_commits

```
my $commits = $api->merge_request_commits(
    $project_id,
    $merge_request_iid,
);
```

Sends a GET request to `projects/:project_id/merge_requests/:merge_request_iid/commits` and returns the decoded response content.

merge_request_with_changes

```
my $merge_request = $api->merge_request_with_changes(
    $project_id,
    $merge_request_iid,
);
```

Sends a GET request to `projects/:project_id/merge_requests/:merge_request_iid/changes` and returns the decoded response content.

create_merge_request

```
my $merge_request = $api->create_merge_request(
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/merge_requests` and returns the decoded response content.

edit_merge_request

```
my $merge_request = $api->edit_merge_request(
    $project_id,
    $merge_request_iid,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/merge_requests/:merge_request_iid` and returns the decoded response content.

delete_merge_request

```
$api->delete_merge_request (
    $project_id,
    $merge_request_iid,
);
```

Sends a DELETE request to `projects/:project_id/merge_requests/:merge_request_iid`.

accept_merge_request

```
my $merge_request = $api->accept_merge_request (
    $project_id,
    $merge_request_iid,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/merge_requests/:merge_request_iid/merge` and returns the decoded response content.

cancel_merge_when_pipeline_succeeds

```
my $merge_request = $api->cancel_merge_when_pipeline_succeeds (
    $project_id,
    $merge_request_iid,
);
```

Sends a PUT request to `projects/:project_id/merge_requests/:merge_request_iid/cancel_merge_when_pipeline_succeeds` and returns the decoded response content.

merge_request_closes_issues

```
my $issues = $api->merge_request_closes_issues (
    $project_id,
    $merge_request_iid,
    \%params,
);
```

Sends a GET request to `projects/:project_id/merge_requests/:merge_request_iid/closes_issues` and returns the decoded response content.

subscribe_to_merge_request

```
my $merge_request = $api->subscribe_to_merge_request (
    $project_id,
    $merge_request_iid,
);
```

Sends a POST request to `projects/:project_id/merge_requests/:merge_request_iid/subscribe` and returns the decoded response content.

unsubscribe_from_merge_request

```
my $merge_request = $api->unsubscribe_from_merge_request (
    $project_id,
    $merge_request_iid,
);
```

Sends a POST request to `projects/:project_id/merge_requests/:merge_request_iid/unsubscribe` and returns the decoded response content.

`create_merge_request_todo`

```
my $todo = $api->create_merge_request_todo(
    $project_id,
    $merge_request_iid,
);
```

Sends a POST request to `projects/:project_id/merge_requests/:merge_request_iid/todo` and returns the decoded response content.

`merge_request_diff_versions`

```
my $versions = $api->merge_request_diff_versions(
    $project_id,
    $merge_request_iid,
);
```

Sends a GET request to `projects/:project_id/merge_requests/:merge_request_iid/versions` and returns the decoded response content.

`merge_request_diff_version`

```
my $version = $api->merge_request_diff_version(
    $project_id,
    $merge_request_iid,
    $version_id,
);
```

Sends a GET request to `projects/:project_id/merge_requests/:merge_request_iid/versions/:version_id` and returns the decoded response content.

`set_merge_request_time_estimate`

```
my $tracking = $api->set_merge_request_time_estimate(
    $project_id,
    $merge_request_iid,
    \%params,
);
```

Sends a POST request to `projects/:project_id/merge_requests/:merge_request_iid/time_estimate` and returns the decoded response content.

`reset_merge_request_time_estimate`

```
my $tracking = $api->reset_merge_request_time_estimate(
    $project_id,
    $merge_request_iid,
);
```

Sends a POST request to `projects/:project_id/merge_requests/:merge_request_iid/reset_time_estimate` and returns the decoded response content.

`add_merge_request_spent_time`

```
my $tracking = $api->add_merge_request_spent_time(
    $project_id,
    $merge_request_iid,
    \%params,
);
```

Sends a POST request to

projects/:project_id/merge_requests/:merge_request_iid/add_spent_time
and returns the decoded response content.

reset_merge_request_spent_time

```
my $tracking = $api->reset_merge_request_spent_time(
    $project_id,
    $merge_request_iid,
);
```

Sends a POST request to projects/:project_id/merge_requests/:merge_request_iid/reset_spent_time and returns the decoded response content.

merge_request_time_stats

```
my $tracking = $api->merge_request_time_stats(
    $project_id,
    $merge_request_iid,
);
```

Sends a GET request to projects/:project_id/merge_requests/:merge_request_iid/time_stats and returns the decoded response content.

Milestones

See <<https://docs.gitlab.com/ce/api/milestones.html>>.

project_milestones

```
my $milestones = $api->project_milestones(
    $project_id,
    \%params,
);
```

Sends a GET request to projects/:project_id/milestones and returns the decoded response content.

project_milestone

```
my $milestone = $api->project_milestone(
    $project_id,
    $milestone_id,
);
```

Sends a GET request to projects/:project_id/milestones/:milestone_id and returns the decoded response content.

create_project_milestone

```
my $milestone = $api->create_project_milestone(
    $project_id,
    \%params,
);
```

Sends a POST request to projects/:project_id/milestones and returns the decoded response content.

edit_project_milestone

```
my $milestone = $api->edit_project_milestone(
    $project_id,
    $milestone_id,
    \%params,
);
```

Sends a PUT request to projects/:project_id/milestones/:milestone_id and returns the decoded response content.

project_milestone_issues

```
my $issues = $api->project_milestone_issues(
    $project_id,
    $milestone_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/milestones/:milestone_id/issues` and returns the decoded response content.

project_milestone_merge_requests

```
my $merge_requests = $api->project_milestone_merge_requests(
    $project_id,
    $milestone_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/milestones/:milestone_id/merge_requests` and returns the decoded response content.

Group milestones

See https://docs.gitlab.com/ce/api/group_milestones.html.

group_milestones

```
my $milestones = $api->group_milestones(
    $group_id,
    \%params,
);
```

Sends a GET request to `groups/:group_id/milestones` and returns the decoded response content.

group_milestone

```
my $milestone = $api->group_milestone(
    $group_id,
    $milestone_id,
);
```

Sends a GET request to `groups/:group_id/milestones/:milestone_id` and returns the decoded response content.

create_group_milestone

```
my $milestone = $api->create_group_milestone(
    $group_id,
    \%params,
);
```

Sends a POST request to `groups/:group_id/milestones` and returns the decoded response content.

edit_group_milestone

```
my $milestone = $api->edit_group_milestone(
    $group_id,
    $milestone_id,
    \%params,
);
```

Sends a PUT request to `groups/:group_id/milestones/:milestone_id` and returns the decoded response content.

group_milestone_issues

```
my $issues = $api->group_milestone_issues(
    $group_id,
    $milestone_id,
    \%params,
);
```

Sends a GET request to `groups/:group_id/milestones/:milestone_id/issues` and returns the decoded response content.

group_milestone_merge_requests

