Continuous integration for containerized scientific workflows

AUGUST 2019

AUTHOR(S):
Leticia Farias Wanderley

IT-CDA-DR

SUPERVISOR(S):
Diego Rodríguez Rodríguez

CERN openlab

PROJECT SPECIFICATION

REANA is an open-source reusable analysis platform that allows researchers to run their analyses seamlessly in a containerized cloud computing environment. The platform takes as input the analysis' data, code, computing environment, and computational workflow and executes it in a distributed containerized compute cloud infrastructure.

The goal of this project is to enable a continuous integration service for researchers performing physics data analyses on containerized compute clouds. The project aims to integrate GitLab [1], a collaborative code development platform, and REANA. The CERN's GitLab instance is used by numerous physics groups to bootstrap and develop data analysis code.

The physics analyses at CERN typically last numerous months with the code contributions from individual researchers and small teams. The integration between REANA and GitLab will enable the GitLab users to have REANA as a scalable continuous integration system for scientific analysis projects hosted on GitLab.

.

ABSTRACT

On this project, we decided to implement two solutions that integrate REANA and GitLab. They vary on two main points. The first one is the amount of configuration necessary to set up the integration, and the second is the flexibility level allowed by the integration, that is, how much the user can customize it.

This decision was made to attend the users' different needs. The first solution relies on GitLab's continuous integration, continuous deployment environment, known as GitLab CI/CD, it gives the users with more specific and advanced requirements the freedom to customize their analyses executions. The second solution is a turnkey plugin that covers most of the general use cases. It is implemented as a GitLab application that connects to GitLab using the users' authorization.

TABLE OF CONTENTS

1. INTRODUCTION

According to a recent Nature [2] survey, scientists from the most varied fields are facing a reproducibility crisis. When asked whether they had ever failed while trying to reproduce their own experiments more than 50% answered yes. When questioned about the reproducibility of someone else's experiment, more than 70% said that they had failed to reproduce someone else's results.

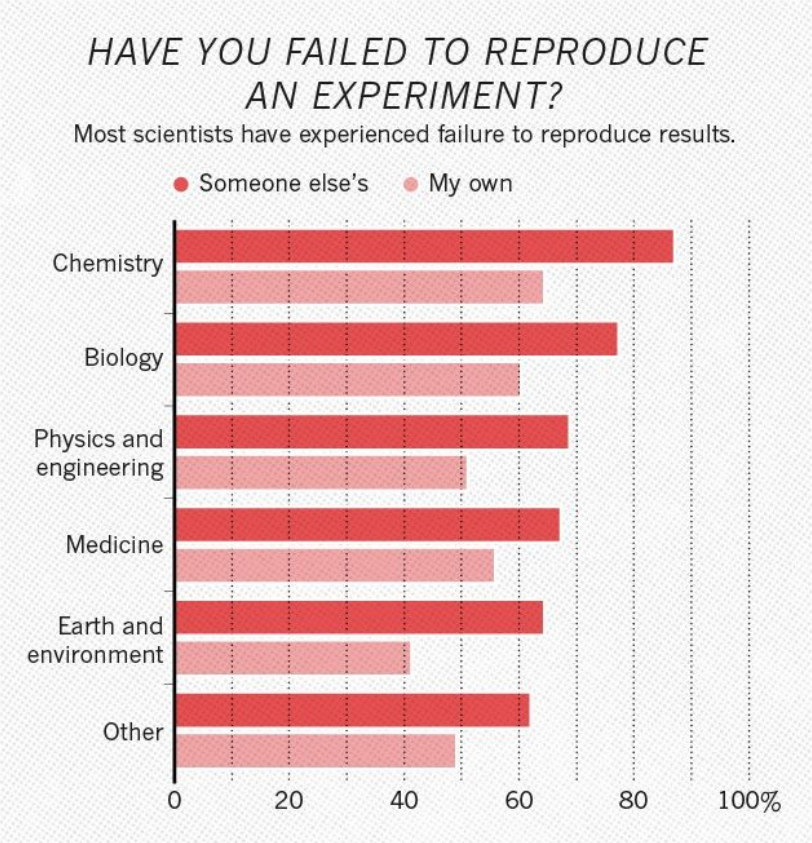

Figure 1: Reproducibility failures in science [2]

However, one of the survey's questions revealed that even though most of the scientists had trouble reproducing experiments and agree that there's a reproducibility crisis happening, less than 31% of them attributed the lack of reproducibility to the results being wrong. To them, the reproducibility challenge lies elsewhere.

Another issue hindering reproducibility is the high turnover of researchers in scientific research centres. Examining CERN it is possible to observe how there is a great rotation of researchers working on each of the experiments. When a researcher leaves an experiment the newcomers have to resume his or her work and it is common for them to take a while to adapt to the new environment and codebase.

