Package 'gitlabr'

April 25, 2017

Title Access to the Gitlab API

Version 0.9

Description Provides R functions to access the API of the project and repository management web application gitlab. For many common tasks (repository file access, issue assignment and status, commenting) convenience wrappers are provided, and in addition the full API can be used by specifying request locations. Gitlab is open-source software and can be self-hosted or used on gitlab.com.

Depends R (>= 3.1.2), magrittr

Imports dplyr (>= 0.4.3), stringr, base64enc, httr (>= 1.1.0), purrr (>= 0.2.2), tibble (>= 1.1), utils, yaml

Suggests devtools (>= 1.8.0), roxygen2, testthat (>= 1.0.2), R.rsp, knitr, shiny (>= 0.13.0)

URL https://blog.points-of-interest.cc/post/gitlabr/

http://doc.gitlab.com/ce/api/

BugReports https://gitlab.points-of-interest.cc/points-of-interest/gitlabr/issues/

VignetteBuilder R.rsp

License GPL (>= 3)

LazyData true

RoxygenNote 5.0.1

NeedsCompilation no

Author Jirka Lewandowski [aut, cre]

Maintainer Jirka Lewandowski <jirka.lewandowski@wzb.eu>

Repository CRAN

Date/Publication 2017-04-25 21:38:36 UTC

R topics documented:

`gitlab` *Request Gitlab API*

Description

This is gitlabr's core function to talk to Gitlab's server API via HTTP(S). Usually you will not use this function directly too often, but either use gitlabr's convenience wrappers or write your own. See the `gitlabr` vignette for more information on this.

Usage

```
gitlab(req, api_root, verb = httr::GET, auto_format = TRUE, debug = FALSE,
  gitlab_con = "default", page = "all", enforce_api_root = TRUE,
  argname_verb = if (identical(verb, httr::GET) | identical(verb,
  httr::DELETE)) {     "query" } else {     "body" }, ...)
```

Arguments

req	vector of characters that represents the call (e.g. c("projects", project_id, "events"))
api_root	URL where the gitlab API to request resides (e.g. https://gitlab.myserver.com/api/v3/)
verb	http verb to use for request in form of one of the httr functions GET, PUT, POST, DELETE
auto_format	whether to format the returned object automatically to a flat data.frame
debug	if TRUE API URL and query will be printed, defaults to FALSE
gitlab_con	function to use for issuing API requests (e.g. as returned by gitlab_connection
page	number of page of API response to get; if "all" (default), all pages are queried successively and combined.
enforce_api_root	if multiple pages are requested, the API root URL is ensured to be the same as in the original call for all calls using the "next page" URL returned by gitlab. This makes sense for security and in cases where gitlab is behind a reverse proxy and ignorant about its URL from external.
argname_verb	name of the argument of the verb that fields and information are passed on to
...	named parameters to pass on to gitlab API (technically: modifies query parameters of request URL), may include private_token and all other parameters as documented for the Gitlab API

Details

Note: currently gitlab API v3 is supported. Support for Gitlab API v4 (for Gitlab version >= 9.0) will be added soon.

Value

the response from the Gitlab API, usually as a ‘data_frame‘ and including all pages

Examples

```
## Not run:
gitlab(req = "projects",
       api_root = "https://gitlab.example.com/api/v4/",
       private_token = "123####89")
gitlab(req = c("projects", 21, "issues"),
       state = "closed",
       gitlab_con = my_gitlab)

## End(Not run)
```

gitlabr *Interface to gitlab API on high and low levels*

Description

gitlabr R package

Details

Interface to gitlab API on high and low levels

Package:	gitlabr
Type:	Package
License:	GPL (>= 3)
LazyLoad:	yes

Author(s)

Jirka Lewandowski <jirka.lewandowski@wzb.eu>

References

http://blog.points-of-interest.cc/

gitlabr-deprecated *Deprecated functions*

Description

Many functions were renamed with version 0.7 to the gl_ naming scheme. Note that the old function names will be removed with version 1.0, expected to be released in 2017.

