

Cookbook for CICD with R and GitHub Actions

Yann Say

2023-02-22

Contents

1	Intro	5
2	R CMD check	7
2.1	R-CMD-check	7
2.2	Prerequisite	7
2.3	Steps	7
3	Test coverage	9
3.1	test-coverage	9
3.2	Prerequisite	9
3.3	Steps	9
4	Style	11
4.1	styler	11
4.2	Prerequisite	11
4.3	Steps	11
4.4	Possible errors	12
5	Bookdown	15
5.1	Bookdown	15
5.2	Prerequisite	15
5.3	Steps	15

6	Shiny deploy	17
6.1	shiny-deploy	17
6.2	Prerequisite	17
6.3	Steps	17

Chapter 1

Intro

This is a small collections of step-by-step on how to set-up of r-lib GitHub Actions for CICD with R programming. It mainly use the package `usethis`(version 2.1.6). More details of the actions can be found [here](#). When I thought necessary I have added a few additional steps, e.g. when using other services from GitHub or Codecov which I thought was missing when I was trying to learn about GitHub Actions.

The book was built with **bookdown**.

Chapter 2

R CMD check

2.1 R-CMD-check

Quick how-to to set a github action for a R CMD check each time there is merge into the **main** (or **master**) branch using **r-lib** check-release.yaml or check-standard.yaml.

2.2 Prerequisite

None in particular but could be good to have some tests already.

2.3 Steps

- Create your package.
- Try to test your package `devtools::test()` or `devtools::check()`
 - While this step is not really necessary, it is to make sure your tests runs once and any further problems do not come from the tests.
- Add the check-release.yaml `usethis::use_github_action("check-release")` or check-standard.yaml `usethis::use_github_action("check-standard")`
 - check-release.yaml will run R CMD check on Ubuntu and current R version

- `check-standard.yaml` will run R CMD check on 3 OS: mac and Windows with the current R version, Ubuntu with the current, development and previous version of R. This what you would want for CRAN.
- Add the badge with `usethis::use_github_actions_badge("check-release")` or `usethis::use_github_actions_badge("check-standard")`
- Push!

Chapter 3

Test coverage

3.1 test-coverage

Quick how-to to set a GitHub action to get code coverage and the badge from Codecov each time there is merge into the **main** (or **master**) branch using `r-lib` `test-coverage.yaml`.

3.2 Prerequisite

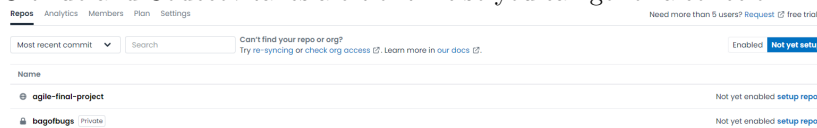
Have a Codecov account link with your repository.

None in particular but could be good to have some tests as to get a coverage of something.

3.3 Steps

- Create your package.
- Link your local to the remote if it has not be done yet (make sure your project is linked to a Github repository).
- Make sure Codecov has synced with your repository. The repository should appear in the *Not yet setup* if this is the first time. The syncing between GitHub and Codecov takes a bit of time so you can go for a coffee or

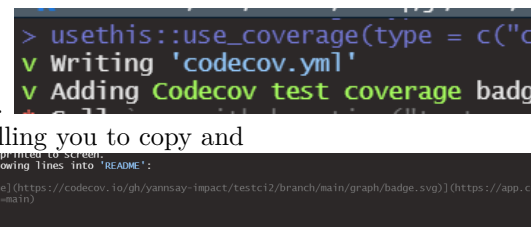
lunch.



- Try to test your package `covr::package_coverage()`.
 - While this step is not really necessary, it is to make sure your tests runs once and any further problems do not come from the tests.
- Add the test-coverage.yaml `usethis::use_github_action("test-coverage")`
- Add the badge with `usethis::use_coverage("codecov")`

- This step should add a badge in your readme file.
- Or should give you a message in the console telling you to copy and paste some lines in the README.
- If nothing happen, you can add the line directly following the syntax or remove the `codecov.yaml` from the project folder and run `usethis::use_coverage("codecov")` again.

