PharmaSUG China 2022 - Paper AT-147

Using CI/CD to Better Manage R Projects

Dan Li, Merck

ABSTRACT

People are using R instead of SAS more and more these days and started to create their R packages. When using a shared repository to collaborate, there are some inconveniences that drains time and energy of the developers, such as too many branches or merge conflicts.

CI/CD (continuous integration and continuous delivery) is a useful tool to simplify the difficulty of code integration and the hassle of repetitively manual building and copying the developed R package to testing environment.

Additionally, CI/CD enforces some good practices such as thinking about testing up front and standardizing commits. They will also enhance development experience as a team and the quality of outcome.

INTRODUCTION

As our company gradually shifting its language focus from SAS to R, colleagues from various apartments are starting to write their own R packages/shiny apps, etc. When writing R code, people also tend to use git as their code version control system. Thus, tools like gitlab, github or bitbucket are usually adopted. To enhance the quality of these tools, developers also review and test their code. However, they rarely used proper way and process that these version control systems suggest. Instead, they use time consuming and inconvenient way to record how they test their tool, such as writing MS word files and emails.

This paper proposed using a lightweight CI/CD tool called drone to automate code style check and running test cases before merging to main branch. Except for these two scenarios, more conveniences brought by CI/CD are also briefly introduced.

STATUS OF MANAGING R PROJECTS

Status of managing R projects in our company can be divided into two categories in terms of using git-related tools (such as internally hosted gitlab or bitbucket) or not.

The first category is doing without git-related tools. Companies may have commercialized platform software that enabled them to version control. Under some cases, development team even don’t do version control. But there will be a lot of obstacles not using git to manage the workflow and process when a team develops a project.

The other category is employing git-related tools to manage an R project. In this way, people can cooperate more cohesively without heavy paperwork. For example, new communications of a certain issue will be brought to participants in real time in issue page. Discussions can turn into wiki so that knowledge can be passed to all members of the team. Comments on a certain line of code can be added during code review and related personnel will automatically be informed. Lots of time and energy can be saved compared to not using git-related tools.

More and more colleagues switched to git-related tools whenever they started their R package or Shiny projects. This is a step forward compared to not using git. However, I noticed very few people use CI/CD tools that come along with git tools they are using.

They are missing huge efficiency and good development experience. CI/CD is not so much automatic testing tool as a workflow that urges the development team to think about testing and the way they cooperate.

CI/CD AND RELATED CONCEPTS

CI/CD is short for Continuous Integration/Continuous Delivery or Deployment.

Nowadays, popular CI/CD tools are: Github Actions, Gitlab CI, Jenkins, etc. In this paper, drone CI is used in the example. Compared to these CI tools listed, drone CI is more suitable for small team or individual developers.

Drone CI is lightweight – simpler in terms of configuration and easier to use compared to Jenkins. When compared to Github Actions and Gitlab CI, you can host your CI server on the same laptop that you write code. Everything stays on your own machine, rather than going to cloud.

Another advantage of using drone CI is that it is a docker image. Any dependencies or software you need to run testing could be downloaded from docker image repository, and installed without concerns of dependency conflicts, due to docker's characteristics. When you use a docker image, a virtual computer will be created inside docker for the software/application you want to install. The cost for docker to create such virtual computer is so low that it can easily create hundreds of virtual computers inside it. Thus each computer can dedicate to a single application. Therefore there will be no dependency conflicts.

Before diving into the example, several concepts need to be presented so that reader can understand the example better.

WEBHOOK

Webhook is an API (Application Programming Interface) that running in a git repository (in this example, github). It will constantly watch if chosen types of events happen. If one type of event happens, webhook will send an HTTP request to a chosen target. In this example, the target is drone CI server.

