Package 'gh'

November 27, 2020

Title 'GitHub' 'API'

Version 1.2.0

Description Minimal client to access the 'GitHub' 'API'.

License MIT + file LICENSE

URL https://gh.r-lib.org/, https://github.com/r-lib/gh#readme

BugReports https://github.com/r-lib/gh/issues

Imports cli (>= 2.0.1), gitcreds, httr (>= 1.2), ini, jsonlite

Suggests covr, knitr, rmarkdown, rprojroot, spelling, testthat (>= 2.3.2), withr

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.1.9000

NeedsCompilation no

Author Gábor Csárdi [cre, ctb],
Jennifer Bryan [aut],
Hadley Wickham [aut],
RStudio [cph, fnd]

Maintainer Gábor Csárdi <csardi.gabor@gmail.com>

Repository CRAN

Date/Publication 2020-11-27 13:50:02 UTC

R topics documented:

---

gh *Query the GitHub API*

Description

This is an extremely minimal client. You need to know the API to be able to use this client. All this function does is:

- Try to substitute each listed parameter into `endpoint`, using the `{parameter}` notation.
- If a GET request (the default), then add all other listed parameters as query parameters.
- If not a GET request, then send the other parameters in the request body, as JSON.
- Convert the response to an R list using `jsonlite::fromJSON()`.

Usage

```
gh(
  endpoint,
  ...,
  per_page = NULL,
  .token = NULL,
  .destfile = NULL,
  .overwrite = FALSE,
  .api_url = NULL,
  .method = "GET",
  .limit = NULL,
  .accept = "application/vnd.github.v3+json",
  .send_headers = NULL,
  .progress = TRUE
)
```

Arguments

`endpoint` GitHub API endpoint. Must be one of the following forms:

- METHOD path, e.g. `GET /rate_limit`,
- `path`, e.g. /rate_limit,
- METHOD url, e.g. GET https://api.github.com/rate_limit,
- `url`, e.g. https://api.github.com/rate_limit.

If the method is not supplied, will use `.method`, which defaults to `"GET"`.

`...`	Name-value pairs giving API parameters. Will be matched into `endpoint` placeholders, sent as query parameters in GET requests, and as a JSON body of POST requests. If there is only one unnamed parameter, and it is a raw vector, then it will not be JSON encoded, but sent as raw data, as is. This can be used for example to add assets to releases. Named `NULL` values are silently dropped. For GET requests, named `NA` values trigger an error. For other methods, named `NA` values are included in the body of the request, as JSON `null`.
`per_page`	Number of items to return per page. If omitted, will be substituted by `max(.limit,100)` if `.limit` is set, otherwise determined by the API (never greater than 100).
`.token`	Authentication token. Defaults to `GITHUB_PAT` or `GITHUB_TOKEN` environment variables, in this order if any is set. See `gh_token()` if you need more flexibility, e.g. different tokens for different GitHub Enterprise deployments.
`.destfile`	Path to write response to disk. If `NULL` (default), response will be processed and returned as an object. If path is given, response will be written to disk in the form sent.
`.overwrite`	If `.destfile` is provided, whether to overwrite an existing file. Defaults to `FALSE`.
`.api_url`	Github API url (default: `https://api.github.com`). Used if `endpoint` just contains a path. Defaults to `GITHUB_API_URL` environment variable if set.
`.method`	HTTP method to use if not explicitly supplied in the `endpoint`.
`.limit`	Number of records to return. This can be used instead of manual pagination. By default it is `NULL`, which means that the defaults of the GitHub API are used. You can set it to a number to request more (or less) records, and also to `Inf` to request all records. Note, that if you request many records, then multiple GitHub API calls are used to get them, and this can take a potentially long time.
`.accept`	The value of the `Accept` HTTP header. Defaults to `"application/vnd.github.v3+json"`. If `Accept` is given in `.send_headers`, then that will be used. This parameter can be used to provide a custom media type, in order to access a preview feature of the API.
`.send_headers`	Named character vector of header field values (except `Authorization`, which is handled via `.token`). This can be used to override or augment the default `User-Agent` header: `"https://github.com/r-lib/gh"`.
`.progress`	Whether to show a progress indicator for calls that need more than one HTTP request.

Value

Answer from the API as a `gh_response` object, which is also a `list`. Failed requests will generate an R error. Requests that generate a raw response will return a raw vector.

See Also

`gh_gql()` if you want to use the GitHub GraphQL API, `gh_whoami()` for details on GitHub API token management.

