# Package 'circle'

August 24, 2022

**Title** R Client Package for Circle CI

**Version** 0.7.2

**Description** Tools for interacting with the 'Circle CI' API (<https://circleci.com/docs/api/v2/>). Besides executing common tasks such as querying build logs and restarting builds, this package also helps setting up permissions to deploy from builds.

**License** GPL-3

**URL** https://docs.ropensci.org/circle/, https://github.com/ropensci/circle

**BugReports** https://github.com/ropensci/circle/issues

**Imports** cli (>= 2.0.0), httr, jsonlite

**Suggests** clipr, covr, gert, gh, knitr, openssl, purrr, rmarkdown, testthat (>= 3.0.0), usethis (>= 2.0.0), utils, vcr, withr

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Config/testthat/parallel** true

**Encoding** UTF-8

**RoxygenNote** 7.2.1

**NeedsCompilation** no

**Author** Patrick Schratz [aut, cre] (<https://orcid.org/0000-0003-0748-6624>), Max Joseph [rev] (Max reviewed the package for ropensci, see <https://github.com/ropensci/software-review/issues/356>), Sharla Gelfand [rev] (Sharla reviewed the package for ropensci, see <https://github.com/ropensci/software-review/issues/356>)

**Maintainer** Patrick Schratz <patrick.schratz@gmail.com>

**Repository** CRAN

**Date/Publication** 2022-08-24 07:50:02 UTC

# R topics documented:



---

`circle-package` *Circle CI API Client*

---

## Description

This package provides functionality for interacting with the Circle CI API. Circle CI is a continuous integration provider which allows for automated testing of software each time that software is publicly committed to a repository on GitHub.

This package interacts with the Circle CI REST API and allows to execute tasks in R without visiting the the website. This includes monitoring builds, modifying build environment settings and environment variables, and cancelling or restarting builds.

Use of this package requires a Circle API key. Unless a key is already set, users will be guided through the creation of a key, API keys are disposable, but should still be treated securely.

The following functions simplify integrating R package testing and deployment with GitHub and Circle CI:

- `enable_repo()` enables Circle CI for your repository,
- `use_circle_deploy()` installs a public deploy key on GitHub and the corresponding private key on Circle CI to simplify deployments to GitHub from Circle CI.

## Examples

```
## Not run:
# check to see if you've authenticated correctly
get_circle_user()

## End(Not run)
```

`browse_circle_token` *Authenticate to Circle CI*

## Description

A Circle CI API token is needed to interact with the Circle CI API. `browse_circle_token()` opens a browser window for the respective Circle CI endpoint to retrieve the key.

## Usage

```
browse_circle_token()
```

## Value

Returns `TRUE` (invisibly).

## Store API token

`circle` supports two ways of storing the Circle API tokens:

- via env vars `R_CIRCLE`
- via `~/.circleci/cli.yml`

The latter should already be present if you already used the `circle` CLI tool at some point in the past. If not, its up to your preference which approach to use.

The following instructions should help to set up `~/.circleci/cli.yml` correctly:

1. Copy the token from the browser after having called `browse_circle_token()`. You can use `edit_circle_config()` to open `~/.circleci/cli.yml`.
2. The token should be stored using the following structure

```
host: https://circleci.com
endpoint: graphql-unstable
token: <token>
```

## Examples

```
## Not run:
browse_circle_token()

edit_circle_config()

## End(Not run)
```

builds *Retrieve Metadata from Circle CI Builds*

## Description

Query information about pipelines, workflows or jobs on Circle CI. The S3 `print()` method for these functions returns the respective pipeline IDs. To inspect the details of each pipeline, save the return value in an object and inspect the respective sub-lists.

If no pipeline or workflow is supplied to `get_workflows()`/`get_jobs()`, the ten most recent pipelines/jobs are queried, respectively.

