

# Package: lozen (via r-universe)

September 5, 2024

**Title** Management tools for missions

**Version** 1.3.0.9001

**Description** Management tools for missions (internal and external).  
Includes weekly, GL projects, etc.

**License** MIT + file LICENSE

**Depends** R (>= 3.5)

**Imports** attachment, bookdown (>= 0.27), cli, desc, dplyr, forcats, fs,  
gert, ggplot2, gh, gitlabr (>= 2.0.1.9000), glue, golem, httr,  
knitr (>= 1.39), lubridate, magrittr, pagedown, pkgdown, purrr,  
rconnect, scales, stats, stringi, stringr, tibble, tidyr,  
usethis, utils, withr, yaml, yesno

**Suggests** covr, devtools, DT, fusen, git2r, gitdown, htmltools,  
markdown, rcmdcheck, remotes, renv, rmarkdown (>= 2.14),  
testdown, testthat (>= 3.0.0), thinkrtemplate

**VignetteBuilder** knitr

**Remotes** ThinkR-open/gitlabr, ThinkR-open/testdown,  
ThinkR-open/thinkrtemplate

**Config/fusen/version** 0.5.2

**Config/testthat/edition** 3

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.1

**Repository** <https://thinkr-open.r-universe.dev>

**RemoteUrl** <https://github.com/ThinkR-open/lozen>

**RemoteRef** HEAD

**RemoteSha** c3f737060226d7519d90e4dd61610781b3caf617

## Contents

add_board . . . . .	3
add_board_github . . . . .	4
add_git_templates . . . . .	5
add_issue_clients . . . . .	5
add_issue_clients_github . . . . .	6
add_issue_dev . . . . .	7
add_issue_dev_github . . . . .	8
add_issue_kickoff . . . . .	8
add_issue_kickoff_github . . . . .	9
add_kit_package . . . . .	10
add_kit_project . . . . .	11
add_labels . . . . .	11
add_wikis . . . . .	12
add_wikis_github . . . . .	13
amend_yaml . . . . .	14
bs4_book_template . . . . .	14
build_pkgdown_with_reports . . . . .	15
check_if_yaml_exists . . . . .	16
clean_image . . . . .	17
clone_locally . . . . .	17
combine_ci . . . . .	18
create_book_project . . . . .	19
create_deploy_ci_stage . . . . .	20
create_group_project . . . . .	20
create_group_project_github . . . . .	21
create_issue_content_clients . . . . .	22
create_issue_content_clients_github . . . . .	23
create_issue_content_dev . . . . .	23
create_issue_content_dev_github . . . . .	24
create_issue_content_kickoff . . . . .	25
create_production . . . . .	25
create_r_project . . . . .	26
deploy_connect_bookdown . . . . .	27
deploy_connect_pkgdown . . . . .	29
deploy_connect_shiny . . . . .	31
fetch_connect . . . . .	33
gh_add_template_issue . . . . .	33
gh_create_weekly . . . . .	34
gh_create_weekly_new_projects_board . . . . .	35
gh_create_weekly_old_and_new_boards . . . . .	37
gl_add_template_issue . . . . .	39
gl_create_weekly . . . . .	40
gl_get_milestones_progress . . . . .	41
graphql_to_tibble . . . . .	42
html_to_odt . . . . .	43
init_project_with_all . . . . .	43

<i>add_board</i>	3
modify_autoclose_and_coverage . . . . .	44
move_issues_from_gitlab_to_github . . . . .	45
paged_template . . . . .	47
project_options . . . . .	48
protect_branches . . . . .	48
push_all_to_branch . . . . .	49
push_main . . . . .	50
push_master . . . . .	50
read_ci . . . . .	51
render_book . . . . .	51
use_dev_history . . . . .	52
use_gitlab_ci . . . . .	53
use_gitlab_ci_deploy_connect . . . . .	54
visualise_commits . . . . .	55
with_gitlab_project . . . . .	57
<b>Index</b>	<b>59</b>

---

<i>add_board</i>	<i>Add board in the correct order</i>
------------------	---------------------------------------

---

**Description**

Add board in the correct order

**Usage**

```
add_board(  
  project_id,  
  name = "Development",  
  labels_order = c("Bloqué", "Prêt", "En cours", "Révision", "Pré-validation",  
    "A valider"),  
  lg = "fr"  
)
```

**Arguments**

project_id	project_id
name	Name of the Board
labels_order	Name of the labels already existing in the project
lg	language of the board, "fr" or "en"

### Examples

```
## Not run:
add_board(
  project_id = project_id
)

## End(Not run)
```

---

add_board_github	<i>add_board_github</i>
------------------	-------------------------

---

### Description

add\_board\_github

### Usage

```
add_board_github(
  owner,
  repo,
  columns = c("Open", "Blocked", "Meta", "Ready", "In Progress", "Review", "Validation")
)
```

### Arguments

owner	Owner of the repository
repo	Repository name
columns	Names of columns to create in the Board

### Examples

```
## Not run:
add_board_github(
  owner = "ghowner",
  repo = "areponame"
)

## End(Not run)
```

---

add_git_templates	<i>Add MR GitLab or GitHub template and local issue template</i>
-------------------	--

---

**Description**

Add MR GitLab or GitHub template and local issue template

**Usage**

```
add_git_templates(  
  project_path = ".",  
  type = c("commit", "mr"),  
  target_dir = ".gitlab"  
)
```

**Arguments**

project_path	project_path
type	type
target_dir	Directory name where to save templates

**Value**

Side effect: template in .gitlab/.github and in .git

**Examples**

```
## Not run:  
add_git_templates(  
  project_path = project_path,  
  type = c("commit", "mr")  
)  
  
## End(Not run)
```

---

add_issue_clients	<i>Add First issue client for GitLab</i>
-------------------	--

---

**Description**

Add First issue client for GitLab

**Usage**

```
add_issue_clients(project_id, project_name, group_url)
```

**Arguments**

project_id	project_id
project_name	project_name
group_url	group_url

**Value**

A tibble with the issue added and GitLab infos

**Examples**

```
## Not run:
add_issue_clients(
  project_id = "<get_your_id_project>",
  project_name = "<get_your_project_name>",
  group_url = "<group_url_repo>" # should looks like "https://gitlab.com/cervan.girard/"
)

## End(Not run)
```

---

add\_issue\_clients\_github

*Add First issue client for GitHub*

---

**Description**

Add First issue client for GitHub

**Usage**

```
add_issue_clients_github(owner, repo)
```

**Arguments**

owner	GitHub owner (username or organisation)
repo	Name of the repository

**Value**

A tibble with the issue added and GitLab infos

### Examples

```
## Not run:
add_issue_clients_github(
  owner = "owner",
  repo = "repo"
)

## End(Not run)
```

---

add\_issue\_dev

*Create issue for developers to initiate the project*

---

### Description

Create issue for developers to initiate the project

### Usage

```
add_issue_dev(project_id, project_name, group_url)
```

### Arguments

project_id	project_id
project_name	project_name
group_url	group_url

### Value

Side effect: new issue on GitLab

### Examples

```
# exemple nécessite des opérations sur gitlab : à voir
## Not run:
add_issue_dev(
  project_id = "<get_your_id_project>",
  project_name = "<get_your_project_name>",
  group_url = "<group_url_repo>" # should looks like "https://gitlab.com/cervan.girard/"
)

## End(Not run)
```

---

add\_issue\_dev\_github    *Create issue for dev on github*

---

**Description**

Create dev first issue in github

**Usage**

```
add_issue_dev_github(owner, repo)
```

**Arguments**

owner	owner of the project
repo	name of the project

**Value**

tibble with github issue info

**Examples**

```
## Not run:  
add_issue_dev_github(  
  owner = "owner",  
  repo = "repo"  
)  
  
## End(Not run)
```