Examples

```
## Repositories of a user, these are equivalent
gh("/users/hadley/repos")
gh("/users/{username}/repos", username = "hadley")

## Starred repositories of a user
gh("/users/hadley/starred")
gh("/users/{username}/starred", username = "hadley")


## Create a repository, needs a token in GITHUB_PAT (or GITHUB_TOKEN)
## environment variable
gh("POST /user/repos", name = "foobar")


## Issues of a repository
gh("/repos/hadley/dplyr/issues")
gh("/repos/{owner}/{repo}/issues", owner = "hadley", repo = "dplyr")

## Automatic pagination
users <- gh("/users", .limit = 50)
length(users)


## Access developer preview of Licenses API (in preview as of 2015-09-24)
gh("/licenses") # used to error code 415
gh("/licenses", .accept = "application/vnd.github.drax-preview+json")


## Access Github Enterprise API
## Use GITHUB_API_URL environment variable to change the default.
gh("/user/repos", type = "public", .api_url = "https://github.foobar.edu/api/v3")


## Use I() to force body part to be sent as an array, even if length 1
## This works whether assignees has length 1 or > 1
assignees <- "gh_user"
assignees <- c("gh_user1", "gh_user2")
gh("PATCH /repos/OWNER/REPO/issues/1", assignees = I(assignees))


## There are two ways to send JSON data. One is that you supply one or
## more objects that will be converted to JSON automatically via
## jsonlite::toJSON(). In this case sometimes you need to use
## jsonlite::unbox() because fromJSON() creates lists from scalar vectors
## by default. The Content-Type header is automatically added in this
## case. For example this request turns on GitHub Pages, using this
## API: https://docs.github.com/v3/repos/pages/#enable-a-pages-site

gh::gh(
  "POST /repos/{owner}/{repo}/pages",
```

```
  owner = "gaborcsardi",
  repo = "playground",
  source = list(
    branch = jsonlite::unbox("master"),
    path = jsonlite::unbox("/docs")
  ),
  .send_headers = c(Accept = "application/vnd.github.switcheroo-preview+json")
)

## The second way is to handle the JSON encoding manually, and supply it
## as a raw vector in an unnamed argument, and also a Content-Type header:

body <- '{ "source": { "branch": "master", "path": "/docs" } }'
gh::gh(
  "POST /repos/{owner}/{repo}/pages",
  owner = "gaborcsardi",
  repo = "playground",
  charToRaw(body),
  .send_headers = c(
    Accept = "application/vnd.github.switcheroo-preview+json",
    "Content-Type" = "application/json"
  )
)
```

`gh_gql` *A simple interface for the GitHub GraphQL API v4.*

Description

See more about the GraphQL API here: https://docs.github.com/graphql

Usage

```
gh_gql(query, ...)
```

Arguments

`query`	The GraphQL query, as a string.
`...`	Name-value pairs giving API parameters. Will be matched into `endpoint` placeholders, sent as query parameters in GET requests, and as a JSON body of POST requests. If there is only one unnamed parameter, and it is a raw vector, then it will not be JSON encoded, but sent as raw data, as is. This can be used for example to add assets to releases. Named `NULL` values are silently dropped. For GET requests, named `NA` values trigger an error. For other methods, named `NA` values are included in the body of the request, as JSON `null`.

Details

Note: pagination and the `.limit` argument does not work currently, as pagination in the GraphQL API is different from the v3 API. If you need pagination with GraphQL, you'll need to do that manually.

See Also

`gh()` for the GitHub v3 API.

Examples

```
gh_gql("query { viewer { login }}")
```

`gh_next` *Get the next, previous, first or last page of results*

Description

Get the next, previous, first or last page of results

Usage

```
gh_next(gh_response)

gh_prev(gh_response)

gh_first(gh_response)

gh_last(gh_response)
```

Arguments

`gh_response`	An object returned by a `gh()` call.

Details

Note that these are not always defined. E.g. if the first page was queried (the default), then there are no first and previous pages defined. If there is no next page, then there is no next page defined, etc.

If the requested page does not exist, an error is thrown.

Value

Answer from the API.

See Also

The `.limit` argument to `gh()` supports fetching more than one page.

Examples

```
x <- gh("/users")
vapply(x, "[[", character(1), "login")
x2 <- gh_next(x)
vapply(x2, "[[", character(1), "login")
```

`gh_rate_limit` *Return GitHub user's current rate limits*

Description

Reports the current rate limit status for the authenticated user, either pulls this information from a previous successful request or directly from the GitHub API.

Usage

```
gh_rate_limit(
  response = NULL,
  .token = NULL,
  .api_url = NULL,
  .send_headers = NULL
)
```

Arguments

`response`	`gh_response` object from a previous gh call, rate limit values are determined from values in the response header. Optional argument, if missing a call to "GET /rate_limit" will be made.
`.token`	Authentication token. Defaults to `GITHUB_PAT` or `GITHUB_TOKEN` environment variables, in this order if any is set. See `gh_token()` if you need more flexibility, e.g. different tokens for different GitHub Enterprise deployments.
`.api_url`	Github API url (default: https://api.github.com). Used if `endpoint` just contains a path. Defaults to `GITHUB_API_URL` environment variable if set.
`.send_headers`	Named character vector of header field values (except `Authorization`, which is handled via `.token`). This can be used to override or augment the default `User-Agent` header: `"https://github.com/r-lib/gh"`.