Usage

```
archive(...)

assign_issue(...)

close_issue(...)

comment_commit(...)

comment_issue(...)
```

create_branch(...)

create_merge_request(...)

delete_branch(...)

edit_commit_comment(...)

edit_issue(...)

edit_issue_comment(...)

file_exists(...)

get_comments(...)

get_commit_comments(...)

get_commits(...)

get_diff(...)

get_file(...)

get_issue(...)

get_issue_comments(...)

get_issues(...)

get_project_id(...)

gitlab_connection(...)

list_branches(...)

list_files(...)

list_projects(...)

new_issue(...)

project_connection(...)

proj_req(...)

push_file(...)

```
reopen_issue(...)

repository(...)

to_issue_id(...)

unassign_issue(...)
```

Arguments

...	Parameters to the new function

Details

archive	is now called gl_archive
assign_issue	is now called gl_assign_issue
close_issue	is now called gl_close_issue
comment_commit	is now called gl_comment_commit
comment_issue	is now called gl_comment_issue
create_branch	is now called gl_create_branch
create_merge_request	is now called gl_create_merge_request
delete_branch	is now called gl_delete_branch
edit_commit_comment	is now called gl_edit_commit_comment
edit_issue	is now called gl_edit_issue
edit_issue_comment	is now called gl_edit_issue_comment
file_exists	is now called gl_file_exists
get_comments	is now called gl_get_comments
get_commit_comments	is now called gl_get_commit_comments
get_commits	is now called gl_get_commits
get_diff	is now called gl_get_diff
get_file	is now called gl_get_file
get_issue	is now called gl_get_issue
get_issue_comments	is now called gl_get_issue_comments
get_issues	is now called gl_list_issues
get_project_id	is now called gl_get_project_id
gitlab_connection	is now called gl_connection
list_branches	is now called gl_list_branches
list_files	is now called gl_list_files
list_projects	is now called gl_list_projects
new_issue	is now called gl_new_issue
project_connection	is now called gl_project_connection
proj_req	is now called gl_proj_req
push_file	is now called gl_push_file
reopen_issue	is now called gl_reopen_issue
repository	is now called gl_repository
to_issue_id	is now called gl_to_issue_id
unassign_issue	is now called gl_unassign_issue

gitlabr_0_7_renaming *renamings from gitlabr version 0.6.4 to 0.7*

Description

List of of old and new function name. Used internally by `update_gitlabr_code`

Format

A data frame with 33 rows and 2 variables

glLoginInput *Shiny module to login to gitlab API*

Description

The UI contains a login and a password field as well as an (optional) login button. The server side function returns a reactive gitlab connection, just as `gl_connection` and `gl_project_connection`.

Usage

```
glLoginInput(id, login_button = TRUE)

glReactiveLogin(input, output, session, gitlab_url, project = NULL,
  success_message = "Gitlab login successful!",
  failure_message = "Gitlab login failed!", on_error = function(...) {
  stop(failure_message) })
```

Arguments

`id`	shiny namespace for the login module
`login_button`	whether to show a login button (TRUE) or be purely reactive (FALSE)
`input`	from shinyServer function, usually not user provided
`output`	from shinyServer function, usually not user provided
`session`	from shinyServer function, usually not user provided
`gitlab_url`	root URL of gitlab instance to login to
`project`	if not NULL, a codegl_project_connection is created to this project
`success_message`	message text to be displayed in the UI on sucessful login
`failure_message`	message text to be displayed in the UI on login failure in addition to HTTP status
`on_error`	function to be returned instead of gitlab connection in case of login failure

Details

glLoginInput is supposed to be used inside a shinyUI, while glReactiveLogin is supposed to be passed on to callModule

---

gl_archive *Get zip archive of a specific repository*

Description

Get zip archive of a specific repository

Usage

```
gl_archive(project, save_to_file = tempfile(fileext = ".zip"), ...)
```

Arguments

project	Project name or id
save_to_file	path where to save archive; if this is NULL, the archive itself is returned as a raw vector
...	further parameters passed on to gitlab API call, may include parameter sha for specifying a commit hash

Value

if save_to_file is NULL, a raw vector of the archive, else the path to the saved archived file