- Push!
- The GitHub action will push the report to Codecov, you can see the report by clicking on the badge.
 - Github and Codecov needs to sync for the percentage to appear



```

> usethis::use_coverage(type = c("codecov"))
v Writing 'codecov.yaml'
v Adding Codecov test coverage badge
[1] "https://app.codecov.io/gh/yannsay-impact/testc12/branch/main/graph/badge.svg"

```

Badge link can only be printed to screen.
 Copy and paste the following lines into 'README':

```

<!-- badges: start -->
[[codecov test coverage]](https://codecov.io/gh/yannsay-impact/testc12/branch/main/graph/badge.svg)
<!-- badges: end -->

```

[copied to clipboard]

Chapter 4

Style

4.1 styler

Quick how-to to set a github action to style each time there is push or a pull request with files that includes `.[rR]`, `.[rR]md`, `.[rR]markdown`, `.[rR]nw`. into the **main** using `r-lib style.yaml`.

4.2 Prerequisite

None in particular.

4.3 Steps

- Create your package.
- Try to test your package `styler::style_pkg()`
 - While this step is not really necessary, it is to make sure your tests runs once and any further problems do not come from the tests.
- Add the style.yaml `usethis::use_github_action("style")`
 - You can change the `styler::style_pkg` to `styler::style_file` or `styler::style_dir`.

```
- name: Style
  run: styler::style_pkg(filetype = c(".R", ".Rmd", ".Rmarkdown", ".Rnw"))
  shell: Rscript {0}
```

– More info and customisation on the package page.

- Push!

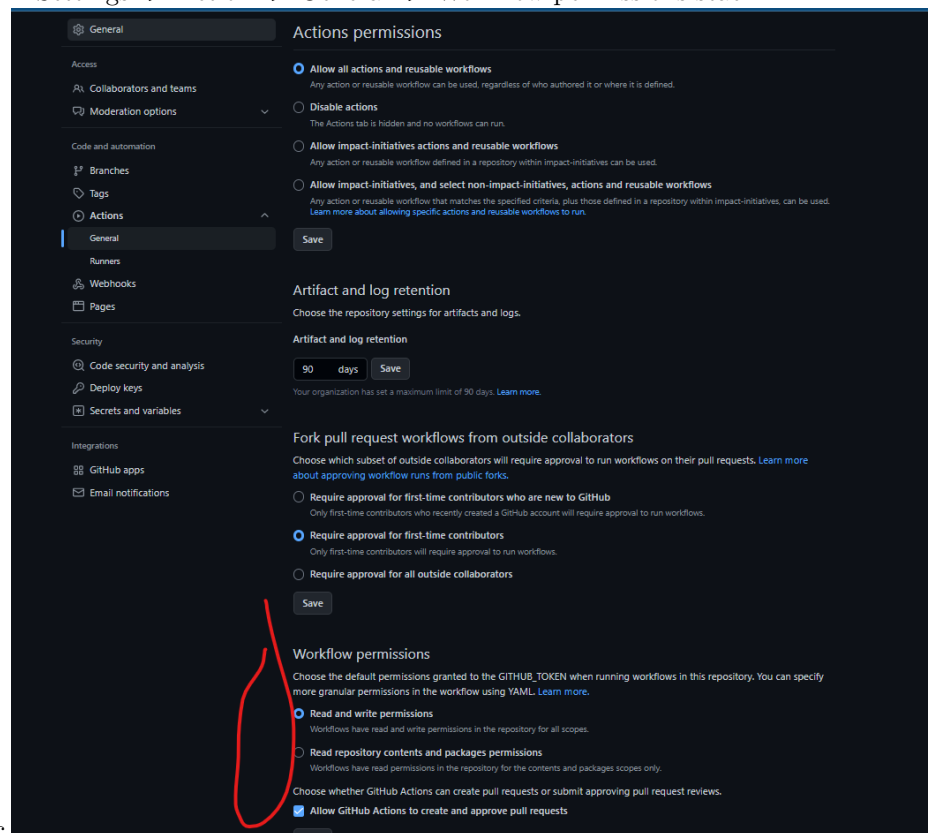
4.4 Possible errors

4.4.1 Error 403

It is possible that once you set your action, it will not run and you will get an error 403. Error 403 is usually link to a forbidden access error. It can look as below. It can happen if you changed ownership from a repository.

```
remote: Permission to /ACCOUNT-NAME/REPO-NAME.git denied to github-actions[bot].
fatal: unable to access 'https://github.com/ACCOUNT-NAME/REPO-NAME/': The requested URL
```