```
my $merge_requests = $api->group_milestone_merge_requests(
    $group_id,
    $milestone_id,
    \%params,
);
```

Sends a GET request to `groups/:group_id/milestones/:milestone_id/merge_requests` and returns the decoded response content.

Namespaces

See <<https://docs.gitlab.com/ce/api/namespaces.html>>.

namespaces

```
my $namespaces = $api->namespaces(
    \%params,
);
```

Sends a GET request to `namespaces` and returns the decoded response content.

namespace

```
my $namespace = $api->namespace(
    $namespace_id,
);
```

Sends a GET request to `namespaces/:namespace_id` and returns the decoded response content.

Notes

See <<https://docs.gitlab.com/ce/api/notes.html>>.

issue_notes

```
my $notes = $api->issue_notes(
    $project_id,
    $issue_iid,
    \%params,
);
```

Sends a GET request to `projects/:project_id/issues/:issue_iid/notes` and returns the decoded response content.

issue_note

```
my $note = $api->issue_note(
    $project_id,
    $issue_iid,
    $note_id,
);
```

Sends a GET request to `projects/:project_id/issues/:issue_iid/notes/:note_id` and returns the decoded response content.

`create_issue_note`

```
my $note = $api->create_issue_note(
    $project_id,
    $issue_iid,
    \%params,
);
```

Sends a POST request to `projects/:project_id/issues/:issue_iid/notes` and returns the decoded response content.

`edit_issue_note`

```
$api->edit_issue_note(
    $project_id,
    $issue_iid,
    $note_id,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/issues/:issue_iid/notes/:note_id`.

`delete_issue_note`

```
$api->delete_issue_note(
    $project_id,
    $issue_iid,
    $note_id,
);
```

Sends a DELETE request to `projects/:project_id/issues/:issue_iid/notes/:note_id`.

`snippet_notes`

```
my $notes = $api->snippet_notes(
    $project_id,
    $snippet_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/snippets/:snippet_id/notes` and returns the decoded response content.

`snippet_note`

```
my $note = $api->snippet_note(
    $project_id,
    $snippet_id,
    $note_id,
);
```

Sends a GET request to `projects/:project_id/snippets/:snippet_id/notes/:note_id` and returns the decoded response content.

`create_snippet_note`

```
my $note = $api->create_snippet_note(
    $project_id,
    $snippet_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/snippets/:snippet_id/notes` and

returns the decoded response content.

`edit_snippet_note`

```
$api->edit_snippet_note(
    $project_id,
    $snippet_id,
    $note_id,
    \%params,
);
```

Sends a PUT request to
projects/:project_id/snippets/:snippet_id/notes/:note_id.

`delete_snippet_note`

```
$api->delete_snippet_note(
    $project_id,
    $snippet_id,
    $note_id,
);
```

Sends a DELETE request to
projects/:project_id/snippets/:snippet_id/notes/:note_id.

`merge_request_notes`

```
my $notes = $api->merge_request_notes(
    $project_id,
    $merge_request_iid,
    \%params,
);
```

Sends a GET request to
projects/:project_id/merge_requests/:merge_request_iid/notes and returns
the decoded response content.

`merge_request_note`

```
my $note = $api->merge_request_note(
    $project_id,
    $merge_request_iid,
    $note_id,
);
```

Sends a GET request to
projects/:project_id/merge_requests/:merge_request_iid/notes/:note_id
and returns the decoded response content.

`create_merge_request_note`

```
my $note = $api->create_merge_request_note(
    $project_id,
    $merge_request_iid,
    \%params,
);
```

Sends a POST request to
projects/:project_id/merge_requests/:merge_request_iid/notes and returns
the decoded response content.

`edit_merge_request_note`

```

    $api->edit_merge_request_note (
        $project_id,
        $merge_request_iid,
        $note_id,
        \%params,
    );

```

Sends a PUT request to
projects/:project_id/merge_requests/:merge_request_iid/notes/:note_id.

delete_merge_request_note

```

    $api->delete_merge_request_note (
        $project_id,
        $merge_request_iid,
        $note_id,
    );

```

Sends a DELETE request to
projects/:project_id/merge_requests/:merge_request_iid/notes/:note_id.

Discussions

See <<https://docs.gitlab.com/ce/api/discussions.html>>.

issue_discussions

```

    my $discussions = $api->issue_discussions (
        $project_id,
        $issue_iid,
        \%params,
    );

```

Sends a GET request to projects/:project_id/issues/:issue_iid/discussions and returns the decoded response content.

issue_discussion

```

    my $discussion = $api->issue_discussion (
        $project_id,
        $issue_iid,
        $discussion_id,
    );

```

Sends a GET request to projects/:project_id/issues/:issue_iid/discussions/:discussion_id and returns the decoded response content.

create_issue_discussion

```

    my $discussion = $api->create_issue_discussion (
        $project_id,
        $issue_iid,
        \%params,
    );

```

Sends a POST request to projects/:project_id/issues/:issue_iid/discussions and returns the decoded response content.

create_issue_discussion_note

```

    $api->create_issue_discussion_note(
        $project_id,
        $issue_iid,
        $discussion_id,
        \%params,
    );

Sends          a          POST          request          to
projects/:project_id/issues/:issue_iid/discussions/:discussion_id/notes.
edit_issue_discussion_note
    $api->edit_issue_discussion_note(
        $project_id,
        $issue_iid,
        $discussion_id,
        $note_id,
        \%params,
    );

Sends          a          PUT          request          to
projects/:project_id/issues/:issue_iid/discussions/:discussion_id/notes/:note_id
delete_issue_discussion_note
    $api->delete_issue_discussion_note(
        $project_id,
        $issue_iid,
        $discussion_id,
        $note_id,
    );

Sends          a          DELETE          request          to
projects/:project_id/issues/:issue_iid/discussions/:discussion_id/notes/:note_id
project_snippet_discussions
    my $discussions = $api->project_snippet_discussions(
        $project_id,
        $snippet_id,
        \%params,
    );

Sends          a          GET          request          to
projects/:project_id/snippets/:snippet_id/discussions and returns the
decoded response content.
project_snippet_discussion
    my $discussion = $api->project_snippet_discussion(
        $project_id,
        $snippet_id,
        $discussion_id,
    );

Sends          a          GET          request          to
projects/:project_id/snippets/:snippet_id/discussions/:discussion_id
and returns the decoded response content.
create_project_snippet_discussion

```

```

my $discussion = $api->create_project_snippet_discussion(
    $project_id,
    $snippet_id,
    \%params,
);

```

Sends a POST request to `projects/:project_id/snippets/:snippet_id/discussions` and returns the decoded response content.

create_project_snippet_discussion_note

```

$api->create_project_snippet_discussion_note(
    $project_id,
    $snippet_id,
    $discussion_id,
    \%params,
);

```

Sends a POST request to `projects/:project_id/snippets/:snippet_id/discussions/:discussion_id/notes`.

edit_project_snippet_discussion_note

```

$api->edit_project_snippet_discussion_note(
    $project_id,
    $snippet_id,
    $discussion_id,
    $note_id,
    \%params,
);

```

Sends a PUT request to `projects/:project_id/snippets/:snippet_id/discussions/:discussion_id/notes/:note_id`.