Examples

```
## Not run:
my_project <- gl_project_connection(project = "example-project", ...) ## fill in login parameters
my_project(gl_archive, save_to_file = "example-project.zip")

## End(Not run)
```

gl_ci_job *Define Gitlab CI jobs*

Description

Define Gitlab CI jobs

Usage

```
gl_ci_job(job_name, stage = job_name, allowed_dependencies = c(), ...)

gl_default_ci_pipeline()

use_gitlab_ci(pipeline = gl_default_ci_pipeline(),
  image = "rocker/r-devel:latest", push_to_remotes = c(),
  path = ".gitlab-ci.yml", overwrite = TRUE, add_to_Rbuildignore = TRUE)
```

Arguments

job_name	Name of job template to get CI definition elements
stage	Name of stage job belongs to
allowed_dependencies	List of job names that are allowed to be listed as dependencies of jobs. Usually this is all existing other jobs.
...	passed on to ci_r_script: booleans vanilla or slave translate to R executable options with the same name
pipeline	a CI pipeline defined as a list of lists
image	Docker image to use in gitlab ci. If NULL, not specified!
push_to_remotes	named list of remotes the code should be pushed to. Only master is pushed and for every remote a job of stage "push" is generated. See example for how to use credentials from environment variables.
path	destination path for writing gitlab CI yml file
overwrite	whether to overwrite existing gitlab CI yml file
add_to_Rbuildignore	add CI yml file (from path) to .Rbuildignore?

Examples

```
gl_ci_job("build", allowed_dependencies = "test")
use_gitlab_ci(image = "pointsofinterest/gitlabr:latest")
use_gitlab_ci(image = "pointsofinterest/gitlabr:latest",
 push_to_remotes = list("github" =
 "https://${GITHUB_USERNAME}:${GITHUB_PASSWORD}@github.com/jirkalewandowski/gitlabr.git"))
```

gl_connection *Connect to a specific gitlab instance API*

Description

Creates a function that can be used to issue requests to the specified gitlab API instance with the specified user private token and (for `gl_project_connection`) only to a specified project.

Usage

```
gl_connection(gitlab_url, login = NULL, email = NULL, password = NULL,
  private_token = NULL, api_version = "v4", api_location = paste0("/api/",
  api_version, "/"))

gl_project_connection(gitlab_url, project, login = NULL, email = NULL,
  password = NULL, private_token = NULL, api_version = "v4",
  api_location = paste0("/api/", api_version, "/"))
```

Arguments

`gitlab_url`	URL to the gitlab instance (e.g. `https://gitlab.myserver.com`)
`login`	name of user to login; either this or email or private token must be specified
`email`	email of user to login; either this or login or private token must be specified
`password`	password of user to login; if no private token but login or email is given, this must be specified
`private_token`	private_token with which to identify; either this or login/email + passsword must be specified to init connection
`api_version`	Either "v3" or "v4" for one of the two gitlab API version. See Details section on API versions.
`api_location`	location of the gitlab API under the `gitlab_url`, usually and by default "/api/$api_version/"
`project`	id or name of project to issue requests to

Details

The returned function should serve as the primary way to access the gitlab API in the following. It can take vector/character arguments in the same way as the function `gitlab` does, as well as the convenience functions provided by this package or written by the user. If it is passed such that function it calls it with the arguments provided in `...` and the gitlab URL, api location and private_token provided when creating it via `gl_connection`.

Note: currently gitlab API v3 is supported. Support for Gitlab API v4 (for Gitlab version >= 9.0) will be added soon.

Value

A function to access a specific gitlab API as a specific user, see details

API versions

Currently (April 2017, Gitlab version 9.0), Gitlab provides two API versions "v3" and "v4", where "v3" is deprecated and to be removed soon from Gitlab. "v4" is the standard API since Gitlab version 9.0. gitlabr supports both API versions, since "v3" was the standard until very recently. gitlabr will support API v3 until gitlabr 1.0 (to be released in 2017), with which it will become deprecated also in gitlabr. "v4" is the default setting in gitlabr from version 0.9 on.