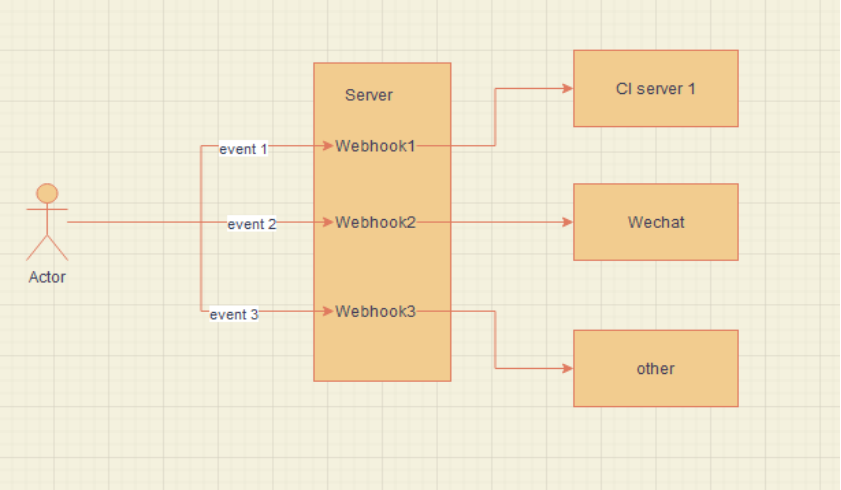

Figure 1. Webhook

CI SERVER, RUNNER

CI server, in this paper, is not a machine. It is an application that does continuous integration. In this example, CI server is a running docker image lives on my laptop.

Runner is also an application which is called and organized by a CI server. Runners execute tasks, such as executing bash commands. Below is an illustration of CI server and runners.

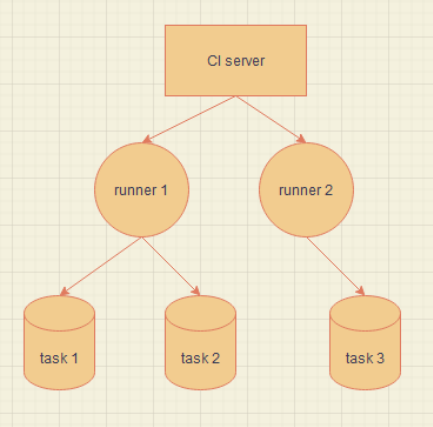

Figure 2. CI server and runners

STEPS, PIPELINE

Steps are basic and unit tasks, such as executing a bash command. Pipeline is a sequence of steps that aimed to do something. For example, a pipeline aimed to test each pull request. A project can have multiple pipelines to achieve test under different scenarios.

OVERALL PICTURE

Below is an overall picture of how a push triggers webhook and webhook finds drone CI server, and drone CI server all runners to execute steps in a pipeline.

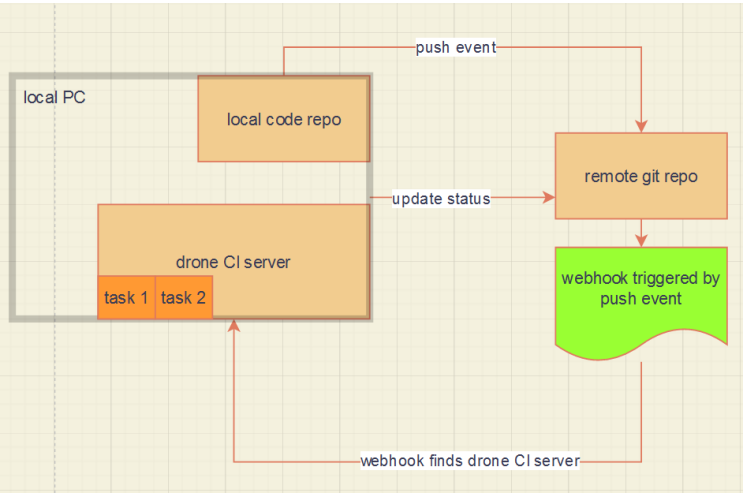

Figure 3. Overall picture

Code repository and drone CI server are all on local PC. When new updates in local code repository pushed to remote git repository, webhook watching push events is triggered and sends an HTTP request to drone CI server on the same computer. This HTTP request will start executing of pre-defined sequence of tasks. After execution completed, drone CI server will send a request to remote git repository to update execution status to show if it's successful.

USING DRONE CI TO DO STYLE CHECK AND ON_PUSH TESTING

A small toy R project having four files is shown in below screenshot.

.drone.yml (top-left) is a file that describles what steps will be executed. It is also described these steps will be executed on branch master and triggered by push event.

test_cases.r (top-right) is test cases to test if functions in test.r (bottom-right) are correct. test_linter.r (bottom-left) is a file having R code that does not follow certain good practices.