delete_project_snippet_discussion_note

```

$api->delete_project_snippet_discussion_note(
    $project_id,
    $snippet_id,
    $discussion_id,
    $note_id,
);

```

Sends a DELETE request to `projects/:project_id/snippets/:snippet_id/discussions/:discussion_id/notes/:note_id`.

merge_request_discussions

```

my $discussions = $api->merge_request_discussions(
    $project_id,
    $merge_request_iid,
    \%params,
);

```

Sends a GET request to `projects/:project_id/merge_requests/:merge_request_iid/discussions` and returns the decoded response content.

merge_request_discussion


```

my $discussion = $api->merge_request_discussion(
    $project_id,
    $merge_request_iid,
    $discussion_id,
);

```

Sends a GET request to
 projects/:project_id/merge_requests/:merge_request_iid/discussions/:discussion_id
 and returns the decoded response content.

create_merge_request_discussion

```

my $discussion = $api->create_merge_request_discussion(
    $project_id,
    $merge_request_iid,
    \%params,
);

```

Sends a POST request to
 projects/:project_id/merge_requests/:merge_request_iid/discussions/:discussion_id
 and returns the decoded response content.

resolve_merge_request_discussion

```

$api->resolve_merge_request_discussion(
    $project_id,
    $merge_request_iid,
    $discussion_id,
    \%params,
);

```

Sends a PUT request to
 projects/:project_id/merge_requests/:merge_request_iid/discussions/:discussion_id

create_merge_request_discussion_note

```

$api->create_merge_request_discussion_note(
    $project_id,
    $merge_request_iid,
    $discussion_id,
    \%params,
);

```

Sends a POST request to
 projects/:project_id/merge_requests/:merge_request_iid/discussions/:discussion_id

edit_merge_request_discussion_note

```

$api->edit_merge_request_discussion_note(
    $project_id,
    $merge_request_iid,
    $discussion_id,
    $note_id,
    \%params,
);

```

Sends a PUT request to
 projects/:project_id/merge_requests/:merge_request_iid/discussions/:discussion_id

delete_merge_request_discussion_note

```

    $api->delete_merge_request_discussion_note(
        $project_id,
        $merge_request_iid,
        $discussion_id,
        $note_id,
    );

```

Sends a DELETE request to
 projects/:project_id/merge_requests/:merge_request_iid/discussions/:discussion_id
 commit_discussions

```

    my $discussions = $api->commit_discussions(
        $project_id,
        $commit_id,
        \%params,
    );

```

Sends a GET request to projects/:project_id/commits/:commit_id/discussions
 and returns the decoded response content.

commit_discussion

```

    my $discussion = $api->commit_discussion(
        $project_id,
        $commit_id,
        $discussion_id,
    );

```

Sends a GET request to
 projects/:project_id/commits/:commit_id/discussions/:discussion_id
 and returns the decoded response content.

create_commit_discussion

```

    my $discussion = $api->create_commit_discussion(
        $project_id,
        $commit_id,
        \%params,
    );

```

Sends a POST request to projects/:project_id/commits/:commit_id/discussions
 and returns the decoded response content.

create_commit_discussion_note

```

    $api->create_commit_discussion_note(
        $project_id,
        $commit_id,
        $discussion_id,
        \%params,
    );

```

Sends a POST request to
 projects/:project_id/commits/:commit_id/discussions/:discussion_id/notes.
 edit_commit_discussion_note

```
$api->edit_commit_discussion_note(
    $project_id,
    $commit_id,
    $discussion_id,
    $note_id,
    \%params,
);
```

Sends a PUT request to
projects/:project_id/commits/:commit_id/discussions/:discussion_id/notes/:note_id
delete_commit_discussion_note

```
$api->delete_commit_discussion_note(
    $project_id,
    $commit_id,
    $discussion_id,
    $note_id,
);
```

Sends a DELETE request to
projects/:project_id/commits/:commit_id/discussions/:discussion_id/notes/:note_id

Resource label events

See <https://docs.gitlab.com/ce/api/resource_label_events.html>.

issue_resource_label_events

```
my $events = $api->issue_resource_label_events(
    $project_id,
    $issue_iid,
);
```

Sends a GET request to
projects/:project_id/issues/:issue_iid/resource_label_events and returns
the decoded response content.

issue_resource_label_event

```
my $event = $api->issue_resource_label_event(
    $project_id,
    $issue_iid,
    $resource_label_event_id,
);
```

Sends a GET request to
projects/:project_id/issues/:issue_iid/resource_label_events/:resource_label_event_id
and returns the decoded response content.

merge_request_resource_label_events

```
my $events = $api->merge_request_resource_label_events(
    $project_id,
    $merge_request_iid,
);
```

Sends a GET request to
projects/:project_id/merge_requests/:merge_request_iid/resource_label_events
and returns the decoded response content.

merge_request_resource_label_event

```
my $event = $api->merge_request_resource_label_event (
    $project_id,
    $merge_request_iid,
    $resource_label_event_id,
);
```

Sends a GET request to `projects/:project_id/merge_requests/:merge_request_iid/resource_label_events/:` and returns the decoded response content.

Notification settings

See <https://docs.gitlab.com/ce/api/notification_settings.html>.

global_notification_settings

```
my $settings = $api->global_notification_settings();
```

Sends a GET request to `notification_settings` and returns the decoded response content.

set_global_notification_settings

```
my $settings = $api->set_global_notification_settings (
    \%params,
);
```

Sends a PUT request to `notification_settings` and returns the decoded response content.

group_notification_settings

```
my $settings = $api->group_notification_settings (
    $group_id,
);
```

Sends a GET request to `groups/:group_id/notification_settings` and returns the decoded response content.

project_notification_settings

```
my $settings = $api->project_notification_settings (
    $project_id,
);
```

Sends a GET request to `projects/:project_id/notification_settings` and returns the decoded response content.

set_group_notification_settings

```
my $settings = $api->set_group_notification_settings (
    $group_id,
    \%params,
);
```

Sends a PUT request to `groups/:group_id/notification_settings` and returns the decoded response content.

set_project_notification_settings

```
my $settings = $api->set_project_notification_settings (
    $project_id,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/notification_settings` and returns the decoded response content.

Licenses

See <<https://docs.gitlab.com/ce/api/templates/licenses.html>>.

license_templates

```
my $templates = $api->license_templates(  
    \%params,  
);
```

Sends a GET request to `templates/licenses` and returns the decoded response content.

license_template

```
my $template = $api->license_template(  
    $template_key,  
    \%params,  
);
```

Sends a GET request to `templates/licenses/:template_key` and returns the decoded response content.

Pages domains

See https://docs.gitlab.com/ce/api/pages_domains.html.

global_pages_domains

```
my $domains = $api->global_pages_domains(  
    \%params,  
);
```

Sends a GET request to `pages/domains` and returns the decoded response content.

pages_domains

```
my $domains = $api->pages_domains(  
    $project_id,  
    \%params,  
);
```

Sends a GET request to `projects/:project_id/pages/domains` and returns the decoded response content.

pages_domain

```
my $domain = $api->pages_domain(  
    $project_id,  
    $domain,  
);
```

Sends a GET request to `projects/:project_id/pages/domains/:domain` and returns the decoded response content.

create_pages_domain

```
my $domain = $api->create_pages_domain(  
    $project_id,  
    \%params,  
);
```

Sends a POST request to `projects/:project_id/pages/domains` and returns the decoded response content.

edit_pages_domain

```
my $domain = $api->edit_pages_domain(  
    $project_id,  
    $domain,  
    \%params,  
);
```

Sends a PUT request to `projects/:project_id/pages/domains/:domain` and returns the decoded response content.

