Package 'gert'

October 29, 2019

Type Package

Title Simple Git Client for R

Version 0.3

Author Jeroen Ooms

Maintainer Jeroen Ooms <jeroen@berkeley.edu>

Description Simple git client based on 'libgit2' with user-friendly authentication and support for both SSH and HTTPS remotes on all platforms. User credentials are shared with command line 'git' through the git-credential store and ssh keys stored on disk or ssh-agent. On Linux, a somewhat recent version of 'libgit2' is required.

License MIT + file LICENSE

SystemRequirements libgit2 (>= 0.26): libgit2-devel (rpm) or libgit2-dev (deb)

URL https://jeroen.cran.dev/gert (website), httsp://github.com/r-lib/gert (devel), https://libgit2.org (upstream)

BugReports http://github.com/r-lib/gert/issues

Encoding UTF-8

RoxygenNote 6.1.1

Imports openssl (>= 1.4.1), credentials (>= 1.0), askpass

Suggests testthat

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-10-29 13:00:02 UTC

R topics documented:

`branch` *Git Branch*

Description

Create, list, and checkout branches.

Usage

```
git_branch_list(repo = ".")

git_branch_checkout(branch, force = FALSE, repo = ".")

git_branch_create(name, ref = "HEAD", checkout = TRUE, repo = ".")

git_branch_delete(name, repo = ".")

git_branch_fast_forward(ref, repo = ".")

git_branch_set_upstream(remote = "origin", repo = ".")
```

Arguments

`repo`	a path to an existing repository, or a `git_repository` object as returned by git_open, git_init or git_clone.
`branch`	name of branch to check out
`force`	ignore conflicts and overwrite modified files
`name`	string with name of the branch / tag / etc
`ref`	string with a branch/tag/commit
`checkout`	move HEAD to the newly created branch
`remote`	name of existing remote from git_remote_list

See Also

Other git: `commit`, `fetch`, `git_config`, `repository`, `signature`

commit *Stage and commit changes*

Description

To commit changes, start with *staging* the files to be included in the commit using `git_add()` or `git_rm()`. Use `git_status()` to see an overview of staged and unstaged changes, and finally `git_commit()` creates a new commit with currently staged files.

git_commit_all is a shorthand that will automatically stage all new and modified files and then commit.

Also `git_log()` shows the most recent commits and `git_ls()` lists all the files that are being tracked in the repository.

Usage

```
git_commit(message, author = NULL, committer = NULL, repo = ".")

git_commit_all(message, author = NULL, committer = NULL, repo = ".")

git_add(files, force = FALSE, repo = ".")

git_rm(files, repo = ".")

git_status(repo = ".")

git_ls(repo = ".")

git_log(ref = "HEAD", max = 100, repo = ".")

git_reset(type = c("soft", "hard", "mixed"), ref = "HEAD",
  repo = ".")
```

Arguments

`message`	a commit message
`author`	A git_signature value, default is git_signature_default.
`committer`	A git_signature value, default is same as `author`
`repo`	a path to an existing repository, or a `git_repository` object as returned by git_open, git_init or git_clone.
`files`	vector of paths relative to the git root directory. Use `"."` to stage all changed files.
`force`	add files even if in gitignore
`ref`	string with a branch/tag/commit
`max`	lookup at most latest n parent commits
`type`	must be one of `"soft"`, `"hard"`, or `"mixed"`

See Also

Other git: branch, fetch, git_config, repository, signature

---

fetch *Push and Pull*

Description

Use git_fetch and git_push to sync a local branch with a remote. Here git_pull is a wrapper for git_fetch which that tries to fast-forward the local branch after fetching.

Usage

```
git_fetch(remote = NULL, refspec = NULL, password = askpass,
  ssh_key = NULL, verbose = interactive(), repo = ".")

git_push(remote = NULL, refspec = NULL, password = askpass,
  ssh_key = NULL, mirror = FALSE, force = FALSE,
  verbose = interactive(), repo = ".")

git_clone(url, path = NULL, branch = NULL, password = askpass,
  ssh_key = NULL, bare = FALSE, mirror = FALSE,
  verbose = interactive())

git_pull(..., repo = ".")
```

Arguments

remote	name of a remote listed in git_remote_list()
refspec	string with mapping between remote and local refs
password	a string or a callback function to get passwords for authentication or password protected ssh keys. Defaults to askpass which checks getOption('askpass').
ssh_key	path or object containing your ssh private key. By default we look for keys in ssh-agent and credentials::ssh_key_info.
verbose	display some progress info while downloading
repo	a path to an existing repository, or a git_repository object as returned by git_open, git_init or git_clone.
mirror	use the --mirror flag
force	use the --force flag
url	remote url. Typically starts with https://github.com/ for public repositories, and https://yourname@github.com/ or git@github.com/ for private repos. You will be prompted for a password or pat when needed.
path	directory of the git repository. For git_init or git_clone this must be a non-existing or empty directory.