For some functions, where the API endpoints differ in logic, a parameter 'force_api_v3' is provided with functions to enforce API v3 logic. This has to be set manually with each call in addition to the api_version parameter of the connection. Rather than using this parameter, it is intended to update your Gitlab installation to support API v4. Use this parameter only as a workaround when this is not possible!

Examples

```
## Not run:
my_gitlab <- gl_connection("http://gitlab.example.com", "123####89")
my_gitlab("projects")
my_gitlab(gl_get_file, "test-project", "README.md", ref = "dev")

## End(Not run)
```

---

gl_create_merge_request

Create a merge request

Description

Create a merge request

Usage

```
gl_create_merge_request(project, source_branch, target_branch = "master",
  title, description, verb = httr::POST, ...)
```

Arguments

project	name or id of project (not repository!)
source_branch	name of branch to be merged
target_branch	name of branch into which to merge
title	title of the merge request
description	description text for the merge request
verb	is ignored, will always be forced to match the action the function name indicates
...	passed on to gitlab. Might contain more fields documented in gitlab API doc.

gl_get_comments *Get the comments/notes of a commit or issue*

Description

Get the comments/notes of a commit or issue

Usage

```
gl_get_comments(object_type = "issue", id, note_id = c(), project, ...)

gl_get_issue_comments(...)

gl_get_commit_comments(...)

gl_comment_commit(project, id, text, ...)

gl_comment_issue(project, id, text, ...)

gl_edit_comment(object_type, text, ...)

gl_edit_issue_comment(...)

gl_edit_commit_comment(...)
```

Arguments

object_type	one of "issue" or "commit". Snippets and merge_requests are not implemented yet.
id	id of object: sha for commits, not issues notes/comments: (project-wide) id for api version 4, (global) iid for api version 3
note_id	id of note
project	project name or id
...	passed on to gitlab API call. See Details.
text	Text of comment/note to add or edit (translates to gitlab API note/body respectively)

Details

For gl_comment_commit ... might also contain path, line and line_type (old or new) to attach the comment to a specific in a file. See http://doc.gitlab.com/ce/api/commits.html

Examples

```
## Not run:
my_project <- gl_project_connection(project = "testor"...) ## fill in login parameters
my_project(gl_get_comments, "issue", 1)
my_project(gl_get_comments, "commit", "8ce5ef240123cd78c1537991e5de8d8323666b15")
my_project(gl_comment_issue, 1, text = "Almost done!")

## End(Not run)
```

---

gl_get_commits *Get commits and diff from a project repository*

Description

Get commits and diff from a project repository

Usage

```
gl_get_commits(project, commit_sha = c(), ...)

gl_get_diff(project, commit_sha, ...)
```

Arguments

project	project name or id
commit_sha	if not null, get only the commit with the specific hash; for gl_get_diff this must be specified
...	passed on to gitlab API call, may contain ref_name for specifying a branch or tag to list commits of

---

gl_get_project_id *Get a project id by name*

Description

Get a project id by name

Usage

```
gl_get_project_id(project_name, verb = httr::GET, auto_format = TRUE, ...)
```

Arguments

project_name	project name
verb	ignored; all calls with this function will have gitlab's default verb httr::GET
auto_format	ignored
...	passed on to gitlab

gl_list_branches *List, create and delete branches*

Description

List, create and delete branches

List, create and delete branches

List, create and delete branches

Usage

```
gl_list_branches(project, verb = httr::GET, ...)

gl_create_branch(project, branch_name, ref = "master", verb = httr::POST,
  ...)

gl_delete_branch(project, branch_name, verb = httr::POST, ...)
```

Arguments

project	name or id of project (not repository!)
verb	is ignored, will always be forced to match the action the function name indicates
...	passed on to gitlab
branch_name	name of branch to create/delete
ref	ref name of origin for newly created branch

gl_list_issues *Get issues of a project or user*

Description

Get issues of a project or user

Usage

```
gl_list_issues(project = NULL, issue_id = NULL, verb = httr::GET,
  force_api_v3 = FALSE, ...)

gl_get_issue(issue_id, project, ...)
```

Arguments

project	project name or id, may be null for all issues created by user
issue_id	optional issue id (projectwide; for API v3 only you can use global iid when force_api_v3 is 'TRUE')
verb	ignored; all calls with this function will have gitlab's default verb httr::GET
force_api_v3	a switch to force deprecated gitlab API v3 behavior that allows filtering by global iid. If 'TRUE' filtering happens by global iid, if false, it happens by projectwide ID. For API v4, this must be FALSE (default)
...	further parameters passed on to gitlab, may be state, labels, issue id, ...