---

add\_issue\_kickoff    *Add Kickoff issue client for GitLab*

---

**Description**

Add Kickoff issue client for GitLab

**Usage**

```
add_issue_kickoff(project_id)
```

**Arguments**

project_id	project_id
------------	------------



**Value**

A tibble with the issue added and GitLab infos

**Examples**

```
## Not run:  
add_issue_kickoff(project_id = "<get_your_id_project>")  
  
## End(Not run)
```

---

```
add_issue_kickoff_github  
      Add Kickoff issue for GitHub
```

---

**Description**

Add Kickoff issue for GitHub

**Usage**

```
add_issue_kickoff_github(owner, repo)
```

**Arguments**

owner	GitHub owner (username or organisation)
repo	Name of the repository

**Value**

A tibble with the issue added and GitHub infos

**Examples**

```
## Not run:  
add_issue_kickoff_github(  
  owner = "owner",  
  repo = "repo"  
)  
  
## End(Not run)
```

---

add_kit_package	<i>Add files necessary for a package, including golem</i>
-----------------	---

---

## Description

Add files necessary for a package, including golem

## Usage

```
add_kit_package(  
  project_path = ".",  
  type = c("package", "renv"),  
  pkgdown_yaml = NULL  
)
```

## Arguments

project_path	project_path
type	Type of dev_history. Multiple types possible among "package", "book", "renv".
pkgdown_yaml	path to yaml conf for pkgdown

## Value

Side Effect, add new files :

- dev/dev\_history
- templates for pkgdown if pkgodwn\_yaml is not null
- testthat

## Examples

```
withr::with_tempdir({  
  project_path <- getwd()  
  usethis::create_package(path = project_path, open = FALSE)  
  add_kit_package(project_path, type = c("package", "book", "renv"))  
})
```

---

add_kit_project	<i>Add necessary files for any R project</i>
-----------------	--

---

**Description**

Add necessary files for any R project

**Usage**

```
add_kit_project(project_path = ".", name_licence, type_licence)
```

**Arguments**

project_path	project_path
name_licence	Name for the licence
type_licence	should be a function, example : type_licence = usethis::use_proprietary_license

**Value**

Side Effect, add new files :

- gitattributes
- NEWS.md

**Examples**

```
withr::with_tempdir({  
  project_path <- getwd()  
  usethis::create_project(path = project_path, open = FALSE)  
  add_kit_project(  
    project_path,  
    name_licence = "Bob",  
    type_licence = usethis::use_proprietary_license  
  )  
})
```

---

add_labels	<i>Add labels to project</i>
------------	------------------------------

---

**Description**

Add labels to project

**Usage**

```
add_labels(project_id, lg = "fr")
```

Arguments

project_id	project_id
lg	by default, fr to get labels in french. Use en for english

Value

Table of added labels. Side effect: labels added on GitLab.

Examples

```
## Not run:
add_labels(
  project_id = project_id
)

## End(Not run)
```

---

add_wikis	<i>Add Wiki</i>
-----------	-----------------

---

Description

Add Wiki

Usage

```
add_wikis(
  project_id,
  project_name,
  group_url,
  group = basename(group_url),
  type = c("home", "cr", "keys")
)
```

Arguments

project_id	project_id
project_name	project_name
group_url	group_url
group	group
type	type

**Details**

Types:

- home: Home Page
- cr: Comptes-rendus
- keys: Key dates of the project

**Value**

Tibble with wikis and GitLab infos.

**Examples**

```
## Not run:
add_wikis(
  project_id = project_id,
  project_name = project_name,
  group_url = group_url,
  group = basename(group_url)
)

## End(Not run)
```

---

add_wikis_github	<i>add_wikis_github</i>
------------------	-------------------------

---

**Description**

add\_wikis\_github

**Usage**

```
add_wikis_github(owner, repo, type = c("home", "cr", "keys", "weekly"))
```

**Arguments**

owner	Owner of the repository
repo	Repository name
type	Wiki type to add

**Examples**

```
## Not run:
add_wikis_github(
  owner = "ghowner",
  repo = "areponame"
)

## End(Not run)
```

---

amend\_yaml

*Amend yaml*


---

### Description

combien, if needed, to an existing .gitlab-ci.yml file

### Usage

```
amend_yaml(path_to_yaml, connect_ci_list)
```

### Arguments

```
path_to_yaml    pat to .gitlab-ci.yml file
connect_ci_list list, output of create_deploy_ci_stage
```

### Examples

```
# amend_yaml()
```

---

bs4\_book\_template

*Render bookdown bs4book*


---

### Description

Render bookdown bs4book

### Usage

```
bs4_book_template(
  theme = bs4_book_theme(),
  repo = NULL,
  ...,
  lib_dir = "libs",
  pandoc_args = NULL,
  extra_dependencies = NULL,
  template = "default",
  split_bib = FALSE,
  footnotes_inline = TRUE
)
```

**Arguments**

theme	A named list or <code>bslib::bs_theme()</code> object. The default, <code>bs4_book_theme()</code> , resets the base font size to 1rem to make reading easier and uses a primary colour with greater contrast against the background.
repo	Either link to repository where book is hosted, used to generate view source and edit buttons or a list with repository base link, default branch, subdir and icon (see "Specifying the repository" in <a href="https://bookdown.org/yihui/bookdown/html.html#bootstrap4-style">https://bookdown.org/yihui/bookdown/html.html#bootstrap4-style</a> ).
lib_dir, pandoc_args, extra_dependencies, split_bib, ...	Passed on to <code>rmarkdown::html_document()</code> .
template	Pandoc template to use for rendering. Pass "default" to use the bookdown default template; pass a path to use a custom template. The default template should be sufficient for most use cases. For advanced user only, in case you want to develop a custom template, we highly recommend to start from the default template: <a href="https://github.com/rstudio/bookdown/blob/master/inst/templates/bs4_book.html">https://github.com/rstudio/bookdown/blob/master/inst/templates/bs4_book.html</a> . Otherwise, some feature may not work anymore.
footnotes_inline	By default, footnotes will be set inline and shown on hover. Set to FALSE to keep footnotes at the bottom of the page with links.

**Value**

A pagedown template

---

build\_pkgdown\_with\_reports

*Build a pkgdown with extra reports tab for testdown and coverage output*

---

**Description**

Build a pkgdown with extra reports tab for testdown and coverage output

**Usage**

```
build_pkgdown_with_reports(
  pkg = ".",
  pkgdown_path = "public",
  assets_path = "pkgdown/assets",
  reports = c("coverage", "testdown", "gitdown"),
  git_branch_ref = "main",
  overwrite_assets = TRUE
)
```

**Arguments**

pkg	character	The path to the package to be build and reported
pkgdown_path	character	The relative path inside the package to store the final site
assets_path	character	The relative path within the package to store the reports outputs
reports	character	A vector of reports to be produced, can be any subset of c("testdown", "coverage")
git_branch_ref	character	A vector of the git branch to use, 'main' by default
overwrite_assets	logical	Whether the assets directory to store reports should be overwritten

**Value**

None Generate a pkgdown with test and coverage reports

**Examples**

```
## Not run:
# build_pkgdown_with_reports(
#   pkg = ".",
#   pkgdown_path = "public",
#   assets_path = "pkgdown/assets",
#   reports = c("testdown", "coverage")
# )

## End(Not run)
```

---

check\_if\_yaml\_exists    *Check if yaml file exist*

---

**Description**

Check if yaml file already exist and ask permission for overwriting

**Usage**

```
check_if_yaml_exists(dir = ".", file_name = ".gitlab-ci.yml", append = TRUE)
```

**Arguments**

dir	directory to scan
file_name	filename to use .gitlab-ci.yml
append	boolean do we allow to append stage to existing ci file. TRUE by default

**Examples**

```
# check_if_yaml_exists()
```



---

clean_image	<i>clean image</i>
-------------	--------------------

---

**Description**

Manage image in yaml

**Usage**

```
clean_image(ci)
```

**Arguments**

ci	List of CI parameters, imported from yaml file
----	--

**Value**

list the list of CI parameters with a unique docker image

**Examples**

```
# pkgdown yaml
full <- yaml::read_yaml(file = system.file("yaml", ".gitlab-ci-pkg.yaml", package = "lozen"))

# shiny yaml
connect <- yaml::read_yaml(file = system.file("yaml", ".gitlab-ci-shiny.yaml", package = "lozen"))

ci_list <- combine_ci(ci1 = full, ci2 = connect)
ci_list <- clean_image(ci_list)
```

---

clone_locally	<i>Clone project locally</i>
---------------	------------------------------

---

**Description**

Clone project locally

**Usage**

```
clone_locally(project_name, group_url, full_url, project_path, open = TRUE)
```

**Arguments**

project_name	project_name
group_url	group_url
full_url	Full url to the repository. Superseeds group_url and project_name if used.
project_path	project_path
open	Logical. Whether to open the RStudio project.

