Package: githubr (via r-universe)

September 18, 2024

Title Easier to Use API Wrapper for 'GitHub'

Version 0.9.1

Description This is a 'GitHub' API wrapper for R.
<https://docs.github.com/en/rest> It uses the 'gh' package but
has things wrapped up for convenient use cases.

License GPL-3

URL https://github.com/fhdsl/githubr

BugReports https://github.com/fhdsl/githubr/issues

Imports gitcreds, dplyr, gh, magrittr, httr,

Encoding UTF-8

RoxygenNote 7.2.3

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Repository https://fhdsl.r-universe.dev

RemoteUrl https://github.com/fhdsl/githubr

RemoteRef HEAD

RemoteSha 3638aa22971444fb7b5b9df508ed1ba196c17d1e

Contents

accept_all_invites 2
check_git_repo 2
get_git_auth 3
get_issues 4
get_repos 5

Index **7**

accept_all_invites *Accept an invite*

Description

Given an a user name, accept all invites

Usage

```
accept_all_invites(git_pat = NULL, verbose = TRUE)
```

Arguments

git_pat	Whatever credentials are given are where invites will be accepted from. If none is supplied, then this will attempt to grab from a git pat set in the environment with usethis::create_github_token().
verbose	TRUE/FALSE do you want more progress messages?

Value

a response from GitHub's API

Examples

```
## Not run:

# First, set up your GitHub credentials using `usethis::gitcreds_set()`.
# Get a GitHub personal access token (PAT)
usethis::create_github_token()

# Give this token to `gitcreds_set()`
gitcreds::gitcreds_set()

# All invites that have been sent to the PAT you have provided you will be accepted
accept_all_invites()

## End(Not run)
```

check_git_repo *Check if a repository exists on GitHub*

Description

Given a repository name, check with git ls-remote whether the repository exists and return a TRUE/FALSE

Usage

```
check_git_repo(
  repo_name,
  git_pat = NULL,
  silent = TRUE,
  return_repo = FALSE,
  verbose = TRUE
)
```

Arguments

repo_name	the name of the repository, e.g. jhudsl/OTTR_Template
git_pat	A personal access token from GitHub. Only necessary if the repository being checked is a private repository.
silent	TRUE/FALSE of whether the warning from the git ls-remote command should be echoed back if it does fail.
return_repo	TRUE/FALSE of whether or not the output from git ls-remote should be saved to a file (if the repo exists)
verbose	TRUE/FALSE do you want more progress messages?

Value

A TRUE/FALSE whether or not the repository exists. Optionally the output from git ls-remote if return_repo = TRUE.

Examples

```
 ## Not run:

exists <- check_git_repo("jhudsl/OTTR_Template")

if (exists) message("Yup, this repo exists")

## End(Not run)
```

---

get_git_auth *Handle GitHub PAT authorization*

Description

Handle things whether or not a GitHub PAT is supplied.

Usage

```
get_git_auth(
  git_pat = NULL,
  git_username = "PersonalAccessToken",
  quiet = FALSE
)
```

Arguments

git_pat	If private repositories are to be retrieved, a github personal access token needs to be supplied. If none is supplied, then this will attempt to grab from a git pat set in the environment with usethis::create_github_token().
git_username	Optional, can include username for credentials.
quiet	Use TRUE if you don't want the warning about no GitHub credentials.

Value

Authorization argument to supply to curl OR a blank string if no authorization is found or supplied.

---

get_issues *Retrieve all issues on a repository*

Description

Given a repository name, get a list of issues.

Usage

```
get_issues(repo_name, how_many = "all", git_pat = NULL, verbose = TRUE)
```

Arguments

repo_name	the name of the repository to retrieve issues from, e.g. jhudsl/OTTR_Template
how_many	put the number of how many you would like returned. If all, put "all". By default will return all issues.
git_pat	A personal access token from GitHub. Only necessary if the repository being checked is a private repository.
verbose	TRUE/FALSE do you want more progress messages?

Value

A data frame that contains information about the issues from the given repository

Examples

```
 ## Not run:

# First, set up your GitHub credentials using `usethis::gitcreds_set()`.
# Get a GitHub personal access token (PAT)
usethis::create_github_token()

# Give this token to `gitcreds_set()`
gitcreds::gitcreds_set()

# Now you can retrieve issues
issues_df <- get_issues("jhudsl/OTTR_Template")
# Alternatively, you can supply the GitHub PAT directly
# to the function to avoid doing the steps above.
issues_df <- get_issues("jhudsl/OTTR_Template", git_pat = "gh_somepersonalaccesstokenhere")

## End(Not run)
```

---

get_repos *Retrieve all repos for an organization or user*

Description

Given a username or organization, retrieve all the repos

Usage

```
get_repos(owner, how_many = "all", git_pat = NULL, verbose = TRUE)
```

Arguments

owner	the name of the organization or user to retrieve a list of repos from. E.g. fhdsl
how_many	put the number of how many you would like returned. If all, put "all". By default will return all issues.
git_pat	A personal access token from GitHub. Only necessary if the repository being checked is a private repository.
verbose	TRUE/FALSE do you want more progress messages?

Value

A data frame that contains information about the issues from the given repository

Examples

```
 ## Not run:

# First, set up your GitHub credentials using `usethis::gitcreds_set()`.
# Get a GitHub personal access token (PAT)
usethis::create_github_token()

# Give this token to `gitcreds_set()`
gitcreds::gitcreds_set()

# Now you can retrieve the repositories
repos_df <- get_repos("fhdsl")

# Alternatively, you can supply the GitHub PAT directly
# to the function to avoid doing the steps above.
repos_df <- get_repos("fhdsl", git_pat = "gh_somepersonalaccesstokenhere")

## End(Not run)
```

Index

accept_all_invites, 2

check_git_repo, 2

get_git_auth, 3
get_issues, 4
get_repos, 5