```
delete_pages_domain
    $api->delete_pages_domain(
        $project_id,
        $domain,
    );
```

Sends a DELETE request to `projects/:project_id/pages/domains/:domain`.

Pipelines

See <<https://docs.gitlab.com/ce/api/pipelines.html>>.

```
pipelines
    my $pipelines = $api->pipelines(
        $project_id,
        \%params,
    );
```

Sends a GET request to `projects/:project_id/pipelines` and returns the decoded response content.

```
pipeline
    my $pipeline = $api->pipeline(
        $project_id,
        $pipeline_id,
    );
```

Sends a GET request to `projects/:project_id/pipelines/:pipeline_id` and returns the decoded response content.

```
create_pipeline
    my $pipeline = $api->create_pipeline(
        $project_id,
        \%params,
    );
```

Sends a POST request to `projects/:project_id/pipeline` and returns the decoded response content.

Git ref (branch or tag) name must be specified in the `ref` field of the `%params` hash. It's also possible to pass variables to a pipeline in the `variables` field like in the following example:

```
my $pipeline = $api->create_pipeline(
    $project_id,
    {
        'ref' => 'master',
        variables => [
            { 'key' => 'VARIABLE1', 'value' => 'VALUE1' },
            { 'key' => 'VARIABLE2', 'value' => 'VALUE2' },
        ],
    },
);
```

```
retry_pipeline_jobs
    my $pipeline = $api->retry_pipeline_jobs(
        $project_id,
        $pipeline_id,
    );
```

Sends a POST request to `projects/:project_id/pipelines/:pipeline_id/retry` and returns the decoded response content.

cancel_pipeline_jobs

```
my $pipeline = $api->cancel_pipeline_jobs(
    $project_id,
    $pipeline_id,
);
```

Sends a POST request to `projects/:project_id/pipelines/:pipeline_id/cancel` and returns the decoded response content.

Pipeline triggers

See <https://docs.gitlab.com/ce/api/pipeline_triggers.html>.

triggers

```
my $triggers = $api->triggers(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/triggers` and returns the decoded response content.

trigger

```
my $trigger = $api->trigger(
    $project_id,
    $trigger_id,
);
```

Sends a GET request to `projects/:project_id/triggers/:trigger_id` and returns the decoded response content.

create_trigger

```
my $trigger = $api->create_trigger(
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/triggers` and returns the decoded response content.

edit_trigger

```
my $trigger = $api->edit_trigger(
    $project_id,
    $trigger_id,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/triggers/:trigger_id` and returns the decoded response content.

take_ownership_of_trigger

```
my $trigger = $api->take_ownership_of_trigger(
    $project_id,
    $trigger_id,
);
```

Sends a POST request to `projects/:project_id/triggers/:trigger_id/take_ownership` and returns the decoded response content.

delete_trigger

```
$api->delete_trigger(
    $project_id,
    $trigger_id,
);
```

Sends a DELETE request to `projects/:project_id/triggers/:trigger_id`.

`trigger_pipeline`

```
my $pipeline = $api->trigger_pipeline(
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/trigger/pipeline` and returns the decoded response content.

The API authentication token (“private_token” or “access_token” parameters in a constructor) is not needed when using this method, however You must pass trigger token (generated at the trigger creation) as `token` field and git ref name as `ref` field in the `%params` hash. You can also pass variables to be set in a pipeline in the `variables` field. Example:

```
my $pipeline = $api->trigger_pipeline(
    $project_id,
    {
        token => 'd69dba9162ab6ac72fa0993496286ada',
        'ref' => 'master',
        variables => {
            variable1 => 'value1',
            variable2 => 'value2',
        },
    },
);
```

Read more at <<https://docs.gitlab.com/ce/ci/triggers/#triggering-a-pipeline>>.

Pipeline schedules

See <https://docs.gitlab.com/ce/api/pipeline_schedules.html>.

`pipeline_schedules`

```
my $schedules = $api->pipeline_schedules(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/pipeline_schedules` and returns the decoded response content.

`pipeline_schedule`

```
my $schedule = $api->pipeline_schedule(
    $project_id,
    $pipeline_schedule_id,
);
```

Sends a GET request to `projects/:project_id/pipeline_schedules/:pipeline_schedule_id` and returns the decoded response content.

`create_pipeline_schedule`


```
my $schedule = $api->create_pipeline_schedule(
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/pipeline_schedules` and returns the decoded response content.

edit_pipeline_schedule

```
my $schedule = $api->edit_pipeline_schedule(
    $project_id,
    $pipeline_schedule_id,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/pipeline_schedules/:pipeline_schedule_id` and returns the decoded response content.

take_ownership_of_pipeline_schedule

```
my $schedule = $api->take_ownership_of_pipeline_schedule(
    $project_id,
    $pipeline_schedule_id,
);
```

Sends a POST request to `projects/:project_id/pipeline_schedules/:pipeline_schedule_id/take_ownership` and returns the decoded response content.

delete_pipeline_schedule

```
my $schedule = $api->delete_pipeline_schedule(
    $project_id,
    $pipeline_schedule_id,
);
```

Sends a DELETE request to `projects/:project_id/pipeline_schedules/:pipeline_schedule_id` and returns the decoded response content.

create_pipeline_schedule_variable

```
my $variable = $api->create_pipeline_schedule_variable(
    $project_id,
    $pipeline_schedule_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/pipeline_schedules/:pipeline_schedule_id/variables` and returns the decoded response content.

edit_pipeline_schedule_variable

```
my $variable = $api->edit_pipeline_schedule_variable(
    $project_id,
    $pipeline_schedule_id,
    $variable_key,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/pipeline_schedules/:pipeline_schedule_id/variables/:variable_key` and returns the decoded response content.

and returns the decoded response content.

`delete_pipeline_schedule_variable`

```
my $variable = $api->delete_pipeline_schedule_variable(
    $project_id,
    $pipeline_schedule_id,
    $variable_key,
);
```

Sends a `DELETE` request to `projects/:project_id/pipeline_schedules/:pipeline_schedule_id/variables/:variable_key` and returns the decoded response content.

Projects

See <https://docs.gitlab.com/ce/api/projects.html>.

`projects`

```
my $projects = $api->projects(
    \%params,
);
```

Sends a GET request to `projects` and returns the decoded response content.

`user_projects`

```
my $projects = $api->user_projects(
    $user_id,
    \%params,
);
```

Sends a GET request to `users/:user_id/projects` and returns the decoded response content.