**Value**

project\_path. Side effect: clone the project.

**Examples**

```
## Not run:
project_path <- clone_locally(
  project_name = the_project[["name"]],
  group_url = group_url,
  open = TRUE
)

## End(Not run)
```

---

 combine\_ci

*combine CI*


---

**Description**

Merge two yaml as a list into a combined list

**Usage**

```
combine_ci(ci1, ci2)
```

**Arguments**

ci1	List of initial CI parameters
ci2	List of CI parameters to append to the initial CI

**Value**

list list of combined CI parameters

**Examples**

```
# pkgdown yaml
full <- yaml::read_yaml(file = system.file("yaml", ".gitlab-ci-pkg.yaml", package = "lozen"))

# shiny yaml
connect <- yaml::read_yaml(file = system.file("yaml", ".gitlab-ci-shiny.yaml", package = "lozen"))

ci_list <- combine_ci(ci1 = full, ci2 = connect)
```

---

create_book_project	<i>Transform project as book with lozen template</i>
---------------------	--

---

## Description

Transform project as book with lozen template

## Usage

```
create_book_project(  
  project_path,  
  bookdown_path = system.file("lozendown", package = "lozen"),  
  css = NULL,  
  footer = NULL,  
  logo = NULL,  
  index = NULL,  
  output_yaml = NULL  
)
```

## Arguments

project_path	project_path
bookdown_path	a path to a bookdown
css	path(s) for css to copy
footer	path for footer to copy
logo	path for logo to copy
index	path for index to copy
output_yaml	path for output_yaml to copy

## Value

Side Effect: Transform project as book

## Examples

```
withr::with_tempdir({  
  project_path <- getwd()  
  create_book_project(project_path)  
})
```

---

```
create_deploy_ci_stage
```

*Title*

---

### Description

Title

### Usage

```
create_deploy_ci_stage(
  image,
  deploy_function,
  stage_name = deploy_function,
  ...
)
```

### Arguments

image	image#'
deploy_function	deploy_function
stage_name	stage_name
...	dots not used

### Examples

```
create_deploy_ci_stage(
  image = "rocker/verse",
  deploy_function = "deploy_connect_shiny"
)
```

---

```
create_group_project
```

*Create new project in a group*

---

### Description

Create new project in a group

### Usage

```
create_group_project(
  project_name,
  namespace_id,
  gitlab_con = "default",
  default_branch = "main"
)
```

**Arguments**

project_name	name of the project to create
namespace_id	id of the group in which to add the project
gitlab_con	Connection credentials to GitLab. Better use set_gitlab_connection() before this function
default_branch	Default branch for the project created. Default to "main".

**Value**

project\_id. Side effect: Create a project on GitLab if not exists.

**Examples**

```
## Not run:
create_group_project(
  project_name = project_name,
  namespace_id = namespace_id,
  default_branch = "main"
)

## End(Not run)
```

---

create\_group\_project\_github  
*Create new project on GitHub*

---

**Description**

Create new project on GitHub

**Usage**

```
create_group_project_github(repo, owner)
```

**Arguments**

repo	name of the project to create
owner	Owner of the repo

**Value**

project\_id. Side effect: Create a project on GitHub if not exists.

**Examples**

```
## Not run:
create_group_project_github(
  repo = "areponame",
  owner = "ghowner"
)

## End(Not run)
```

---

create\_issue\_content\_clients

*Create issue content for client*

---

**Description**

Generate text for client first issue in gitlab

**Usage**

```
create_issue_content_clients(project_name, group_url)
```

**Arguments**

project_name	name of the project
group_url	url to gitlab website

**Value**

character vector of the issue content

**Examples**

```
## Not run:
create_issue_content_clients(
  project_name = "<get_your_project_name>",
  group_url = "<group_url_repo>" # should looks like "https://gitlab.com/cervan.girard/"
)

## End(Not run)
```

---

```
create_issue_content_clients_github
```

*Create issue content for client*

---

**Description**

Generate text for client first issue in github

**Usage**

```
create_issue_content_clients_github(owner, repo)
```

**Arguments**

owner	owner of the project
repo	name of the project

**Value**

character vector of the issue content

**Examples**

```
## Not run:  
create_issue_content_clients_github(  
  owner = "owner",  
  repo = "repo"  
)  
  
## End(Not run)
```

---

```
create_issue_content_dev
```

*Create issue content for devs*

---

**Description**

Generate text for devs first issue in gitlab

**Usage**

```
create_issue_content_dev(project_id, group_url, project_name)
```

**Arguments**

project_id	project_id
group_url	group_url
project_name	project_name

**Value**

character vector of the issue content

**Examples**

```
## Not run:
create_issue_content_dev(
  project_id = "<get_your_id_project>",
  project_name = "<get_your_project_name>",
  group_url = "<group_url_repo>" # should looks like "https://gitlab.com/cervan.girard/"
)

## End(Not run)
```

---

```
create_issue_content_dev_github
      Create issue content for dev
```

---

**Description**

Generate text for dev first issue in github

**Usage**

```
create_issue_content_dev_github(owner, repo)
```

**Arguments**

owner	owner of the project
repo	name of the project

**Value**

character vector of the issue content



**Examples**

```
## Not run:
create_issue_content_dev_github(
  owner = "owner",
  repo = "repo"
)

## End(Not run)
```

---

```
create_issue_content_kickoff
      Create issue content for kickoff
```

---

**Description**

Generate text for kickoff issue

**Usage**

```
create_issue_content_kickoff()
```

**Value**

character vector of the issue content

**Examples**

```
create_issue_content_kickoff()
```

---

```
create_production      Create branch production from master
```

---

**Description**

Create branch production from master

**Usage**

```
create_production(project_path)
```

**Arguments**

project\_path    project\_path

**Value**

Side effect: New branch on the server

## Examples

```
## Not run:
create_production(
  project_path = project_path
)

## End(Not run)
```

---

create_r_project	<i>Create a R project with full content skeleton</i>
------------------	--

---

## Description

Create a R project with full content skeleton

## Usage

```
create_r_project(
  project_path,
  type = c("package", "golem", "book"),
  name_licence,
  type_licence
)
```

## Arguments

project_path	project_path
type	type
name_licence	name for the licence
type_licence	type of the licence, should be a function like <code>usethis::use_mit_licence</code>

## Value

Side Effect: Transform project as package or book:

- With package, golem or book skeleton
- git necessary files
- pkgdown or book template
- GitLab CI

## Examples

```
## Not run:
# path to your local project
project_path <- tempfile("my.local.project")

### R package skeleton
create_r_project(
  project_path = project_path,
  type = c("package"),
  name_licence = "Bob",
  type_licence = usethis::use_proprietary_license
)

### Shiny app
create_r_project(
  project_path = project_path,
  type = c("golem"),
  name_licence = "Bob",
  type_licence = usethis::use_proprietary_license
)

### Bookdown
create_r_project(
  project_path = project_path,
  type = c("book"),
  name_licence = "Bob",
  type_licence = usethis::use_proprietary_license
)

## End(Not run)
```

---

deploy\_connect\_bookdown

*deploy\_connect\_bookdown*

---

## Description

Before using it, please follow these steps :

## Usage

```
deploy_connect_bookdown(
  connect_url = Sys.getenv("CONNECT_URL"),
  connect_user = Sys.getenv("CONNECT_USER"),
  connect_api_token = Sys.getenv("CONNECT_TOKEN"),
  app_name = NULL,
  deploy_dir = file.path(getwd(), "_book"),
  connect_name = Sys.getenv("CONNECT_NAME", unset = "connect"),
  file_to_ignore_regex =
```

```

    ".Rprofile$|^\.Renvirons$|renv/|rstudio_.*|/deliverables/|dev/|data-raw/|dockerfiles/",
    forceUpdate = FALSE,
    lint = FALSE,
    ...
  )