Details

gl_get_issue provides a wrapper with swapped arguments for convenience, esp. when using a project connection

Examples

```
## Not run:
my_project <- gl_project_connection(project = "testor"...) ## fill in login parameters
my_project(gl_list_issues)
my_project(gl_get_issue, 1)
my_project(gl_new_issue, 1, "Implement new feature", description = "It should be awesome.")

## End(Not run)
```

---

gl_list_projects *List projects in Gitlab*

Description

List projects in Gitlab

Usage

```
gl_list_projects(...)
```

Arguments

...	passed on to gitlab

Examples

```
## Not run:
my_gitlab <- gl_connection(...) ## fill in login parameters
my_gitlab(gl_list_projects)

## End(Not run)
```

gl_new_issue *Post a new issue or edit one*

Description

Post a new issue or edit one

Post a new issue or edit one

Usage

```
gl_new_issue(title, project, ...)

gl_edit_issue(issue_id, project, force_api_v3 = FALSE, ...)

gl_close_issue(issue_id, project, ...)

gl_reopen_issue(issue_id, project, ...)

gl_assign_issue(issue_id, assignee_id = NULL, project, ...)

gl_unassign_issue(issue_id, project, ...)
```

Arguments

title	title of the issue
project	project where the issue should be posted
...	further parameters passed to the API call, may contain description, asignee_id, milestone_id, labels, state_event (for edit_issue).
issue_id	issue id (projectwide; for API v3 only you can use global iid when force_api_v3 is 'TRUE' although this is not recommended!)
force_api_v3	a switch to force deprecated gitlab API v3 behavior that allows filtering by global iid. If 'TRUE' filtering happens by global iid, if false, it happens by projectwide ID. For API v4, this must be FALSE (default)
assignee_id	numeric id of users as returned in '/users/' API request

gl_pipelines *Access the Gitlab CI builds*

Description

List the jobs with `gl_jobs`, the pipelines with `gl_pipelines` or download the most recent artifacts archive with `gl_latest_build_artifact`. For every branch and job combination only the most recent artifacts archive is available. `gl_builds` is the equivalent for gitlab API v3.

Usage

```
gl_pipelines(project, ...)

gl_jobs(project, ...)

gl_builds(project, force_api_v3 = TRUE, ...)

gl_latest_build_artifact(project, job, ref_name = "master",
  save_to_file = tempfile(fileext = ".zip"), ...)
```

Arguments

project	project name or id, required
...	passed on to gitlab API call
force_api_v3	Since gl_builds is no longer working for Gitlab API v4, this must be set to TRUE in order to avoid deprecation warning and HTTP error. It currently default to TRUE, but this will change with gitlabr 1.0.
job	Name of the job to get build artifacts from
ref_name	name of ref (i.e. branch, commit, tag)
save_to_file	either a path where to store .zip file or NULL if raw should be returned

Value

returns the file path if save_to_file is TRUE, or the archive as raw otherwise.

Examples

```
## Not run:
my_gitlab <- gl_connection(...) ## fill in login parameters
my_gitlab(gl_pipelines, "test-project")
my_gitlab(gl_jobs, "test-project")
my_gitlab(gl_latest_build_artifact, "test-project", job = "build")

## End(Not run)
```

---

gl_proj_req *Create a project specific request*

Description

Prefixes the request location with "project/:id" and automatically translates project names into ids

Usage

```
gl_proj_req(project, req, ...)
```

Arguments

project	project name or id
req	character vector of request location
...	passed on to gl_get_project_id

---

gl_push_file *Upload a file to a gitlab repository*

Description

If the file already exists, it is updated/overwritten by default

Usage

```
gl_push_file(project, file_path, content, commit_message,
  branch_name = "master", overwrite = TRUE, ...)
```