`project`

```
my $project = $api->project(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id` and returns the decoded response content.

`project_users`

```
my $users = $api->project_users(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/users` and returns the decoded response content.

`create_project`

```
my $project = $api->create_project(
    \%params,
);
```

Sends a POST request to `projects` and returns the decoded response content.

`create_project_for_user`

```
$api->create_project_for_user(
    $user_id,
    \%params,
);
```

Sends a POST request to `projects/user/:user_id`.

edit_project

```
$api->edit_project (
    $project_id,
    \%params,
);
```

Sends a PUT request to `projects/:project_id`.

fork_project

```
$api->fork_project (
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/fork`.

project_forks

```
my $forks = $api->project_forks (
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/forks` and returns the decoded response content.

start_project

```
my $project = $api->start_project (
    $project_id,
);
```

Sends a POST request to `projects/:project_id/star` and returns the decoded response content.

unstar_project

```
my $project = $api->unstar_project (
    $project_id,
);
```

Sends a POST request to `projects/:project_id/unstar` and returns the decoded response content.

project_languages

```
my $languages = $api->project_languages (
    $project_id,
);
```

Sends a GET request to `projects/:project_id/languages` and returns the decoded response content.

archive_project

```
my $project = $api->archive_project (
    $project_id,
);
```

Sends a POST request to `projects/:project_id/archive` and returns the decoded response content.

unarchive_project

```
my $project = $api->unarchive_project (
    $project_id,
);
```

Sends a POST request to `projects/:project_id/unarchive` and returns the decoded

response content.

delete_project

```
$api->delete_project (
    $project_id,
);
```

Sends a DELETE request to `projects/:project_id`.

upload_file_to_project

```
my $upload = $api->upload_file_to_project (
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/uploads` and returns the decoded response content.

The `file` parameter must point to a readable file on the local filesystem.

share_project_with_group

```
$api->share_project_with_group (
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/share`.

unshare_project_with_group

```
$api->unshare_project_with_group (
    $project_id,
    $group_id,
);
```

Sends a DELETE request to `projects/:project_id/share/:group_id`.

project_hooks

```
my $hooks = $api->project_hooks (
    $project_id,
);
```

Sends a GET request to `projects/:project_id/hooks` and returns the decoded response content.

project_hook

```
my $hook = $api->project_hook (
    $project_id,
    $hook_id,
);
```

Sends a GET request to `projects/:project_id/hooks/:hook_id` and returns the decoded response content.

create_project_hook

```
my $hook = $api->create_project_hook (
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/hooks` and returns the decoded response content.

edit_project_hook

```
my $hook = $api->edit_project_hook (
    $project_id,
    $hook_id,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/hooks/:hook_id` and returns the decoded response content.

delete_project_hook

```
$api->delete_project_hook (
    $project_id,
    $hook_id,
);
```

Sends a DELETE request to `projects/:project_id/hooks/:hook_id`.

set_project_fork

```
$api->set_project_fork (
    $project_id,
    $from_project_id,
);
```

Sends a POST request to `projects/:project_id/fork/:from_project_id`.

clear_project_fork

```
$api->clear_project_fork (
    $project_id,
);
```

Sends a DELETE request to `projects/:project_id/fork`.

start_housekeeping

```
$api->start_housekeeping (
    $project_id,
);
```

Sends a POST request to `projects/:project_id/housekeeping`.

transfer_project_to_namespace

```
$api->transfer_project_to_namespace (
    $project_id,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/transfer`.

Project access requests

See https://docs.gitlab.com/ce/api/access_requests.html.

project_access_requests

```
my $requests = $api->project_access_requests (
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/access_requests` and returns the decoded response content.

request_project_access

```
my $request = $api->request_project_access(
    $project_id,
);
```

Sends a POST request to `projects/:project_id/access_requests` and returns the decoded response content.

approve_project_access

```
my $request = $api->approve_project_access(
    $project_id,
    $user_id,
);
```

Sends a PUT request to `projects/:project_id/access_requests/:user_id/approve` and returns the decoded response content.

deny_project_access

```
$api->deny_project_access(
    $project_id,
    $user_id,
);
```

Sends a DELETE request to `projects/:project_id/access_requests/:user_id`.

Project badges

See https://docs.gitlab.com/ce/api/project_badges.html.

project_badges

```
my $badges = $api->project_badges(
    $project_id,
);
```

Sends a GET request to `projects/:project_id/badges` and returns the decoded response content.

project_badge

```
my $badge = $api->project_badge(
    $project_id,
    $badge_id,
);
```

Sends a GET request to `projects/:project_id/badges/:badge_id` and returns the decoded response content.

create_project_badge

```
my $badge = $api->create_project_badge(
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/badges` and returns the decoded response content.

edit_project_badge

```
my $badge = $api->edit_project_badge(
    $project_id,
    $badge_id,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/badges/:badge_id` and returns the

decoded response content.

delete_project_badge

```
$api->delete_project_badge(
    $project_id,
    $badge_id,
);
```

Sends a DELETE request to `projects/:project_id/badges/:badge_id`.

preview_project_badge

```
my $preview = $api->preview_project_badge(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/badges/render` and returns the decoded response content.

Project import/export

See https://docs.gitlab.com/ce/api/project_import_export.html.

schedule_project_export

```
$api->schedule_project_export(
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/export`.

project_export_status

```
my $status = $api->project_export_status(
    $project_id,
);
```

Sends a GET request to `projects/:project_id/export` and returns the decoded response content.

download_project_export

```
my $download = $api->download_project_export(
    $project_id,
);
```

Sends a GET request to `projects/:project_id/export/download` and returns the decoded response content.

schedule_project_import

```
$api->schedule_project_import(
    \%params,
);
```

Sends a POST request to `projects/import`.

project_import_status

```
my $status = $api->project_import_status(
    $project_id,
);
```

Sends a GET request to `projects/:project_id/import` and returns the decoded response content.

Project members

See <<https://docs.gitlab.com/ce/api/members.html>>.

project_members

```
my $members = $api->project_members (
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/members` and returns the decoded response content.

all_project_members

```
my $members = $api->all_project_members (
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/members/all` and returns the decoded response content.

project_member

```
my $member = $api->project_member (
    $project_id,
    $user_id,
);
```

Sends a GET request to `projects/:project_id/members/:user_id` and returns the decoded response content.

add_project_member

```
my $member = $api->add_project_member (
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/members` and returns the decoded response content.

update_project_member

```
my $member = $api->update_project_member (
    $project_id,
    $user_id,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/members/:user_id` and returns the decoded response content.

remove_project_member

```
$api->remove_project_member (
    $project_id,
    $user_id,
);
```

Sends a DELETE request to `projects/:project_id/members/:user_id`.