```

## Arguments

<code>connect_url</code>	URL of the Connect server
<code>connect_user</code>	User name to use to connect to the Connect server
<code>connect_api_token</code>	API token to use to connect to the Connect server
<code>app_name</code>	Name of the app to deploy
<code>deploy_dir</code>	Directory to deploy
<code>connect_name</code>	Name of the Connect server
<code>file_to_ignore_regex</code>	Regex to use to ignore files
<code>forceUpdate</code>	What should happen if there's no deployment record for the app, but there's an app with the same name on the server? If TRUE, will always update the previously-deployed app. If FALSE, will ask the user what to do, or fail if not in an interactive context.  Defaults to TRUE when called automatically by the IDE, and FALSE otherwise. You can override the default by setting option <code>rsconnect.force.update.apps</code> .
<code>lint</code>	Lint the project before initiating deployment, to identify potentially problematic code?
<code>...</code>	Other arguments to pass to <code>rsconnect::deployApp</code>

## Details

- Ask the Mission Lead Dev for their deployment token on Connect, this is the one you will need to use.
- Add the environment variables to your personal ".Renvirons" to manually deploy to Connect:
  - Add `CONNECT_USER` with username.
  - Add `CONNECT_TOKEN` with the token.
  - Add `CONNECT_URL` with the connect url

TODO

## Value

used for side effects

## Examples

```
## Not run:
if (
  Sys.getenv("CONNECT_URL") != "" &
  Sys.getenv("CONNECT_USER") != "" &
  Sys.getenv("CONNECT_TOKEN") != ""
) {
  project_name <- "lozen-example-bookdown"

  tmpdir <- tempfile(pattern = "book-")
  dir.create(tmpdir)
  project_path <- file.path(tmpdir, project_name)

  bookdown::create_bs4_book(path = project_path)

  bookdown::render_book(input = project_path)

  deploy_connect_bookdown(
    app_name = project_name,
    deploy_dir = file.path(project_path, "_book")
  )
}

## End(Not run)
```

---

deploy\_connect\_pkgdown

*deploy\_connect\_pkgdown*

---

## Description

Before using it, please follow these steps :

## Usage

```
deploy_connect_pkgdown(
  connect_url = Sys.getenv("CONNECT_URL"),
  connect_user = Sys.getenv("CONNECT_USER"),
  connect_api_token = Sys.getenv("CONNECT_TOKEN"),
  app_name = NULL,
  deploy_dir = c(file.path(getwd(), "public"), file.path(getwd(), "docs"),
    file.path(getwd(), "inst/site/"), file.path(getwd(), ".")),
  connect_name = Sys.getenv("CONNECT_NAME", unset = "connect"),
  file_to_ignore_regex =
    ".Rprofile$|^\\.Renv$|^renv/|^rstudio_.*|^deliverables/|^dev/|^data-raw/|^dockerfiles/",
  forceUpdate = FALSE,
  lint = FALSE,
  app_primary_doc = "index.html",
```

```
    ...  
  )
```

### Arguments

connect_url	URL of the Connect server
connect_user	User name to use to connect to the Connect server
connect_api_token	API token to use to connect to the Connect server
app_name	Name of the app to deploy
deploy_dir	vector of Directory path to deploy, the first available will be used. default is <code>c(file.path(getwd(), "public"), file.path(getwd(), "docs"), file.path(getwd(), "inst/site/"), file.path(getwd(), "."))</code>
connect_name	Name of the Connect server
file_to_ignore_regex	Regex to use to ignore files
forceUpdate	What should happen if there's no deployment record for the app, but there's an app with the same name on the server? If TRUE, will always update the previously-deployed app. If FALSE, will ask the user what to do, or fail if not in an interactive context.  Defaults to TRUE when called automatically by the IDE, and FALSE otherwise. You can override the default by setting option <code>rsconnect.force.update.apps</code> .
lint	Lint the project before initiating deployment, to identify potentially problematic code?
app_primary_doc	If the application contains more than one document, this parameter indicates the primary one, as a path relative to <code>appDir</code> . Can be NULL, in which case the primary document is inferred from the contents being deployed.
...	Other arguments to pass to <code>rsconnect::deployApp</code>

### Details

- Ask the Mission Lead Dev for their deployment token on Connect, this is the one you will need to use.
- Add the environment variables to your personal ".Renviron" to manually deploy to Connect:
  - Add `CONNECT_USER` with username.
  - Add `CONNECT_TOKEN` with the token.
  - Add `CONNECT_URL` with the connect url

TODO

### Value

used for side effects

## Examples

```
## Not run:
# We assume that you are working on a R package
# if not done yet, create your pkgdown
pkgdown::build_site(
  pkg = ".",
  override = list(destination = "inst/site/")
)
if (Sys.getenv("CONNECT_URL") != "" &
    Sys.getenv("CONNECT_USER") != "" &
    Sys.getenv("CONNECT_TOKEN") != "") {
  deploy_connect_pkgdown(
    app_name = "titi",
    deploy_dir = file.path(project_path, "inst/site/")
  )
}

## End(Not run)
```

---

deploy\_connect\_shiny    *deploy\_connect*

---

## Description

Before using it, please follow these steps :

## Usage

```
deploy_connect_shiny(
  connect_url = Sys.getenv("CONNECT_URL"),
  connect_user = Sys.getenv("CONNECT_USER"),
  connect_api_token = Sys.getenv("CONNECT_TOKEN"),
  app_name = NULL,
  deploy_dir = getwd(),
  connect_name = Sys.getenv("CONNECT_NAME", unset = "connect"),
  file_to_ignore_regex =
    ".Rprofile$|^\\.Renv$|^\\.renv$|^\\.rstudio.*$|^\\.deliverables$|^\\.dev$|^\\.data-raw$|^\\.dockerfiles$",
  forceUpdate = FALSE,
  lint = FALSE,
  ...
)
```

## Arguments

connect_url	URL of the Connect server
connect_user	User name to use to connect to the Connect server
connect_api_token	API token to use to connect to the Connect server

<code>app_name</code>	Name of the app to deploy
<code>deploy_dir</code>	Directory to deploy
<code>connect_name</code>	Name of the Connect server
<code>file_to_ignore_regex</code>	Regex to use to ignore files
<code>forceUpdate</code>	What should happen if there's no deployment record for the app, but there's an app with the same name on the server? If TRUE, will always update the previously-deployed app. If FALSE, will ask the user what to do, or fail if not in an interactive context.  Defaults to TRUE when called automatically by the IDE, and FALSE otherwise. You can override the default by setting option <code>rsconnect.force.update.apps</code> .
<code>lint</code>	Lint the project before initiating deployment, to identify potentially problematic code?
<code>...</code>	Other arguments to pass to <code>rsconnect::deployApp</code>

### Details

- Ask the Mission Lead Dev for their deployment token on Connect, this is the one you will need to use.
- Add the environment variables to your personal ".Renviron" to manually deploy to Connect:
  - Add `CONNECT_USER` with username.
  - Add `CONNECT_TOKEN` with the token.
  - Add `CONNECT_URL` with the connect url

TODO

### Value

used for side effects

### Examples

```
## Not run:
deploy_connect_shiny(
  connect_url = Sys.getenv("CONNECT_URL"),
  connect_user = Sys.getenv("CONNECT_USER"),
  connect_api_token = Sys.getenv("CONNECT_TOKEN"),
  app_name = "app_test"
)

## End(Not run)
```



---

fetch_connect	<i>fetch_connect</i>
---------------	----------------------

---

**Description**

Fetch deployment status on Connect of a given app

**Usage**

```
fetch_connect(
  app_name,
  connect_api_token = Sys.getenv("CONNECT_TOKEN"),
  connect_url = Sys.getenv("CONNECT_URL")
)
```

**Arguments**

app_name	character Name of the app to be fetched
connect_api_token	character API token to enable access to Connect
connect_url	character URL to Connect server

**Value**

list A list containing the HTTP code status and the guid of the app

**Examples**

```
## Not run:
fetch_connect(app_name = "my_app_name")

## End(Not run)
```

---

gh_add_template_issue	<i>Create the template for standard issue on gitlab</i>
-----------------------	---

---

**Description**

Create the template for standard issue on gitlab

**Usage**

```
gh_add_template_issue(
  project_path = ".",
  language = c("fr", "en"),
  type = c("full", "light")
)
```

**Arguments**

project_path	Path to the project
language	Language. Should be one of 'fr' or 'en'
type	Verbosity of comments in the template. Should be 'full' or 'light'.

