Cookbook for CICD with R and GitHub Actions

Yann Say

2022-11-23

Contents

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

Shiny deploy

3.1 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

3.2 Prerequisite

Make sure you have the rsconnect information.

- https://shiny.rstudio.com/articles/shinyapps.html

3.3 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.

```yaml
- 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.RSCO
    rsconnect::deployApp(appName = "${{ env.APPNAME }}", account = "${{ env.ACCO
  shell: Rscript {0}
```

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

```yaml
- name: Authorize and deploy app
  run: |
    rsconnect::setAccountInfo("${{ secrets.RSCONNECT_USER }}", "${{ secrets.RSCO
    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!