Arguments

project	Project name or id
file_path	path where to store file in gl_repository
content	file content (text)
commit_message	Message to use for commit with new/updated file
branch_name	name of branch where to append newly generated commit with new/updated file
overwrite	whether to overwrite files that already exist
...	passed on to gitlab

Value

returns a data_frame with changed branch and path (0 rows if nothing was changed, since overwrite is FALSE)

Examples

```
## Not run:
my_project <- gl_project_connection(project = "example-project", ...) ## fill in login parameters
my_project(gl_push_file, "data/test_data.csv",
           content = readLines("test-data.csv"),
           commit_message = "New test data")

## End(Not run)
```

gl_repository *Access to repository functions and files in Gitlab API*

Description

Access to repository functions and files in Gitlab API

For `gl_file_exists` dots are passed on to `gl_list_files` and gitlab API call

Get a file from a gitlab repository

Usage

```
gl_repository(req = c("tree"), project, ...)

gl_list_files(...)

gl_file_exists(project, file_path, ...)

gl_get_file(project, file_path, ref = "master", to_char = TRUE,
  force_api_v3 = FALSE, ...)
```

Arguments

`req`	request to perform on repository (everything after '/repository/' in gitlab API, as vector or part of URL)
`project`	name or id of project (not repository!)
`...`	passed on to `gitlab` API call
`file_path`	path to file
`ref`	name of ref (commit branch or tag)
`to_char`	flag if output should be converted to char; otherwise it is of class raw
`force_api_v3`	a switch to force deprecated gitlab API v3 behavior. See details section "API version" of `gl_connection`

Examples

```
## Not run:
my_project <- gl_project_connection(project = "example-project", ...) ## fill in login parameters
my_project(gl_list_files)
my_project(gl_get_file, "data.csv")

## End(Not run)
```

gl_to_issue_id *Translate projectwide issue id to global gitlab API issue id*

Description

This functions is only intended to be used with gitlab API v3. With v4, the global iid is no longer functional.

Usage

```
gl_to_issue_id(issue_id, project, force_api_v3 = TRUE, ...)
```

Arguments

issue_id	projectwide issue id (as seen by e.g. gitlab website users)
project	project name or id
force_api_v3	Since this function is no longer necessary for Gitlab API v4, this must be set to TRUE in order to avoid deprecation warning and HTTP error. It currently default to TRUE, but this will change with gitlabr 1.0.
...	passed on to gitlab

iff *Apply a function if and only if test is TRUE*

Description

otherwise return input value unchanged

Usage

```
iff(obj, test, fun, ...)

iffn(obj, test, fun, ...)
```

Arguments

obj	object to apply test and fun to
test	logical or function to apply to test
fun	function to apply
...	passed on to test

Details

iffn is ... if and only if test is FALSE

pipe_into *Pipe into specific formal argument*

Description

This rotates the order of the arguments such that the one named in param_name comes first and then calls the function.

Usage

```
pipe_into(x, param_name, fun, ...)
```

Arguments

x	value to be piped into fun
param_name	name of the argument that x should be assigned to
fun	function
...	further arguments for fun

set_gitlab_connection *Get/set a gitlab connection for all calls*

Description

This sets the default value of gitlab_con in a call to gitlab

Usage

```
set_gitlab_connection(gitlab_con = NULL, ...)

unset_gitlab_connection()
```

Arguments

gitlab_con	A function used for gitlab API calls, such as gitlab or as returned by gl_connection.
...	if gitlab_con is NULL, a new connection is created used the parameters is ... using gl_connection

Examples

```
## Not run:
set_gitlab_connection("http://gitlab.example.com", private_token = "123####89")

## End(Not run)
```

`update_gitlabr_code` *Convert gitlabr legacy code to 0.7 compatible*

Description

CAUTION: This functions output/results should be checked manually before committing the code, since it uses regular expression heuristically to parse code and cannot guarantee complete and correct code replacement

Usage

```
update_gitlabr_code(file, text = readLines(file), internal = FALSE)
```

Arguments

`file`	file to read from/write to. Maybe NULL for input and return only
`text`	lines of code to convert
`internal`	whether to also replace names of internal functions

Index