**Value**

Side effect: create the template

**Examples**

```
# Try with a temp project
## Create temp dir and file
project_path <- tempfile("myproject")
dir.create(project_path)

## Add the full template in french
gh_add_template_issue(project_path, language = "fr", type = "full")

## Add the full template in english
gh_add_template_issue(project_path, language = "en", type = "full")

## Add the light template in french
gh_add_template_issue(project_path, language = "fr", type = "light")

## Add the light template in english
gh_add_template_issue(project_path, language = "en", type = "light")

# suppress tmp folder
unlink(project_path, recursive = TRUE)
```

---

gh\_create\_weekly

*Create a weekly issues summary for GitHub*

---

**Description**

Create a weekly issues summary for GitHub

**Usage**

```
gh_create_weekly(
  date_min = Sys.Date() - 7,
  date_max = Sys.Date(),
  user = "thinkr-open",
  repo = "fusen",
  verbose = FALSE,
  board_url
)
```

**Arguments**

date_min	Minimal date to look for issues
date_max	Maximal date to look for issues
user	username or company name as shown on GitHub
repo	GitHub repository
verbose	Logical. Whether to return output in the console too.
board_url	url of the GitHub board. If using the "classic" board, which is deprecated by GitHub, you may want to use this url: <code>glue("/repos/{user}/{repo}/projects")</code>

**Value**

A Weekly to copy-paste in a Wiki and a tibble

**Examples**

```
## Not run:
if (Sys.getenv("GITHUB_PAT") != "") {
  user <- "ThinkR-open"
  repo <- "example-weekly"
  weekly <- gh_create_weekly(
    date_min = "2022-06-30",
    date_max = "2022-06-30",
    user = user,
    repo = repo,
    board_url = glue::glue("/repos/{user}/{repo}/projects")
  )
}

## End(Not run)
# Copier dans le presse papier pour copier directement
# clipr::write_clip(weekly$weekly_info)
```

---

gh\_create\_weekly\_new\_projects\_board

*Create a weekly issues summary for GitHub (new board)*

---

**Description**

Create a weekly issues summary for GitHub (new board)

**Usage**

```
gh_create_weekly_new_projects_board(
  date_min,
  date_max,
  user,
```

```

    repo,
    board_url,
    github_token = Sys.getenv("GITHUB_PAT"),
    verbose = FALSE
  )

```

### Arguments

<code>date_min</code>	Minimal date to look for issues
<code>date_max</code>	Maximal date to look for issues
<code>user</code>	username or company name as shown on GitHub
<code>repo</code>	GitHub repository
<code>board_url</code>	url of the GitHub board.
<code>github_token</code>	token access to graphQL API
<code>verbose</code>	Logical. Whether to return output in the console too.

### Value

A Weekly to copy-paste in a Wiki and a tibble

### Examples

```

## Not run:
board_url <- "https://github.com/orgs/ThinkR-open/projects/4/"
date_min <- "2022-06-30"
date_max <- "2022-06-30"
user <- "ThinkR-open"
repo <- "example-weekly"

if (Sys.getenv("GITHUB_PAT") != "") {
  weekly <- gh_create_weekly_new_projects_board(
    date_min = date_min,
    date_max = date_max,
    user = user,
    repo = repo,
    board_url = board_url,
    github_token = Sys.getenv("GITHUB_PAT"),
    verbose = FALSE
  )
}

## End(Not run)
# Copier dans le presse papier pour copier directement
# clipr::write_clip(weekly$weekly_info)

```

---

gh\_create\_weekly\_old\_and\_new\_boards

*Create a weekly issues summary for GitHub*

---

## Description

Create a weekly issues summary for GitHub

## Usage

```
gh_create_weekly_old_and_new_boards(
    date_min,
    date_max,
    user,
    repo,
    board_url,
    new_board = TRUE,
    regex_done = "close|closed|done",
    regex_validation = "a valider|validation",
    regex_blocked = "blocked|bloque|bloqu\\\\\\u00e9",
    regex_inprogress =
        "in progress|en cours|review|revision|r\\\\\\u00e9vision|r\\\\\\u00e9-validation",
    regex_ready = "ready|pret|pr\\\\\\u00eat",
    github_token = Sys.getenv("GITHUB_PAT"),
    verbose = FALSE
)
```

## Arguments

date_min	Minimal date to look for issues
date_max	Maximal date to look for issues
user	username or company name as shown on GitHub
repo	GitHub repository
board_url	url of the GitHub board. If using the "classic" board, which is deprecated by GitHub, you may want to use this url: glue("/repos/{user}/{repo}/projects") otherwise you may want to use this kind of url: "https://github.com/orgs/myorganization/projects" Please not this not mandatory if you are setting new_board = FALSE since board_url can be guessed from user and repo
new_board	logical. Are you working with the classic board or the new board.
regex_done	Regular expression to detect labels for issues done by developers (and potentially awaiting validation by clients)
regex_validation	Regular expression to detect labels for issues waiting for validation
regex_blocked	Regular expression to detect labels for issues blocked (and potentially awaiting information by clients)

regex_inprogress	Regular expression to detect labels for issues in progress (and potentially awaiting review by lead dev.)
regex_ready	Regular expression to detect labels for issues ready (and potentially reorder by lead dev.)
github_token	token access to github API (read:project scope is required for the new github board).
verbose	Logical. Whether to return output in the console too.

**Value**

A Weekly to copy-paste in a Wiki and a tibble

**Examples**

```
## Not run:
user <- "ThinkR-open"
repo <- "example-weekly"
date_min <- "2022-06-30"
date_max <- "2022-06-30"
board_url_old <- glue::glue("/repos/{user}/{repo}/projects")
board_url_new <- "https://github.com/orgs/ThinkR-open/projects/4/"

# old board
weekly_old <- gh_create_weekly_old_and_new_boards(
  date_min = date_min,
  date_max = date_max,
  user = user,
  repo = repo,
  board_url = board_url_old,
  new_board = FALSE,
  verbose = FALSE
)
cat(weekly_old$weekly_info)

# clipr::write_clip(weekly_old$weekly_info)

# new board
weekly_new <- gh_create_weekly_old_and_new_boards(
  date_min = date_min,
  date_max = date_max,
  user = user,
  repo = repo,
  board_url = board_url_new,
  github_token = Sys.getenv("GITHUB_PAT"),
  new_board = TRUE,
  verbose = FALSE
)
cat(weekly_new$weekly_info)
# clipr::write_clip(weekly_new$weekly_info)
```

```
## End(Not run)
```

---

gl\_add\_template\_issue *Create the template for standard issue on gitlab*

---

## Description

Create the template for standard issue on gitlab

## Usage

```
gl_add_template_issue(  
  project_path = ".",  
  language = c("fr", "en"),  
  type = c("full", "light")  
)
```

## Arguments

project_path	Path to the project
language	Language. Should be one of 'fr' or 'en'
type	Verbosity of comments in the template. Should be 'full' or 'light'.

## Value

Side effect: create the template

## Examples

```
# Try with a temp project  
## Create temp dir and file  
project_path <- tempfile("myproject")  
dir.create(project_path)  
  
## Add the template in full french  
gl_add_template_issue(project_path, language = "fr", type = "full")  
  
## Add the template in full english  
gl_add_template_issue(project_path, language = "en", type = "full")  
  
## Add the template in light french  
gl_add_template_issue(project_path, language = "fr", type = "light")  
  
## Add the template in light english  
gl_add_template_issue(project_path, language = "en", type = "light")  
  
# Suppress tmp folder  
unlink(project_path, recursive = TRUE)
```

---

gl_create_weekly	<i>Create a Weekly/Daily summary of what happened in your GitLab Board</i>
------------------	--

---

## Description

Create a Weekly/Daily summary of what happened in your GitLab Board

## Usage

```
gl_create_weekly(
  project_id,
  date_min = Sys.Date() - 7,
  date_max = Sys.Date(),
  language = c("fr", "en"),
  gitlab_url = Sys.getenv("GITLAB_URL", unset = "https://gitlab.com"),
  private_token = Sys.getenv("GITLAB_TOKEN"),
  verbose = FALSE,
  regex_done = "close|closed|done",
  regex_validation = "a valider|validation",
  regex_blocked = "blocked|bloque|bloqu\\\\\\u00e9",
  regex_inprogress =
    "in progress|en cours|review|revision|r\\\\\\u00e9vision|r\\\\\\u00e9-validation",
  regex_ready = "ready|pret|pr\\\\\\u00eat",
  daily = FALSE,
  date_daily = Sys.Date(),
  max_page_opened = 10,
  max_page_closed = 5
)

gl_create_daily(...)
```