Project snippets

See <https://docs.gitlab.com/ce/api/project_snippets.html>.

project_snippets

```
my $snippets = $api->project_snippets (
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/snippets` and returns the decoded response content.

project_snippet

```
my $snippet = $api->project_snippet (
    $project_id,
    $snippet_id,
);
```

Sends a GET request to `projects/:project_id/snippets/:snippet_id` and returns the decoded response content.

create_project_snippet

```
$api->create_project_snippet (
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/snippets`.

edit_project_snippet

```
$api->edit_project_snippet (
    $project_id,
    $snippet_id,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/snippets/:snippet_id`.

delete_project_snippet

```
$api->delete_project_snippet (
    $project_id,
    $snippet_id,
);
```

Sends a DELETE request to `projects/:project_id/snippets/:snippet_id`.

project_snippet_content

```
my $content = $api->project_snippet_content (
    $project_id,
    $snippet_id,
);
```

Sends a GET request to `projects/:project_id/snippets/:snippet_id/raw` and returns the decoded response content.

project_snippet_user_agent_detail

```
my $user_agent = $api->project_snippet_user_agent_detail (
    $project_id,
    $snippet_id,
);
```

Sends a GET request to `projects/:project_id/snippets/:snippet_id/user_agent_detail` and returns the decoded response content.

Protected branches

See <https://docs.gitlab.com/ce/api/protected_branches.html>.

protected_branches

```
my $branches = $api->protected_branches(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/protected_branches` and returns the decoded response content.

protected_branch

```
my $branch = $api->protected_branch(
    $project_id,
    $branch_name,
);
```

Sends a GET request to `projects/:project_id/protected_branches/:branch_name` and returns the decoded response content.

protect_branch

```
my $branch = $api->protect_branch(
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/protected_branches` and returns the decoded response content.

unprotect_branch

```
$api->unprotect_branch(
    $project_id,
    $branch_name,
);
```

Sends a DELETE request to `projects/:project_id/protected_branches/:branch_name`.

Protected tags

See <https://docs.gitlab.com/ce/api/protected_tags.html>.

protected_tags

```
my $tags = $api->protected_tags(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/protected_tags` and returns the decoded response content.

protected_tag

```
my $tag = $api->protected_tag(
    $project_id,
    $tag_name,
);
```

Sends a GET request to `projects/:project_id/protected_tags/:tag_name` and returns the decoded response content.

protect_tag

```
my $tag = $api->protect_tag(  
    $project_id,  
    \%params,  
);
```

Sends a POST request to `projects/:project_id/protected_tags` and returns the decoded response content.

unprotect_tag

```
$api->unprotect_tag(  
    $project_id,  
    $tag_name,  
);
```

Sends a DELETE request to `projects/:project_id/protected_tags/:tag_name`.

Releases

See <https://docs.gitlab.com/ce/api/releases/index.html>.

releases

```
my $releases = $api->releases(  
    $project_id,  
    \%params,  
);
```

Sends a GET request to `projects/:project_id/releases` and returns the decoded response content.

release

```
my $release = $api->release(  
    $project_id,  
    $tag_name,  
);
```

Sends a GET request to `projects/:project_id/releases/:tag_name` and returns the decoded response content.

create_release

```
my $release = $api->create_release(  
    $project_id,  
    \%params,  
);
```

Sends a POST request to `projects/:project_id/releases` and returns the decoded response content.

update_release

```
my $release = $api->update_release(  
    $project_id,  
    $tag_name,  
    \%params,  
);
```

Sends a PUT request to `projects/:project_id/releases/:tag_name` and returns the decoded response content.

delete_release

```
my $release = $api->delete_release(
    $project_id,
    $tag_name,
);
```

Sends a DELETE request to `projects/:project_id/releases/:tag_name` and returns the decoded response content.

Release Links

See <<https://docs.gitlab.com/ce/api/releases/links.html>>.

release_links

```
my $links = $api->release_links(
    $project_id,
    $tag_name,
    \%params,
);
```

Sends a GET request to `projects/:project_id/releases/:tag_name/assets/links` and returns the decoded response content.

release_link

```
my $link = $api->release_link(
    $project_id,
    $tag_name,
    $link_id,
);
```

Sends a GET request to `projects/:project_id/releases/:tag_name/assets/links/:link_id` and returns the decoded response content.

create_release_link

```
my $link = $api->create_release_link(
    $project_id,
    $tag_name,
    \%params,
);
```

Sends a POST request to `projects/:project_id/releases/:tag_name/assets/links` and returns the decoded response content.

update_release_link

```
my $link = $api->update_release_link(
    $project_id,
    $tag_name,
    $link_id,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/releases/:tag_name/assets/links/:link_id` and returns the decoded response content.

delete_release_link

```
my $link = $api->delete_release_link(
    $project_id,
    $tag_name,
    $link_id,
);
```

Sends a DELETE request to `projects/:project_id/releases/:tag_name/assets/links/:link_id` and returns the decoded response content.

Repositories

See <<https://docs.gitlab.com/ce/api/repositories.html>>.

tree

```
my $tree = $api->tree(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/repository/tree` and returns the decoded response content.

blob

```
my $blob = $api->blob(
    $project_id,
    $sha,
);
```

Sends a GET request to `projects/:project_id/repository/blobs/:sha` and returns the decoded response content.

raw_blob

```
my $raw_blob = $api->raw_blob(
    $project_id,
    $sha,
);
```

Sends a GET request to `projects/:project_id/repository/blobs/:sha/raw` and returns the decoded response content.

archive

```
my $archive = $api->archive(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/repository/archive` and returns the raw response content.

compare

```
my $comparison = $api->compare(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/repository/compare` and returns the decoded response content.

contributors

```
my $contributors = $api->contributors(  
    $project_id,  
    \%params,  
);
```

Sends a GET request to `projects/:project_id/repository/contributors` and returns the decoded response content.

Repository files

See <https://docs.gitlab.com/ce/api/repository_files.html>.

file

```
my $file = $api->file(  
    $project_id,  
    $file_path,  
    \%params,  
);
```

Sends a GET request to `projects/:project_id/repository/files/:file_path` and returns the decoded response content.

raw_file

```
my $content = $api->raw_file(  
    $project_id,  
    $file_path,  
    \%params,  
);
```

Sends a GET request to `projects/:project_id/repository/files/:file_path/raw` and returns the raw response content.

create_file

```
$api->create_file(  
    $project_id,  
    $file_path,  
    \%params,  
);
```

Sends a POST request to `projects/:project_id/repository/files/:file_path`.

edit_file

```
$api->edit_file(  
    $project_id,  
    $file_path,  
    \%params,  
);
```

Sends a PUT request to `projects/:project_id/repository/files/:file_path`.

delete_file

```
$api->delete_file(  
    $project_id,  
    $file_path,  
    \%params,  
);
```

Sends a DELETE request to `projects/:project_id/repository/files/:file_path`.