## Usage

```
get_pipelines(
  repo = github_info()$name,
  user = github_info()$owner$login,
  limit = 30,
  vcs_type = "gh",
  api_version = "v2"
)

get_workflows(
  pipeline_id = NULL,
  repo = github_info()$name,
  user = github_info()$owner$login
)

get_jobs(
  workflow_id = NULL,
  repo = github_info()$name,
  user = github_info()$owner$login,
  vcs_type = "gh"
)

retry_workflow(workflow_id = NULL)
```

## Arguments

| | |
|---|---|
| `repo` | `[character]`<br>The repository slug to use. Must follow the "`user/repo`" structure. |
| `user` | `[character]`<br>The username for the repository. By default queried using `get_user()`. |
| `limit` | `[integer]`<br>How many builds should be returned? Maximum allowed by Circle is 30. |
| `vcs_type` | `[character]`<br>The version control system to use. Defaults to "gh" (Github). |

| | |
|---|---|
| api_version | [character]<br>A character string specifying the Circle CI API version. This usually does not need to be changed by the user. |
| pipeline_id | [character]<br>A Circle CI pipeline ID. |
| workflow_id | [character]<br>A Circle CI workflow ID. |

## Details

While the `get_*()` functions query information about the respective build level details (pipeline - workflow - job), `retry_workflow()` let's users rerun a specific workflow. By default, the workflow from the most recent pipeline will be rerun if no pipeline ID was supplied.

## Value

An object of class `circle_collection` containing list information on the queried Circle CI pipelines/workflows/jobs.

## Examples

```
## Not run:
pipelines <- get_pipelines()

workflows <- get_workflows()

jobs <- get_jobs()

# rerun most recent workflow
retry_workflow()

## End(Not run)
```

---

# checkout_key *Interact with "Checkout Keys" on Circle CI*

---

## Description

Create, delete, query or check different types of checkout keys for a specific Circle CI project. Valid values for argument `type` are `"user-key"` or `"deploy-key"`.

A "Checkout Key" on Circle CI is a specific SSH key which is used to checkout repositories into a Circle CI build and possible deploy changes to the repository. Circle CI subdivides "Checkout Keys" into "user-key" and "deploy-key".

Please see "Deployment" section in the "Getting Started" vignette for more information.

## Usage

```
create_checkout_key(
  repo = github_info()$name,
  user = github_info()$owner$login,
  type = "user-key",
  api_version = "v2",
  vcs_type = "gh",
  quiet = FALSE
)

get_checkout_keys(
  repo = github_info()$name,
  user = github_info()$owner$login,
  vcs_type = "gh",
  api_version = "v2"
)

delete_checkout_key(
  fingerprint = NULL,
  repo = github_info()$name,
  user = github_info()$owner$login,
  type = "user-key",
  api_version = "v2",
  vcs_type = "gh"
)

has_checkout_key(
  repo = github_info()$name,
  user = github_info()$owner$login,
  type = "github-user-key",
  vcs_type = "gh",
  preferred = TRUE
)
```

## Arguments

| | |
|---|---|
| repo | [character]<br>The repository slug to use. Must follow the "user/repo" structure. |
| user | [character]<br>The username for the repository. By default queried using get_user(). |
| type | [character]<br>Type of key to add. Options are "user-key" and "deploy-key". |
| api_version | [character]<br>A character string specifying the Circle CI API version. This usually does not need to be changed by the user. |
| vcs_type | [character]<br>The version control system to use. Defaults to "gh" (Github). |

| | |
|---|---|
| quiet | [logical]<br>If TRUE, console output is suppressed. |
| fingerprint | [character]<br>The fingerprint of the checkout key which should be deleted. |
| preferred | [logical]<br>Checks whether the requested type is the "preferred" key. |

## Value

An object of class circle_api with the following elements

- content (queried content)
- path (API request)
- response (HTTP response information)

## Examples

```
## Not run:
# by default a "user-key" will be created which can also be used for build
# deployments
create_checkout_key()

# A "deploy-key" can only be used to checkout code from the repository into
# a Circle CI build
create_checkout_key(type = "deploy-key")

## End(Not run)
## Not run:
get_checkout_keys()

## End(Not run)
## Not run:
delete_checkout_key()

## End(Not run)
## Not run:
has_checkout_key()

## End(Not run)
```

---

# circle *Circle CI HTTP Requests*

---

## Description

Workhorse function for executing API requests to Circle CI.