## Arguments

project_id	Id of the GitLab project
date_min	Minimal date to search from. The oldest date.
date_max	Maximal date to search to. The more recent date.
language	Character. Language: 'fr' ou 'en'.
gitlab_url	URL of the GitLab Forge
private_token	Your private GitLab instance token allowed to use the "api".
verbose	Logical. Whether to show the outputs in the console.
regex_done	Regular expression to detect labels for issues done by developers (and potentially awaiting validation by clients)
regex_validation	Regular expression to detect labels for issues waiting for validation



regex_blocked	Regular expression to detect labels for issues blocked (and potentially awaiting information by clients)
regex_inprogress	Regular expression to detect labels for issues in progress (and potentially awaiting review by lead dev.)
regex_ready	Regular expression to detect labels for issues ready (and potentially reorder by lead dev.)
daily	Logical. Create the daily issues summary for GitLab
date_daily	Date to search from.
max_page_opened	a numeric. The maximum number of pages to retrieve for open issues. Defaults to 10.
max_page_closed	a numeric. The maximum number of pages to retrieve for close issues. Defaults to 5.
...	Additional arguments

**Value**

A Weekly to copy-paste in a Wiki and a tibble

---

gl\_get\_milestones\_progress

*Visualise the progress of the milestones for GitLab*

---

**Description**

Visualise the progress of the milestones for GitLab

**Usage**

```
gl_get_milestones_progress(
  project_id,
  language = c("fr", "en"),
  gitlab_url = Sys.getenv("GITLAB_URL", unset = "https://gitlab.com"),
  private_token = Sys.getenv("GITLAB_TOKEN"),
  color = "#f15522"
)
```

**Arguments**

project_id	Id of the GitLab project
language	Character. Language: 'fr' ou 'en'.
gitlab_url	URL of the GitLab Forge
private_token	Your private GitLab instance token allowed to use the "api".
color	by default, warning color

**Value**

A ggplot

**Examples**

```
## Not run:
gl_get_milestones_progress(
  project_id = "<get_your_id_project>",
  private_token = Sys.getenv("GITLAB_TOKEN")
)

## End(Not run)
```

---

graphql_to_tibble	<i>Query GraphQL Github API</i>
-------------------	---------------------------------

---

**Description**

This function helps you retrieving the status of a project board within the organization

**Usage**

```
graphql_to_tibble(board_url, github_token = Sys.getenv("GITHUB_PAT"))
```

**Arguments**

board_url	url of the github project board
github_token	access token to the graphql api

**Value**

a tibble

**Examples**

```
## Not run:

# Example with board hosted in an organization github account
board_url_organization <- "https://github.com/orgs/ThinkR-open/projects/4/"
github_token <- Sys.getenv("GITHUB_PAT")

graphql_to_tibble(
  board_url = board_url_organization,
  github_token = github_token
)

# Example with board hosted in a user github account
board_url_user <- "https://github.com/users/the-thinkr/projects/1"
```

```

github_token <- Sys.getenv("GITHUB_PAT")

graphql_to_tibble(
  board_url = board_url_user,
  github_token = github_token
)

## End(Not run)

```

---

html_to_odt	<i>Convert html to odt with template</i>
-------------	--

---

### Description

Convert html to odt with template

### Usage

```
html_to_odt(input_html = "_main.html", output_odt = "_main.odt")
```

### Arguments

input_html	Path to html file to convert
output_odt	Path to odt to create

### Examples

```

## Not run:
html_to_odt(input_html = "_main.html", output_odt = "_main.odt")

## End(Not run)

```

---

init_project_with_all	<i>Init a new project with everything in one command</i>
-----------------------	--

---

### Description

Init a new project with everything in one command

### Usage

```

init_project_with_all(
  project_name,
  project_gitlab_id = NULL,
  config_path,
  project_path = tempfile("clone"),
  ...
)

```

**Arguments**

project_name	The name of the project to be created
project_gitlab_id	GitLab ID of the project if it already exist on your Forge
config_path	The path to the yaml configuration file to use with your options to modify the default one. See details.
project_path	The path to the project if your want to keep it locally. Default to temporary directory.
...	Any parameter existing in the configuration file that you would like to use to override your config files

**Details**

By default, the project is a R package created on <https://gitlab.com/about/> in your personal repository. Use your own configuration file to amend the default one. The configuration file is a yaml file with all possible options. You do not have to specify all options as it will be combined with our default ones. Open the default one to see what is in it: `file.edit(system.file("config_default_thinkr_gitlab.yaml", package = "lozen"))`

**Examples**

```
if (interactive()) {
  # Default on GitLab.com
  init_project_with_all(project_name = "newprojectpkg")
  # Change default options with your own config file
  init_project_with_all(project_name = "newprojectpkg", config_path = "<my-config-path>")
  # Add any extra option to override values of your config file once
  init_project_with_all(project_name = "newprojectpkg", config_path = "<my-config-path>", gitlab_namespace_id = "6")
}
```

---

modify\_autoclose\_and\_coverage

*modify\_autoclose\_and\_coverage on GitLab*

---

**Description**

modify\_autoclose\_and\_coverage on GitLab

**Usage**

```
modify_autoclose_and_coverage(
  project_id,
  autoclose = FALSE,
  build_coverage_regex = "Coverage: \\d+\\.\\d+"
)
```

**Arguments**

project_id	project_id
autoclose	Logical. Whether to autoclose issues when merged to main
build_coverage_regex	Character. regex used to retrieve code coverage in Ci logs.

**Value**

Side Effect on GitLab

**Examples**

```
## Not run:
modify_autoclose_and_coverage(
  project_id = project_id
)

## End(Not run)
```

---

move\_issues\_from\_gitlab\_to\_github

*Move Issues from GitLab to GitHub*

---

**Description**

Move Issues from GitLab to GitHub

**Usage**

```
move_issues_from_gitlab_to_github(
  gitlab_url = "https://gitlab.com/",
  github_url = "https://github.com/",
  gitlab_group,
  gitlab_repo,
  gitlab_project_id,
  github_owner,
  github_repo,
  gitlab_private_token = Sys.getenv("GITLAB_TOKEN"),
  github_token = NULL,
  sleep_time = 60,
  issue_start_over = 1,
  max_page = 10,
  force = FALSE
)
```

**Arguments**

<code>gitlab_url</code>	GitLab instance URL
<code>github_url</code>	GitHub URL, default to <github.com>
<code>gitlab_group</code>	group name. Can be sub-group with "group/sub-group" format
<code>gitlab_repo</code>	Repo name on GitLab
<code>gitlab_project_id</code>	Repo ID on GitLab
<code>github_owner</code>	Owner of the project on GitHub
<code>github_repo</code>	Repo name of the project on GitHub. It should already exists.
<code>gitlab_private_token</code>	GitLab Token with API permissions
<code>github_token</code>	GitHub token. If NULL, will search for GITHUB_PAT in your env. variables
<code>sleep_time</code>	Time in second to wait between each set of 5 issues. This is to reduce risk of being banned by GitHub. If you have time, Set to 60s by default.
<code>issue_start_over</code>	For debug. Number of the issue in GitLab to start over with in case of temporary ban by GitHub.
<code>max_page</code>	Maximum number of pages of issues to retrieve from GitLab.
<code>force</code>	Whether to force writing issue to GitHub, even if issue numbers mismatch with GitLab.