To remediate to this, you need to change the workflow permission in your repository settings: - Settings -> Action -> General -> Workflow permissions stack-



overflow topic

4.4.2 Protected branch

It is possible that the action fails if the branch is protected and needs review.
It will solve once the review is approved.

```
remote: error: GH006: Protected branch update failed for refs/heads/master.  
remote: error: At least 1 approving review is required by reviewers with write access.
```


Chapter 5

Bookdown

5.1 Bookdown

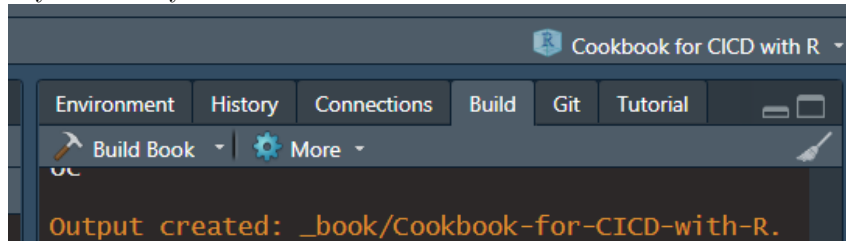
Quick how to set a github action for a bookdown deployment onto the **gh-pages** branch each time there is merge into the **main** (or **master**) branch using **r-lib** bookdown.yaml

5.2 Prerequisite

- The action will require a *renv* environment.
- Book to build (you can also use the faithful geyser example).

5.3 Steps

- Create your book using **renv** (The yaml use assume **renv** is used).
- Try to build your book with the *Build Book* button on the *Build tab*.



- While this step is not really necessary, it is to make sure your tests runs once and any further problems do not come from the tests.

- Add the bookdown.yaml `usethis::use_github_action("bookdown")`
- Push!
 - There is no need to have the **gh-pages** set up beforehand. The action will set it up. If the the **gh-pages** branch is not set up, it will take some time to get it set up.
 - There is no need to have the book build neither.

Chapter 6

Shiny deploy

6.1 shiny-deploy

Quick how to set a github action for a shiny deployment each time there is merge into the **main** (or **master**) branch using **r-lib** shiny-deploy.yaml

6.2 Prerequisite

Make sure you have the rsconnect information.

- <https://shiny.rstudio.com/articles/shinyapps.html>
- The action will require a *renv* environment.
- App to deploy (you can also use the faithful geyser example).

6.3 Steps

- Create your app using **renv** (The yaml use assume **renv** is used).
 - *It is probably also a best practice to use renv.*
- Try to deploy your app (either with the publish/redeploy button) or `rsconnect::deployApp()`
 - While this step is not really necessary, it is to make sure your app runs once and any further problems do not come from the app itself.
- Add a description `usethis::use_description(check_name = F)`.

- `check_name = F` to avoid checking name valid for CRAN.
- Add the shiny-deploy.yaml `usethis::use_github_action("shiny-deploy.yaml")`
- Edit the shiny-deploy.yaml, especially the following part.
 - You can either fill in for the APPNAME, ACCOUNT, SERVER.

```
- name: Authorize and deploy app
  env:
    # Provide your app name, account name, and server to be deployed below
    APPNAME: your-app-name
    ACCOUNT: your-account-name
    SERVER: shinyapps.io # server to deploy
  run: |
    rsconnect::setAccountInfo("${{ secrets.RSCONNECT_USER }}", "${{ secrets.RSC
    rsconnect::deployApp(appName = "${{ env.APPNAME }}", account = "${{ env.ACCO
  shell: Rscript {0}
```

- Or you can remove the `env` block and change the `rsconnect::deployApp` call.

```
- name: Authorize and deploy app
  run: |
    rsconnect::setAccountInfo("${{ secrets.RSCONNECT_USER }}", "${{ secrets.RSC
    rsconnect::deployApp()
  shell: Rscript {0}
```

- Add 3 secrets in the repository to store you account, token and secret (see prerequisite). They should have those names
 - `RSCONNECT_USER`, `RSCONNECT_TOKEN`, and secret `RSCONNECT_SECRET`
 - You can follow these guides if needed
- Link the shiny app folder to the github repo (add the remote and origin) (if you have not done it already).
- Push!