Runners

See <<https://docs.gitlab.com/ce/api/runners.html>>.

runners

```
my $runners = $api->runners (
    \%params,
);
```

Sends a GET request to runners and returns the decoded response content.

all_runners

```
my $runners = $api->all_runners (
    \%params,
);
```

Sends a GET request to runners/all and returns the decoded response content.

runner

```
my $runner = $api->runner (
    $runner_id,
);
```

Sends a GET request to runners/:runner_id and returns the decoded response content.

update_runner

```
my $runner = $api->update_runner (
    $runner_id,
    \%params,
);
```

Sends a PUT request to runners/:runner_id and returns the decoded response content.

delete_runner

```
my $runner = $api->delete_runner (
    $runner_id,
);
```

Sends a DELETE request to runners/:runner_id and returns the decoded response content.

runner_jobs

```
my $jobs = $api->runner_jobs (
    $runner_id,
    \%params,
);
```

Sends a GET request to runners/:runner_id/jobs and returns the decoded response content.

project_runners

```
my $runners = $api->project_runners (
    $project_id,
    \%params,
);
```

Sends a GET request to projects/:project_id/runners and returns the decoded response content.

enable_project_runner

```
my $runner = $api->enable_project_runner (
    $project_id,
    \%params,
);
```

Sends a POST request to projects/:project_id/runners and returns the decoded response content.

disable_project_runner

```
my $runner = $api->disable_project_runner(
    $project_id,
    $runner_id,
);
```

Sends a DELETE request to `projects/:project_id/runners/:runner_id` and returns the decoded response content.

Search

See <<https://docs.gitlab.com/ce/api/search.html>>.

search

```
my $results = $api->search(
    \%params,
);
```

Sends a GET request to `search` and returns the decoded response content.

Services

See <<https://docs.gitlab.com/ce/api/services.html>>.

project_service

```
my $service = $api->project_service(
    $project_id,
    $service_name,
);
```

Sends a GET request to `projects/:project_id/services/:service_name` and returns the decoded response content.

edit_project_service

```
$api->edit_project_service(
    $project_id,
    $service_name,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/services/:service_name`.

delete_project_service

```
$api->delete_project_service(
    $project_id,
    $service_name,
);
```

Sends a DELETE request to `projects/:project_id/services/:service_name`.

Application settings

See <<https://docs.gitlab.com/ce/api/settings.html>>.

settings

```
my $settings = $api->settings();
```

Sends a GET request to `application/settings` and returns the decoded response content.

update_settings

```
my $settings = $api->update_settings(
    \%params,
);
```

Sends a PUT request to `application/settings` and returns the decoded response content.

Sidekiq Metrics

See <https://docs.gitlab.com/ce/api/sidekiq_metrics.html>.

queue_metrics

```
my $metrics = $api->queue_metrics();
```

Sends a GET request to `sidekiq/queue_metrics` and returns the decoded response content.

process_metrics

```
my $metrics = $api->process_metrics();
```

Sends a GET request to `sidekiq/process_metrics` and returns the decoded response content.

job_stats

```
my $stats = $api->job_stats();
```

Sends a GET request to `sidekiq/job_stats` and returns the decoded response content.

compound_metrics

```
my $metrics = $api->compound_metrics();
```

Sends a GET request to `sidekiq/compound_metrics` and returns the decoded response content.

System hooks

See <https://docs.gitlab.com/ce/api/system_hooks.html>.

hooks

```
my $hooks = $api->hooks(
    \%params,
);
```

Sends a GET request to `hooks` and returns the decoded response content.

create_hook

```
$api->create_hook(
    \%params,
);
```

Sends a POST request to `hooks`.

test_hook

```
my $hook = $api->test_hook(
    $hook_id,
);
```

Sends a GET request to `hooks/:hook_id` and returns the decoded response content.

delete_hook

```
$api->delete_hook(
    $hook_id,
);
```

Sends a DELETE request to `hooks/:hook_id`.

Tags

See <<https://docs.gitlab.com/ce/api/tags.html>>.

tags

```
my $tags = $api->tags(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/repository/tags` and returns the decoded response content.

tag

```
my $tag = $api->tag(
    $project_id,
    $tag_name,
);
```

Sends a GET request to `projects/:project_id/repository/tags/:tag_name` and returns the decoded response content.

create_tag

```
my $tag = $api->create_tag(
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/repository/tags` and returns the decoded response content.

delete_tag

```
$api->delete_tag(
    $project_id,
    $tag_name,
);
```

Sends a DELETE request to `projects/:project_id/repository/tags/:tag_name`.

create_tag_release

```
my $release = $api->create_tag_release(
    $project_id,
    $tag_name,
    \%params,
);
```

Sends a POST request to `projects/:project_id/repository/tags/:tag_name/release` and returns the decoded response content.

update_tag_release

```
my $release = $api->update_tag_release(
    $project_id,
    $tag_name,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/repository/tags/:tag_name/release` and returns the decoded response content.

Todos

See <<https://docs.gitlab.com/ce/api/todos.html>>.

todos

```
my $todos = $api->todos(
    \%params,
);
```

Sends a GET request to `todos` and returns the decoded response content.

mark_todo_done

```
my $todo = $api->mark_todo_done(
    $todo_id,
);
```

Sends a POST request to `todos/:todo_id/mark_as_done` and returns the decoded response content.

`mark_all_todos_done`

```
$api->mark_all_todos_done();
```

Sends a POST request to `todos/mark_as_done`.

Users

See <https://docs.gitlab.com/ce/api/users.html>.

`users`

```
my $users = $api->users(
    \%params,
);
```

Sends a GET request to `users` and returns the decoded response content.

`user`

```
my $user = $api->user(
    $user_id,
);
```

Sends a GET request to `users/:user_id` and returns the decoded response content.

`create_user`

```
$api->create_user(
    \%params,
);
```

Sends a POST request to `users`.

`edit_user`

```
$api->edit_user(
    $user_id,
    \%params,
);
```

Sends a PUT request to `users/:user_id`.

`delete_user`

```
$api->delete_user(
    $user_id,
);
```

Sends a DELETE request to `users/:user_id`.

`current_user`

```
my $user = $api->current_user();
```

Sends a GET request to `user` and returns the decoded response content.

`current_user_ssh_keys`

```
my $keys = $api->current_user_ssh_keys(
    \%params,
);
```

Sends a GET request to `user/keys` and returns the decoded response content.

user_ssh_keys

```
my $keys = $api->user_ssh_keys (
    $user_id,
    \%params,
);
```

Sends a GET request to `users/:user_id/keys` and returns the decoded response content.

user_ssh_key

```
my $key = $api->user_ssh_key (
    $key_id,
);
```

Sends a GET request to `user/keys/:key_id` and returns the decoded response content.

create_current_user_ssh_key

```
$api->create_current_user_ssh_key (
    \%params,
);
```

Sends a POST request to `user/keys`.

create_user_ssh_key

```
$api->create_user_ssh_key (
    $user_id,
    \%params,
);
```

Sends a POST request to `users/:user_id/keys`.

delete_current_user_ssh_key

```
$api->delete_current_user_ssh_key (
    $key_id,
);
```

Sends a DELETE request to `user/keys/:key_id`.

delete_user_ssh_key

```
$api->delete_user_ssh_key (
    $user_id,
    $key_id,
);
```

Sends a DELETE request to `users/:user_id/keys/:key_id`.

current_user_gpg_keys

```
my $keys = $api->current_user_gpg_keys (
    \%params,
);
```

Sends a GET request to `user/gpg_keys` and returns the decoded response content.