**Value**

Used for side effects. Issues are dowloaded and copied to GitHub

**Examples**

```
## Not run:
move_issues_from_gitlab_to_github(
  gitlab_url = "https://gitlab.com/",
  github_url = "https://github.com/",
  gitlab_group = "thinkr-open",
  gitlab_repo = "example-weekly",
  gitlab_project_id = 37585948,
  github_owner = "ThinkR-open",
  github_repo = "example.mirror.issues.gl.gh",
  gitlab_private_token = Sys.getenv("GITLAB_TOKEN"),
  github_token = Sys.getenv("GITHUB_PAT"),
  sleep_time = 60,
  issue_start_over = 1,
  max_page = 10,
  force = FALSE
)

## End(Not run)
```

---

paged_template	<i>Render pagedown html_paged</i>
----------------	-----------------------------------

---

## Description

Render pagedown html\_paged

## Usage

```
paged_template(
  ...,
  css = c("default-fonts", "default-page", "default"),
  theme = NULL,
  template = pkg_resource("html", "paged.html"),
  cs1 = NULL,
  front_cover = NULL,
  back_cover = NULL
)
```

## Arguments

...	Arguments passed to bookdown: <a href="#">:html_document2</a> .
css	A character vector of CSS and Sass file paths. If a path does not contain the <code>' .css'</code> , <code>' .sass'</code> , or <code>' .scss'</code> extension, it is assumed to be a built-in CSS file. For example, <code>default-fonts</code> means the <code>filepagedown:::pkg_resource('css', 'default-fonts.css')</code> . To see all built-in CSS files, run <code>pagedown:::list_css()</code> .
theme	The Bootstrap theme. By default, Bootstrap is not used.
template	The path to the Pandoc template to convert Markdown to HTML.
cs1	The path of the Citation Style Language (CSL) file used to format citations and references (see the <a href="#">Pandoc documentation</a> ).
front_cover, back_cover	Paths or urls to image files to be used as front or back covers. Theses images are available through CSS variables (see Details).

## Value

A pagedown template

---

project_options	<i>Define project status for a user</i>
-----------------	---

---

**Description**

Define project status for a user

**Usage**

```
project_options(project_id, level = "watch")
```

**Arguments**

project_id	project_id
level	The notification levels among disabled, participating, watch, global, mention, custom

**Value**

Side Effect on GitLab, allow reception of notifications.

**Examples**

```
## Not run:
project_options(
  project_id,
  level = "watch"
)

## End(Not run)
```

---

protect_branches	<i>Protect 'main' and 'production' from push on server</i>
------------------	--

---

**Description**

Protect 'main' and 'production' from push on server

**Usage**

```
protect_branches(project_id, unprotect = FALSE)
```

**Arguments**

project_id	project_id
unprotect	Logical. Whether to unprotect branches or not.



**Value**

Side effect: Branches protected from push on server.

**Examples**

```
## Not run:
protect_branches(
  project_id = project_id
)
# unprotect if wanted
# protect_branches(project_id, unprotect = TRUE)

## End(Not run)
```

---

push_all_to_branch	<i>Push everything on a branch</i>
--------------------	------------------------------------

---

**Description**

Push everything on a branch

**Usage**

```
push_all_to_branch(
  project_path,
  branch = "main",
  main_branch = "main",
  message = "Init repo"
)
```

**Arguments**

project_path	project_path
branch	branch name. main by default
main_branch	main branch name. main by default
message	commit message

**Value**

Side effect: commit output

**Examples**

```
## Not run:
push_all_to_branch(
  project_path = project_path
)

## End(Not run)
```

---

push\_main

*Push main to server*

---

### Description

Push main to server

### Usage

```
push_main(  
    project_path,  
    branch = "main",  
    main_branch = "main",  
    message = "Init repo"  
)
```

### Arguments

project_path	project_path
branch	branch name. main by default
main_branch	main branch name. main by default
message	commit message

---

push\_master

*Push master to server*

---

### Description

Push master to server

### Usage

```
push_master(  
    project_path,  
    branch = "master",  
    main_branch = "master",  
    message = "Init repo"  
)
```

### Arguments

project_path	project_path
branch	branch name. main by default
main_branch	main branch name. main by default
message	commit message

---

read_ci	<i>read_ci</i>
---------	----------------

---

**Description**

Import yaml CI files as list

**Usage**

```
read_ci(path = "./.gitlab-ci.yml")
```

**Arguments**

path                      Path to the yaml file

**Value**

list a list of all CI parameters

**Examples**

```
yaml_path <- system.file("yaml", ".gitlab-ci-shiny.yaml", package = "lozen")
ci_list <- read_ci(path = yaml_path)
```

---

render_book	<i>Render a book with lozen formats</i>
-------------	---

---

**Description**

Render a book with lozen formats

**Usage**

```
render_book(
  rmd_to_render = "index.Rmd",
  output_format = c("lozen::bs4_book_template", "lozen::paged_template"),
  output_dir = "_book",
  ...
)
```

**Arguments**

rmd\_to\_render    rmd file to render  
 output\_format   format to be user to render the book  
 output\_dir       output directory  
 ...               extra params to be used in bookdown::render\_book

**Value**

a rendered book

**Examples**

```
## Not run:
render_book("index.Rmd", output_format = "lozen::bs4_book_template")
render_book("index.Rmd", output_format = "lozen::paged_template")

## End(Not run)
```

---

use_dev_history	<i>Une a dev histroy file</i>
-----------------	-------------------------------

---

**Description**

Une a dev histroy file

Une a dev histroy file

**Usage**

```
use_dev_history(path = ".", type = c("package", "renv"))
use_dev_history(path = ".", type = c("package", "renv"))
```

**Arguments**

path	Path to project to add dev_history in
type	Type of dev_history. Multiple types possible among "package", "book", "renv".

**Examples**

```
withr::with_tempdir({
  use_dev_history(path = ".", type = c("package", "book", "renv"))
})
withr::with_tempdir({
  use_dev_history(path = ".", type = c("package", "book", "renv"))
})
```

---

use_gitlab_ci	<i>Set gitlab continuous integration</i>
---------------	--

---

## Description

Set gitlab continuous integration

## Usage

```
use_gitlab_ci(
  image = "rocker/verse",
  repo_name = "https://packagemanager.rstudio.com/all/__linux__/focal/latest",
  project_path = ".",
  type = "check-coverage-pkgdown",
  bookdown_output_format = c("lozen::paged_template", "lozen::bs4_book_template"),
  overwrite = TRUE
)
```

## Arguments

image	Docker image used as basis. See <a href="https://github.com/rocker-org/rocker">https://github.com/rocker-org/rocker</a>
repo_name	REPO_NAME environment variable for R CRAN mirror used
project_path	Path of the project to add CI in.
type	type of the CI template to use
bookdown_output_format	If type="bookdown" it corresponds to the function used to output the bookdown
overwrite	whether to overwrite existing GitLab CI yml file

## Details

See [use\\_gitlab\\_ci](#)

## Examples

```
withr::with_tempdir({
  use_gitlab_ci(image = "r-base")
})

withr::with_tempdir({
  use_gitlab_ci(
    image = "rocker/verse",
    repo_name = "https://packagemanager.rstudio.com/all/__linux__/focal/latest",
    type = "check-coverage-pkgdown"
  )
})
```

---

```
use_gitlab_ci_deploy_connect
```

*Add a job to deploy a something on Connect*

---

## Description

This function creates a file `.gitlab-ci.yml` that will deploy something on Connect using a `lozen::deploy_connect_*` functions.

## Usage

```
use_gitlab_ci_deploy_connect(
  deploy_function = c("deploy_connect_shiny", "deploy_connect_pkgdown",
    "deploy_connect_bookdown"),
  stage_name = "deploy_connect",
  image = "rocker/verse:latest",
  dir = ".",
  append = TRUE,
  file_name = ".gitlab-ci.yml",
  ...
)
```

```
use_gitlab_ci_deploy_connect_bookdown(...)
```

```
use_gitlab_ci_deploy_connect_pkgdown(...)
```

```
use_gitlab_ci_deploy_connect_shiny(...)
```

## Arguments

<code>deploy_function</code>	string caracter name of the <code>deploy_connect_*</code> functions to use.
<code>stage_name</code>	name of the CI stage (need to be unique in the <code>.gitlab-ci.yml</code> file)
<code>image</code>	Docker image to use
<code>dir</code>	Directory to deploy
<code>append</code>	append the file <code>.gitlab-ci.yml</code> if it already exists
<code>file_name</code>	Name of the yaml file
<code>...</code>	param to pass to <code>deploy_function</code>

## Details

Before using it, please follow these steps :

- Ask the Mission Lead Dev for their deployment token on Connect, this is the one you will need to use.