branch	name of branch to check out locally
bare	use the --bare flag
...	arguments passed to git_fetch

See Also

Other git: branch, commit, git_config, repository, signature

Examples

```
{# Clone a small repository
git_dir <- file.path(tempdir(), 'antiword')
git_clone('https://github.com/ropensci/antiword', git_dir)

# Change into the repo directory
olddir <- getwd()
setwd(git_dir)

# Show some stuff
git_log()
git_branch_list()
git_remote_list()

# Add a file
write.csv(iris, 'iris.csv')
git_add('iris.csv')

# Commit the change
jerry <- git_signature("Jerry", "jerry@hotmail.com")
git_commit('added the iris file', author = jerry)

# Now in the log:
git_log()

# Cleanup
setwd(olddir)
unlink(git_dir, recursive = TRUE)
}
```

---

git_checkout_pull_request

GitHub Wrappers

Description

Some wrappers for working with Github.

Usage

```
git_checkout_pull_request(pr = 1, remote = "origin", repo = ".")
```

Arguments

pr	number with PR to check out
remote	name of a remote listed in git_remote_list()
repo	a path to an existing repository, or a git_repository object as returned by git_open, git_init or git_clone.

---

git_config *Version info*

Description

Shows the version of libgit2 and which features have been enabled.

Usage

```
libgit2_config()

git_config(repo = ".")

git_config_global()

git_config_set(name, value, repo = ".")

git_config_global_set(name, value)
```

Arguments

repo	a path to an existing repository, or a git_repository object as returned by git_open, git_init or git_clone.
name	setting name
value	setting value, must be string, bool, number or NULL

See Also

Other git: branch, commit, fetch, repository, signature

remotes *Git Remotes*

Description

Add, remove and list remotes.

Usage

```
git_remote_list(repo = ".")

git_remote_add(name, url, refspec = NULL, repo = ".")

git_remote_remove(name, repo = ".")

git_refspecs(repo = ".")
```

Arguments

repo	a path to an existing repository, or a `git_repository` object as returned by git_open, git_init or git_clone.
name	unique name of the remote
url	server url (https or ssh)
refspec	optional string with the remote fetch value

repository *Create or open a git repository*

Description

Use `git_init()` to start a new repository or `git_clone()` to download a repository from a remote.

Usage

```
git_init(path = ".")

git_open(path = ".")

git_find(path = ".")

git_info(repo = ".")
```

Arguments

path	directory of the git repository. For git_init or git_clone this must be a non-existing or empty directory.
repo	a path to an existing repository, or a git_repository object as returned by git_open, git_init or git_clone.

Details

You may use git_find() and git_open() to explicitly discover and open existing git repositories, but this is usually not needed because all gert functions also take a path argument which implicitly opens the repo.

See Also

Other git: branch, commit, fetch, git_config, signature

---

signature *Author Signature*

Description

A signature contains the author and timestamp of a commit. This is needed by the git_commit function.

Usage

```
git_signature_default(repo = ".")

git_signature(name, email, time = NULL)
```

Arguments

repo	a path to an existing repository, or a git_repository object as returned by git_open, git_init or git_clone.
name	Real name of the committer
email	Email address of the commmitter
time	timestamp of class POSIXt or NULL

See Also

Other git: branch, commit, fetch, git_config, repository

stash *Stashing changes*

Description

Temporary stash away changed from the working directory.

Usage

```
git_stash_save(message = "", keep_index = FALSE,
  include_untracked = FALSE, include_ignored = FALSE, repo = ".")

git_stash_pop(index = 0, repo = ".")

git_stash_drop(index = 0, repo = ".")

git_stash_list(repo = ".")
```

Arguments

message	optional message to store the stash
keep_index	changes already added to the index are left intact in the working directory
include_untracked	untracked files are also stashed and then cleaned up from the working directory
include_ignored	ignored files are also stashed and then cleaned up from the working directory
repo	a path to an existing repository, or a git_repository object as returned by git_open, git_init or git_clone.
index	The position within the stash list. 0 points to the most recent stashed state.

tag *Git Tag*

Description

Create and list tags.

Usage

```
git_tag_list(match = "*", repo = ".")

git_tag_create(name, message, ref = "HEAD", repo = ".")

git_tag_delete(name, repo = ".")

git_tag_push(name, ..., repo = ".")
```

Arguments

match	pattern to filter tags (use * for wildcard)
repo	a path to an existing repository, or a git_repository object as returned by git_open, git_init or git_clone.
name	tag name
message	tag message
ref	target reference to tag
...	other arguments passed to git_push

Index