current_user_gpg_key

```
my $key = $api->current_user_gpg_key (
    $key_id,
);
```

Sends a GET request to `user/gpg_keys/:key_id` and returns the decoded response content.

create_current_user_gpg_key

```
$api->create_current_user_gpg_key(
    \%params,
);
```

Sends a POST request to `user/gpg_keys`.

```
delete_current_user_gpg_key
    $api->delete_current_user_gpg_key(
        $key_id,
    );
```

Sends a DELETE request to `user/gpg_keys/:key_id`.

```
user_gpg_keys
    my $keys = $api->user_gpg_keys(
        $user_id,
        \%params,
    );
```

Sends a GET request to `users/:user_id/gpg_keys` and returns the decoded response content.

```
user_gpg_key
    my $key = $api->user_gpg_key(
        $user_id,
        $key_id,
    );
```

Sends a GET request to `users/:user_id/gpg_keys/:key_id` and returns the decoded response content.

```
create_user_gpg_key
    my $keys = $api->create_user_gpg_key(
        $user_id,
        \%params,
    );
```

Sends a POST request to `users/:user_id/gpg_keys` and returns the decoded response content.

```
delete_user_gpg_key
    $api->delete_user_gpg_key(
        $user_id,
        $key_id,
    );
```

Sends a DELETE request to `users/:user_id/gpg_keys/:key_id`.

```
current_user_emails
    my $emails = $api->current_user_emails(
        \%params,
    );
```

Sends a GET request to `user/emails` and returns the decoded response content.

```
user_emails
    my $emails = $api->user_emails(
        $user_id,
        \%params,
    );
```

Sends a GET request to `users/:user_id/emails` and returns the decoded response content.

current_user_email

```
my $email = $api->current_user_email(  
    $email_id,  
);
```

Sends a GET request to `user/emails/:email_id` and returns the decoded response content.

create_current_user_email

```
my $email = $api->create_current_user_email(  
    \%params,  
);
```

Sends a POST request to `user/emails` and returns the decoded response content.

create_user_email

```
my $email = $api->create_user_email(  
    $user_id,  
    \%params,  
);
```

Sends a POST request to `users/:user_id/emails` and returns the decoded response content.

delete_current_user_email

```
$api->delete_current_user_email(  
    $email_id,  
);
```

Sends a DELETE request to `user/emails/:email_id`.

delete_user_email

```
$api->delete_user_email(  
    $user_id,  
    $email_id,  
);
```

Sends a DELETE request to `users/:user_id/emails/:email_id`.

block_user

```
my $success = $api->block_user(  
    $user_id,  
);
```

Sends a POST request to `users/:user_id/block` and returns the decoded response content.

unblock_user

```
my $success = $api->unblock_user(  
    $user_id,  
);
```

Sends a POST request to `users/:user_id/unblock` and returns the decoded response content.

user_impersonation_tokens

```
my $tokens = $api->user_impersonation_tokens(  
    $user_id,  
    \%params,  
);
```

Sends a GET request to `users/:user_id/impersonation_tokens` and returns the decoded response content.

user_impersonation_token

```
my $token = $api->user_impersonation_token(
    $user_id,
    $impersonation_token_id,
);
```

Sends a GET request to `users/:user_id/impersonation_tokens/:impersonation_token_id` and returns the decoded response content.

create_user_impersonation_token

```
my $token = $api->create_user_impersonation_token(
    $user_id,
    \%params,
);
```

Sends a POST request to `users/:user_id/impersonation_tokens` and returns the decoded response content.

delete_user_impersonation_token

```
$api->delete_user_impersonation_token(
    $user_id,
    $impersonation_token_id,
);
```

Sends a DELETE request to `users/:user_id/impersonation_tokens/:impersonation_token_id`.

all_user_activities

```
my $activities = $api->all_user_activities(
    \%params,
);
```

Sends a GET request to `user/activities` and returns the decoded response content.

Validate the `.gitlab-ci.yml`

See <<https://docs.gitlab.com/ce/api/lint.html>>.

lint

```
my $result = $api->lint(
    \%params,
);
```

Sends a POST request to `lint` and returns the decoded response content.

Version

See <<https://docs.gitlab.com/ce/api/version.html>>.

version

```
my $version = $api->version();
```

Sends a GET request to `version` and returns the decoded response content.

Wikis

See <<https://docs.gitlab.com/ce/api/wikis.html>>.

wiki_pages

```
my $pages = $api->wiki_pages(
    $project_id,
    \%params,
);
```

Sends a GET request to `projects/:project_id/wikis` and returns the decoded response content.

wiki_page

```
my $pages = $api->wiki_page(
    $project_id,
    $slug,
);
```

Sends a GET request to `projects/:project_id/wikis/:slug` and returns the decoded response content.

create_wiki_page

```
my $page = $api->create_wiki_page(
    $project_id,
    \%params,
);
```

Sends a POST request to `projects/:project_id/wikis` and returns the decoded response content.

edit_wiki_page

```
my $page = $api->edit_wiki_page(
    $project_id,
    $slug,
    \%params,
);
```

Sends a PUT request to `projects/:project_id/wikis/:slug` and returns the decoded response content.

delete_wiki_page

```
$api->delete_wiki_page(
    $project_id,
    $slug,
);
```

Sends a DELETE request to `projects/:project_id/wikis/:slug`.

CONTRIBUTING

This module is auto-generated from a set of YAML files defining the interface of GitLab's API. If you'd like to contribute to this module then please feel free to make a fork on GitHub <<https://github.com/bluefeet/GitLab-API-v4>> and submit a pull request, just make sure you edit the files in the `authors/` directory instead of `lib/GitLab/API/v4.pm` directly.

Please see <<https://github.com/bluefeet/GitLab-API-v4/blob/master/author/README.pod>> for more information.

Alternatively, you can open a ticket <<https://github.com/bluefeet/GitLab-API-v4/issues>>.

SUPPORT

Please submit bugs and feature requests to the GitLab-API-v4 GitHub issue tracker:

<<https://github.com/bluefeet/GitLab-API-v4/issues>>

ACKNOWLEDGEMENTS

Thanks to ZipRecruiter <<https://www.ziprecruiter.com/>> for encouraging their employees to contribute back to the open source ecosystem. Without their dedication to quality software development this distribution would not exist.

AUTHORS

Aran Clary Deltac <bluefeet@gmail.com>
Dotan Dimet <dotan@corky.net>
Nigel Gregoire <nigelgregoire@gmail.com>
trunov-ms <trunov.ms@gmail.com>
Marek R. Sotola <Marek.R.Sotola@nasa.gov>
José Joaquín Atria <jjatria@gmail.com>
Dave Webb <github@d5ve.com>
Simon Ruderich <simon@ruderich.org>
royce55 <royce@ecs.vuw.ac.nz>
gregor herrmann <gregoa@debian.org>
Luc Didry <luc@framasoftware.org>
Kieren Diment <kieren.diment@staples.com.au>
Dmitry Frolov <dmitry.frolov@gmail.com>

COPYRIGHT AND LICENSE

Copyright (C) 2014 Aran Clary Deltac

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.