## Usage

```
circle(
  verb = "GET",
  path = "",
  query = list(),
  body = "",
  api_version = "v2",
  encode = "json"
)
```

## Arguments

| | |
|---|---|
| verb | [character]<br>A character string containing an HTTP verb, defaulting to GET. |
| path | [character]<br>A character string with the API endpoint (should begin with a slash). |
| query | [character]<br>A list specifying any query string arguments to pass to the API. This is used to pass the API token. |
| body | [character]<br>A named list or array of what should be passed in the request. Corresponds to the "-d" argument of the curl command. |
| api_version | [character]<br>A character string specifying the Circle CI API version. This usually does not need to be changed by the user. |
| encode | [character]<br>Encoding format. See httr::POST. |

## Details

In almost all cases, users should not need to execute API calls directly. However, if desired this functions makes it possible to issue any API request. If you experience calling a custom request heavily, consider opening a feature request on GitHub.

## Value

An object of class circle_api with the following elements

- content (queried content)
- path (API request)
- response (HTTP response information)

## Examples

```
## Not run:
circle(verb = "GET", path = "/project/gh/ropensci/circle/checkout-key")

## End(Not run)
```

## edit_circle_config *Open circle Configuration file*

### Description

Opens `~/.circleci/cli.yml`.

### Usage

```
edit_circle_config()
```

### Value

No return value, called for side effects.

## enable_repo *Enable a repo on Circle CI*

### Description

"Follows" a repo on Circle CI so that builds can be triggered.

### Usage

```
enable_repo(
  repo = github_info()$name,
  user = github_info()$owner$login,
  vcs_type = "gh",
  api_version = "v1.1",
  quiet = FALSE
)
```

### Arguments

| | |
|---|---|
| `repo` | `[character]`<br>The repository slug to use. Must follow the "`user/repo`" structure. |
| `user` | `[character]`<br>The username for the repository. By default queried using `get_user()`. |
| `vcs_type` | `[character]`<br>The version control system to use. Defaults to "gh" (Github). |
| `api_version` | `[character]`<br>A character string specifying the Circle CI API version. This usually does not need to be changed by the user. |
| `quiet` | `[logical]`<br>If `TRUE`, console output is suppressed. |

## Value

An object of class `circle_api` with the following elements

- `content` (queried content)
- `path` (API request)
- `response` (HTTP response information)

## Examples

```
## Not run:
enable_repo()

## End(Not run)
```

---

`env_var` *Interact with Environment Variable(s) on Circle CI*

---

## Description

Add, get or set Circle CI environment variable(s) for a repo on Circle CI.

## Usage

```
get_env_vars(
  name = NULL,
  repo = github_info()$name,
  user = github_info()$owner$login,
  vcs_type = "gh",
  api_version = "v2"
)

set_env_var(
  var,
  repo = github_info()$name,
  user = github_info()$owner$login,
  vcs_type = "gh",
  api_version = "v2",
  quiet = FALSE
)

delete_env_var(
  var,
  repo = github_info()$name,
  user = github_info()$owner$login,
  vcs_type = "gh",
  api_version = "v2",
  quiet = FALSE
)
```

### Arguments

| | |
|---|---|
| name | [character]<br>Name of a specific environment variable. If not set, all env vars are returned. |
| repo | [character]<br>The repository slug to use. Must follow the "user/repo" structure. |
| user | [character]<br>The username for the repository. By default queried using get_user(). |
| vcs_type | [character]<br>The version control system to use. Defaults to "gh" (Github). |
| api_version | [character]<br>A character string specifying the Circle CI API version. This usually does not need to be changed by the user. |
| var | [list]<br>A list containing key-value pairs of environment variable and its value. |
| quiet | [logical]<br>If TRUE, console output is suppressed. |

### Value

An object of class circle_api with the following elements

- content (queried content)
- path (API request)
- response (HTTP response information)

### Examples

```
## Not run:
# get env var
get_env_vars()

# set env var
set_env_var(var = list("foo" = "123"))

# delete env var
delete_env_var("foo")

## End(Not run)
```

---

## get_build_artifacts *Get Build Artifacts of a Specific Job*

---

### Description

Retrieve artifacts from a specific build.