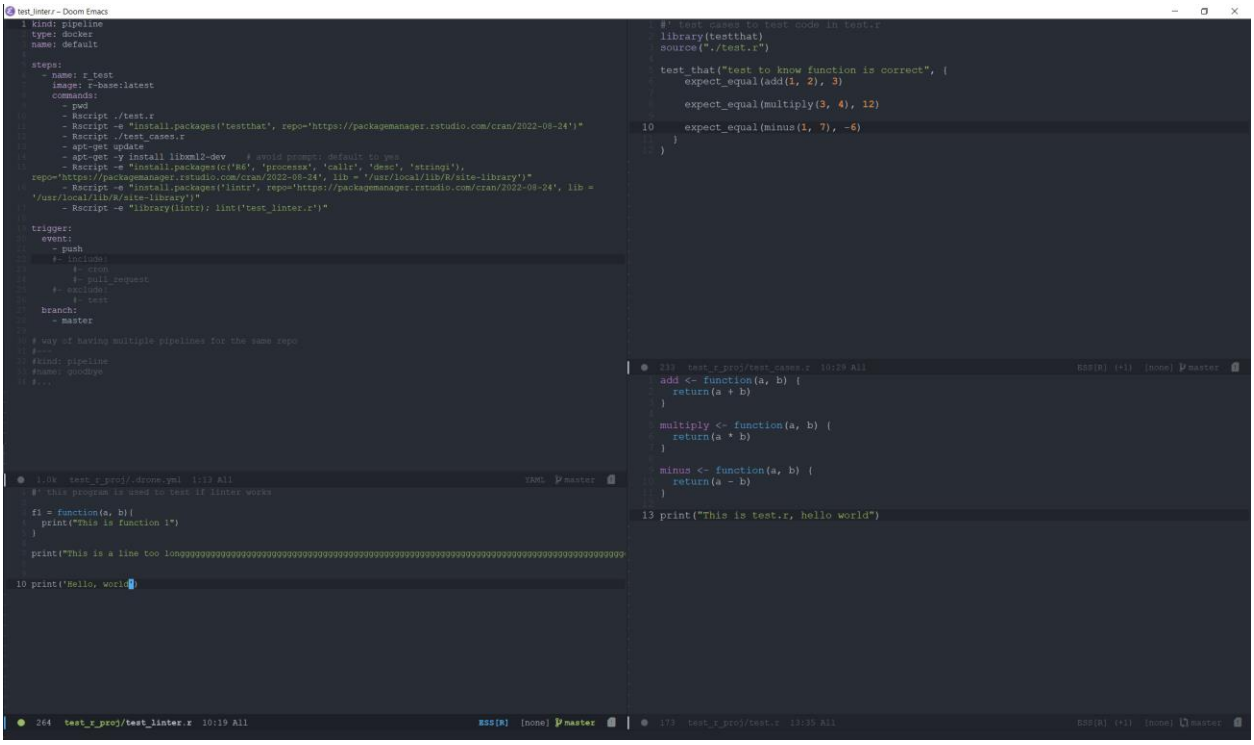

Figure 4. Example R project

When updates of this project pushed to remote repository, webhook will be triggered and send an HTTP request to CI server. Steps defined in .drone.yml will start to execute when this HTTP request arrived.

.drone.yml line 11 – 12 is trying to execute test_cases.r file. This step aims to confirm if current push does not break existing functionalities in test.r. Lines 13 – 17 is trying to run lintr::lint("test_linter.r"). This is an illustration on how to check coding style of an R file when it is pushed to repository.