- Add the environment variables in the private variable hidden in the GitLab repository: Settings > CI/CD > Variables > Expand > Add variable /\ if you check "Protected" when you create the variable, then it will be active only for protected branches
  - Add CONNECT\_USER with username.
  - Add CONNECT\_TOKEN with the token.

If you want to deploy an app Be sure to have an app.R to deploy at the root of the folder, (if needed: use `golem::add_rstudioconnect_file()`)

NB: The environment variable CONNECT\_URL is already defined generically on forge (only for group "thinkr").

## Value

used for side effects

## Examples

```
#
use_gitlab_ci_deploy_connect_bookdown()
# Exemple avec pkgdown
deployed_pkgdown <- tempfile(pattern = "pkgdown")
dir.create(deployed_pkgdown)
# rstudioapi::filesPaneNavigate(deployed_pkgdown)
withr::with_dir(deployed_pkgdown, {
  use_gitlab_ci(type = "check-coverage-pkgdown")
  use_gitlab_ci_deploy_connect_pkgdown()
})
deployed_shiny <- tempfile(pattern = "shiny")
dir.create(deployed_shiny)
# rstudioapi::filesPaneNavigate(deployed_shiny)
# Exemple avec shiny
withr::with_dir(deployed_shiny, {
  use_gitlab_ci(type = "check-coverage-pkgdown")
  use_gitlab_ci_deploy_connect_shiny()
})
```

---

visualise\_commits

*Visualise the nature of the developments (conventional commits)*

---

## Description

Visualise the nature of the developments (conventional commits)

## Usage

```
visualise_commits(
  project_id,
  gitlab_url = Sys.getenv("GITLAB_URL", unset = "https://gitlab.com"),
```

```

private_token = Sys.getenv("GITLAB_TOKEN"),
path,
ref = NULL,
date_min = Sys.Date() - 7,
date_max = Sys.Date(),
language = c("fr", "en"),
conv_tags = c("feat", "fix", "doc", "test", "ci", "refactor", "style", "chore"),
color = "#15b7d6"
)

```

### Arguments

project_id	Id of the GitLab project
gitlab_url	URL of the GitLab Forge
private_token	Your private GitLab instance token allowed to use the "api".
path	Path of the project if already pulled locally
ref	Branch name (or commit sha) from which to explore commits. By default NULL, it use the default branch
date_min	Minimal date to search from. The oldest date.
date_max	Maximal date to search to. The more recent date.
language	Character. Language: 'fr' ou 'en'.
conv_tags	Conventional commits tags
color	blue color for a graph

### Value

A ggplot

### Examples

```

## Not run:

visualise_commits(
  project_id = "<get_your_id_project>",
  gitlab_url = Sys.getenv("GITLAB_URL", unset = "https://gitlab.com"),
  date_min = "2022-09-22",
  date_max = "2022-09-29",
  private_token = Sys.getenv("GITLAB_TOKEN")
)

# Or on project already cloned
tempdir <-
  clone_locally(full_url = "https://gitlab.com/my_name/my_repo", open = FALSE)

visualise_commits(
  path = tempdir,
  date_min = "2022-09-22",
  date_max = "2022-09-29"
)

```



```
)

## End(Not run)
```

---

with\_gitlab\_project     *Test gitlab CI workflows*

---

## Description

This functions allows its users to launch an expression to initiate a given gitlab ci workflow within a dummy project on a #' gitlab instance.

## Usage

```
with_gitlab_project(
  gitlab_url = Sys.getenv("GITLAB_URL", unset = "https://gitlab.com"),
  namespace_id,
  private_token = Sys.getenv("GITLAB_TOKEN"),
  connect_url = Sys.getenv("CONNECT_URL"),
  connect_api_token = Sys.getenv("CONNECT_TOKEN"),
  connect_user = Sys.getenv("CONNECT_USER"),
  connect_name = Sys.getenv("CONNECT_NAME", unset = "connect"),
  project_name = "lozen.test.project",
  branch_focus_for_ci = "main",
  exp
)
```

## Arguments

gitlab_url	A character string. The url of the gitlab instance.
namespace_id	An integer. The id of the gitlab group to create the project in.
private_token	A character string. The token to gain access to the gitlab api.
connect_url	A character string. The URL to the Connect server.
connect_api_token	A character string. The token to gain access to the Connect API. The token must be granted admin access.
connect_user	A character string. User name on Connect.
connect_name	A character string. Name of the Connect server (default "connect")
project_name	A character string. The name of the project to be created.
branch_focus_for_ci	Name of the branch to be targeted for the CI pipeline.
exp	A valid R expression initializing a project and a gitlab ci workflow.

## Value

A data.frame containing information about jobs of the gitlab project.

**Examples**

```
## Not run:
output <- with_gitlab_project(
  gitlab_url = Sys.getenv("GITLAB_URL", unset = "https://gitlab.com"),
  namespace_id = NULL,
  private_token = Sys.getenv("GITLAB_TOKEN"),
  project_name = "bookdown.test.project",
  exp = {
    lozen::create_r_project(
      project_path = getwd(),
      type = "book",
      name_licence = "Bobo",
      type_licence = usethis::use_mit_license
    )
    lozen::use_gitlab_ci(type = "bookdown", bookdown_output_format = "lozen::paged_template")
  }
)
output

## End(Not run)
```

# Index

add\_board, 3  
add\_board\_github, 4  
add\_git\_templates, 5  
add\_issue\_clients, 5  
add\_issue\_clients\_github, 6  
add\_issue\_dev, 7  
add\_issue\_dev\_github, 8  
add\_issue\_kickoff, 8  
add\_issue\_kickoff\_github, 9  
add\_kit\_package, 10  
add\_kit\_project, 11  
add\_labels, 11  
add\_wikis, 12  
add\_wikis\_github, 13  
amend\_yaml, 14  
  
bs4\_book\_template, 14  
bslib::bs\_theme(), 15  
build\_pkgdown\_with\_reports, 15  
  
check\_if\_yaml\_exists, 16  
clean\_image, 17  
clone\_locally, 17  
combine\_ci, 18  
create\_book\_project, 19  
create\_deploy\_ci\_stage, 20  
create\_group\_project, 20  
create\_group\_project\_github, 21  
create\_issue\_content\_clients, 22  
create\_issue\_content\_clients\_github, 23  
create\_issue\_content\_dev, 23  
create\_issue\_content\_dev\_github, 24  
create\_issue\_content\_kickoff, 25  
create\_production, 25  
create\_r\_project, 26  
  
deploy\_connect\_bookdown, 27  
deploy\_connect\_pkgdown, 29  
deploy\_connect\_shiny, 31  
  
fetch\_connect, 33  
  
gh\_add\_template\_issue, 33  
gh\_create\_weekly, 34  
gh\_create\_weekly\_new\_projects\_board, 35  
gh\_create\_weekly\_old\_and\_new\_boards, 37  
gl\_add\_template\_issue, 39  
gl\_create\_daily(gl\_create\_weekly), 40  
gl\_create\_weekly, 40  
gl\_get\_milestones\_progress, 41  
golem::add\_rstudioconnect\_file(), 55  
graphql\_to\_tibble, 42  
  
html\_document2, 47  
html\_to\_odt, 43  
  
init\_project\_with\_all, 43  
  
modify\_autoclose\_and\_coverage, 44  
move\_issues\_from\_gitlab\_to\_github, 45  
  
paged\_template, 47  
project\_options, 48  
protect\_branches, 48  
push\_all\_to\_branch, 49  
push\_main, 50  
push\_master, 50  
  
read\_ci, 51  
render\_book, 51  
rmarkdown::html\_document(), 15  
  
use\_dev\_history, 52  
use\_gitlab\_ci, 53, 53  
use\_gitlab\_ci\_deploy\_connect, 54  
use\_gitlab\_ci\_deploy\_connect\_bookdown  
(use\_gitlab\_ci\_deploy\_connect), 54

`use_gitlab_ci_deploy_connect_pkgdown`  
    (`use_gitlab_ci_deploy_connect`),  
    [54](#)

`use_gitlab_ci_deploy_connect_shiny`  
    (`use_gitlab_ci_deploy_connect`),  
    [54](#)

`visualise_commits`, [55](#)

`with_gitlab_project`, [57](#)