### Usage

```
get_build_artifacts(
  job_id = NULL,
  repo = github_info()$name,
  user = github_info()$owner$login,
  vcs_type = "gh",
  api_version = "v2"
)
```

### Arguments

| | |
|---|---|
| job_id | [character]<br>A Circle CI job id. |
| repo | [character]<br>The repository slug to use. Must follow the "user/repo" structure. |
| user | [character]<br>The username for the repository. By default queried using get_user(). |
| vcs_type | [character]<br>The version control system to use. Defaults to "gh" (Github). |
| api_version | [character]<br>A character string specifying the Circle CI API version. This usually does not need to be changed by the user. |

### Value

An object of class circle_api with the following elements

- content (queried content)
- path (API request)
- response (HTTP response information)

### Examples

```
## Not run:
job_id <- get_jobs()[[1]]$id
get_build_artifacts(job_id)

## End(Not run)
```

---

## get_circle_user *Get Circle CI user*

---

### Description

Retrieve details about the authenticated Circle CI user.

### Usage

```
get_circle_user()
```

### Value

A named vector of class `circle_user` containing information about about the authenticated user:

- Full name
- Username
- ID
- API endpoint

### Examples

```
## Not run:
get_circle_user()

## End(Not run)
```

---

## `list_projects` *List Circle CI Projects*

---

### Description

Retrieve a list of Circle CI repositories for the authenticated user.

### Usage

```
list_projects(repo = github_info()$name, user = github_info()$owner$login)
```

### Arguments

| | |
|---|---|
| `repo` | `[character]`<br>The repository slug to use. Must follow the "`user/repo`" structure. |
| `user` | `[character]`<br>The username for the repository. By default queried using `get_user()`. |

### Details

Retrieves a very detailed list of repository and repo-related information for all Circle CI repository attached to the current user.

This endpoint uses API v1.1 and will probably be removed in the near future.

## Value

An object of class `circle_api` with the following elements

- `content` (queried content)
- `path` (API request)
- `response` (HTTP response information)

## See Also

`get_pipelines()`, `get_workflows()`, `get_jobs()`

## Examples

```
## Not run:
list_projects()

## End(Not run)
```

---

`new_build` *Trigger a New Build on Circle CI*

---

## Description

Triggers a new build for a specific repo branch.

## Usage

```
new_build(
  repo = github_info()$name,
  user = github_info()$owner$login,
  vcs_type = "gh",
  branch = "master",
  quiet = FALSE
)
```

## Arguments

| | |
|---|---|
| `repo` | `[character]`<br>The repository slug to use. Must follow the "`user/repo`" structure. |
| `user` | `[character]`<br>The username for the repository. By default queried using `get_user()`. |
| `vcs_type` | `[character]`<br>The version control system to use. Defaults to "gh" (Github). |
| `branch` | A character string specifying the repository branch. |
| `quiet` | `[logical]`<br>If `TRUE`, console output is suppressed. |

### Details

Trigger a new Circle CI build for a specific repo branch.

### Value

An object of class `circle_api` with the following elements

- `content` (queried content)
- `path` (API request)
- `response` (HTTP response information)

### See Also

`retry_workflow()`

### Examples

```
## Not run:
new_build()

## End(Not run)
```

---

## use_circle_deploy *Set Up Build Deployment Between Circle CI And Github*

---

### Description

Creates a Circle CI "user-key" (= SSH key pair) if none exists yet to enable deployment from Circle CI builds to GitHub.

### Usage

```
use_circle_deploy(
  repo = github_info()$name,
  user = github_info()$owner$login,
  quiet = FALSE
)
```

### Arguments

| | |
|---|---|
| `repo` | `[character]`<br>The repository slug to use. Must follow the "`user/repo`" structure. |
| `user` | `[character]`<br>The username for the repository. By default queried using `get_user()`. |
| `quiet` | `[logical]`<br>If `TRUE`, console output is suppressed. |

## Details

The easiest way to achieve a deployment from Circle CI builds to a Github repo is by creating a so called "user-key" (i.e. an SSH key pair) on Circle CI.

`use_circle_deploy()` tries to be smart by exiting early if such a key is already present.

If the repo has not been enabled yet on Circle CI, please run `enable_repo()` first. Also to be able to authenticate to Github in the first place a personal access token needs to be set (via env var `GITHUB_TOKEN`). `usethis::github_token()` can be used to check if one is already set. If none is set, this function will prompt you to create one.

## Value

No return value, called for side effects.

## Examples

```
## Not run:
use_circle_deploy()

## End(Not run)
```

# Index