Details

Further details on GitHub's API rate limit policies are available at https://docs.github.com/v3/#rate-limiting.

Value

A `list` object containing the overall `limit`, `remaining` limit, and the limit `reset` time.

`gh_token` *Return the local user's GitHub Personal Access Token (PAT)*

Description

If gh can find a personal access token (PAT) via `gh_token()`, it includes the PAT in its requests. Some requests succeed without a PAT, but many require a PAT to prove the request is authorized by a specific GitHub user. A PAT also helps with rate limiting. If your gh use is more than casual, you want a PAT.

gh calls `gitcreds::gitcreds_get()` with the `api_url`, which checks session environment variables and then the local Git credential store for a PAT appropriate to the `api_url`. Therefore, if you have previously used a PAT with, e.g., command line Git, gh may retrieve and re-use it. You can call `gitcreds::gitcreds_get()` directly, yourself, if you want to see what is found for a specific URL. If no matching PAT is found, `gitcreds::gitcreds_get()` errors, whereas `gh_token()` does not and, instead, returns `""`.

See GitHub's documentation on Creating a personal access token, or use `usethis::create_github_token()` for a guided experience, including pre-selection of recommended scopes. Once you have a PAT, you can use `gitcreds::gitcreds_set()` to add it to the Git credential store. From that point on, gh (via `gitcreds::gitcreds_get()`) should be able to find it without further effort on your part.

Usage

```
gh_token(api_url = NULL)
```

Arguments

`api_url`	GitHub API URL. Defaults to the `GITHUB_API_URL` environment variable, if set, and otherwise to `https://api.github.com`.

Value

A string of 40 hexadecimal digits, if a PAT is found, or the empty string, otherwise. For convenience, the return value has an S3 class in order to ensure that simple printing strategies don't reveal the entire PAT.

Examples

```
## Not run:
gh_token()

format(gh_token())

str(gh_token())

## End(Not run)
```

gh_tree_remote *Find the GitHub remote associated with a path*

Description

This is handy helper if you want to make gh requests related to the current project.

Usage

```
gh_tree_remote(path = ".")
```

Arguments

path	Path that is contained within a git repo.

Value

If the repo has a github remote, a list containing `username` and `repo`. Otherwise, an error.

Examples

```
gh_tree_remote()
```

gh_whoami *Info on current GitHub user and token*

Description

Reports wallet name, GitHub login, and GitHub URL for the current authenticated user, the first bit of the token, and the associated scopes.

Usage

```
gh_whoami(.token = NULL, .api_url = NULL, .send_headers = NULL)
```

Arguments

.token	Authentication token. Defaults to `GITHUB_PAT` or `GITHUB_TOKEN` environment variables, in this order if any is set. See `gh_token()` if you need more flexibility, e.g. different tokens for different GitHub Enterprise deployments.
.api_url	Github API url (default: https://api.github.com). Used if `endpoint` just contains a path. Defaults to `GITHUB_API_URL` environment variable if set.
.send_headers	Named character vector of header field values (except `Authorization`, which is handled via `.token`). This can be used to override or augment the default `User-Agent` header: `"https://github.com/r-lib/gh"`.

Details

Get a personal access token for the GitHub API from https://github.com/settings/tokens and select the scopes necessary for your planned tasks. The repo scope, for example, is one many are likely to need. The token itself is a string of 40 letters and digits. You can store it any way you like and provide explicitly via the .token argument to gh().

However, many prefer to define an environment variable GITHUB_PAT (or GITHUB_TOKEN) with this value in their .Renviron file. Add a line that looks like this, substituting your PAT:

```
GITHUB_PAT=8c70fd8419398999c9ac5bacf3192882193cadf2
```

Put a line break at the end! If you're using an editor that shows line numbers, there should be (at least) two lines, where the second one is empty. Restart R for this to take effect. Call gh_whoami() to confirm success.

To get complete information on the authenticated user, call gh("/user").

For token management via API (versus the browser), use the OAuth Authorizations API. This API requires Basic Authentication using your username and password, not tokens, and is outside the scope of the gh package.

Value

A gh_response object, which is also a list.

Examples

```
gh_whoami()


## explicit token + use with GitHub Enterprise
gh_whoami(.token = "8c70fd8419398999c9ac5bacf3192882193cadf2",
          .api_url = "https://github.foobar.edu/api/v3")
```

---

print.gh_response *Print the result of a GitHub API call*

Description

Print the result of a GitHub API call

Usage

```
## S3 method for class 'gh_response'
print(x, ...)
```

Arguments

x	The result object.
...	Ignored.

Value

The JSON result.

Index