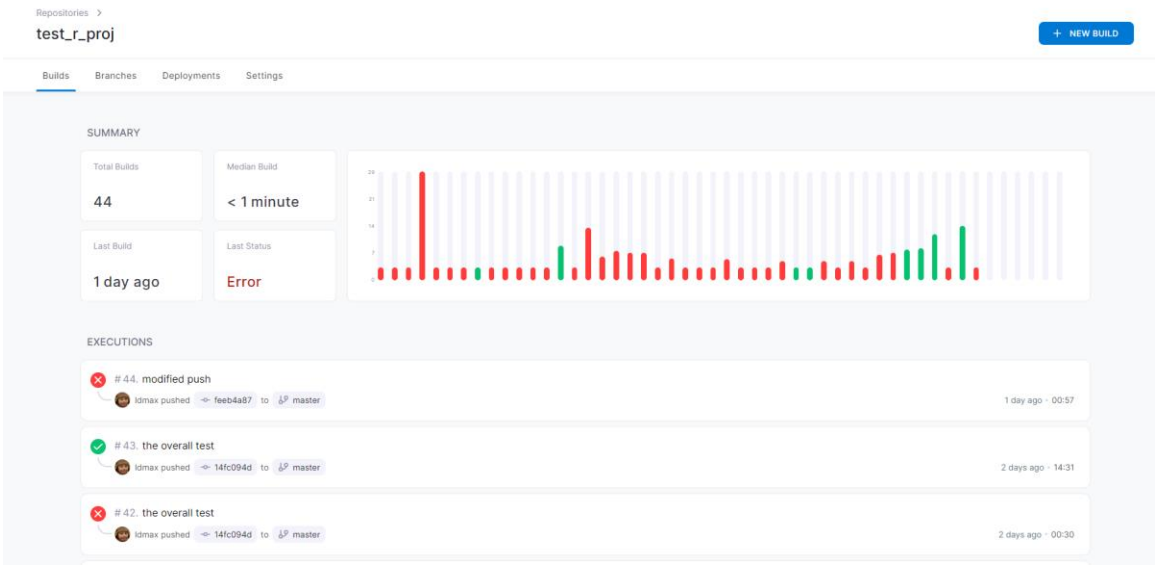

Figure 5. Build records on CI server

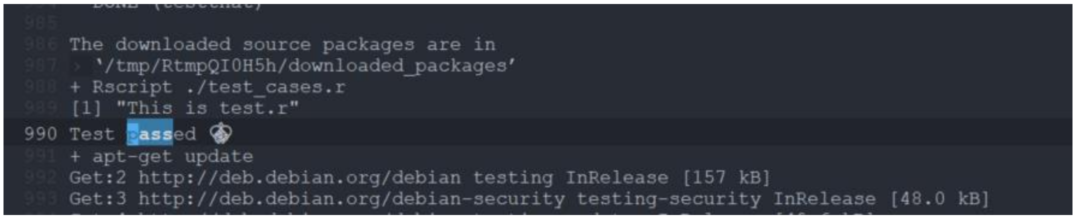

Figure 6. Test passed

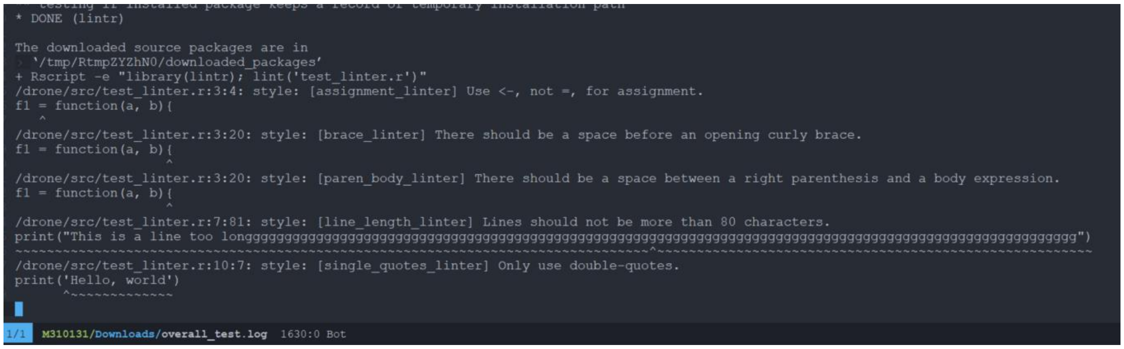

Figure 7. Lintr output

Visiting CI server page, user can see build records show in Figure 5. If clicking on one record, a log of pipeline execution will be displayed. Figure 6 and figure 7 displayed parts of this log. In figure 6, line 990 shows that the testing passed, meaning functions in test.r are correct after this push.

Figure 7 shows that the output of analysis of test_linter.r file. Bad coding style such as using "=" instead of "<-", or no space between ")" and "{" when defining a function, or a line with more than 80 characters, or quoting string with single quotes rather than double quotes will be warned.

This example shows that with the help of drone CI, this R project can analyze coding style and test R code when a chosen event (in this example push) happens. A further step could be writing scripts to refuse updates that failed to pass these testing cases, so that the quality of an R project will be better ensured.

REFERENCES

Drone CI, "Github" https://docs.drone.io/server/provider/github/

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Dan Li
dan-li@merckgroup.com