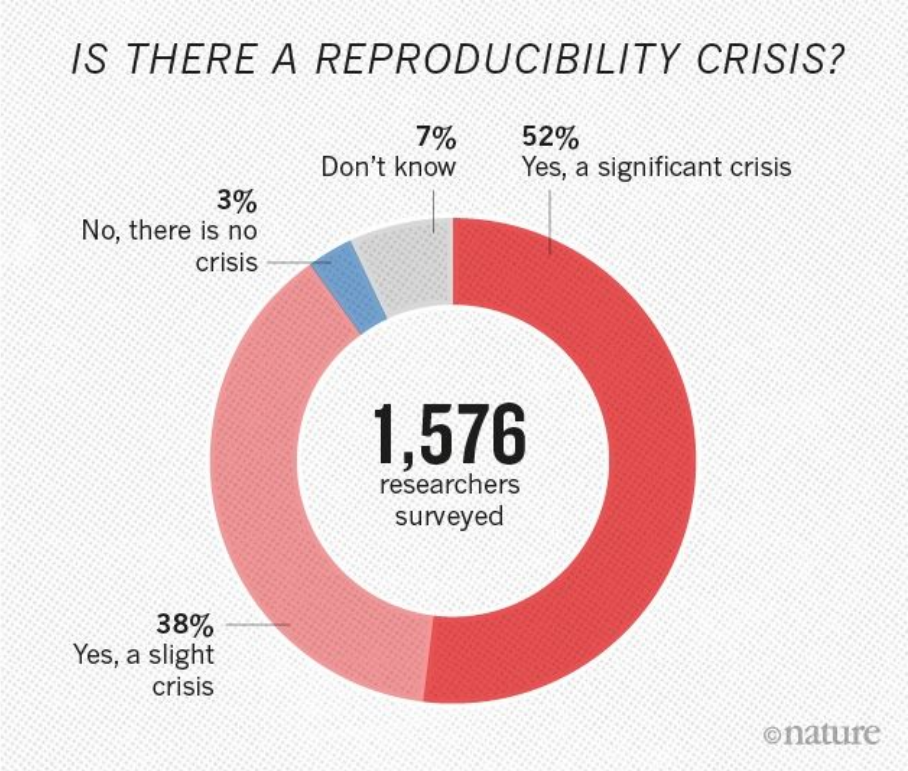

Figure 2: Reproducibility crisis [2]

Thinking about these issues, the IT-CDA-DR section at CERN proposed a new open-source platform to aid research reproducibility. REANA is a reusable and reproducible research data analysis platform that allows its users to structure their analyses and execute them on a distributed containerized compute cloud.

The REANA platform is composed of several microservices, each responsible for some part of the analysis execution or the connection between the scientist's workstation and the server on the cloud cluster. The REANA-Client component, for example, provides a command-line interface for the REANA user to connect to a REANA cluster, create and execute an analysis, and download the analysis' results.

Using REANA-Client [3] was so far the only way for scientists to execute analyses on a REANA cluster. It meant that they would have to not only learn how to use the command-line interface but also manually re-execute the analyses to check their reproducibility after updating something. With this project, we endeavoured to make this process easier by using REANA as a continuous integration service connected to GitLab.

A continuous integration service works by integrating the changes to the codebase on every update, it means that after every change added to the project, an execution build is automatically triggered [4]. The REANA GitLab integration allows GitLab to execute projects on a REANA cluster whenever those projects are updated and to give immediate feedback about their execution status to users.

2. REQUIREMENTS

Analyses consist of at least four main components, data, code, environment, and workflow. This means that to guarantee the reproducibility and reusability of an analysis the researchers must keep track of all those components. Whenever any of the components changes or is updated, no matter how big or small the update is, there must be a comprehensible way to ensure that the analysis is still executable.

The REANA GitLab integration is that way. It triggers a new REANA analysis execution whenever the analysis project hosted on GitLab is modified and provides real-time feedback to the user about the execution progress. We defined two distinct use cases for this integration. The first attends users who need the freedom to tailor their REANA analyses executions and the second attends the general use case in which users need a simple way to continuously integrate their analyses.

a. Configurable solution

This solution's users require a highly customizable integration. Apart from executing the analysis that is hosted on GitLab on a REANA cluster, they need to be able to run specific commands before, during, and after the REANA analysis execution. With this solution, for example, the users can perform custom calls whenever the analysis execution meets certain conditions. The solution needs to:

- Allow users to have freedom before, during, and after the REANA execution;
- Show the REANA logs on the GitLab CI/CD interface;
- Display the analyses' results on the GitLab interface.

b. Turnkey solution

The users of the turnkey solution need a simple solution to continuously integrate their analysis, They are general users that do not need a lot of customization and are satisfied with the default setup automatically created by the integration. This solution allows the user to add REANA as a continuous integration application in a few clicks. The solution needs to:

- Allow the selection of GitLab projects to be executed on REANA;
- Notify REANA when GitLab projects are updated;
- Notify GitLab when analysis executions on REANA change status.

3. SOLUTIONS

As previously mentioned, we decided on having two solutions to integrate REANA and GitLab. One of them uses the GitLab Continuous Integration, Continuous Deployment, and Continuous Delivery toolset, known as GitLab CI/CD [5], to facilitate the connection between GitLab and a REANA cluster. The other solution is a GitLab Application that gets the user's authorization to connect REANA to the user's GitLab projects.

a. GitLab CI/CD

The GitLab CI/CD toolset allows GitLab to verify code changes added to projects. It works by watching the changes then building, testing, and validating them before adding them to the main branch [5]. For the

REANA GitLab integration, the CI pipeline needs to execute the analysis project hosted on GitLab on a REANA cluster, then it should display the execution results on the GitLab interface.

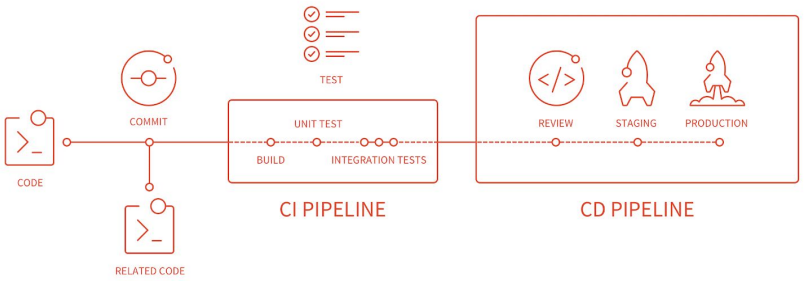

Figure 3: GitLab CI/CD [6]

The principal component of this solution is the *.gitlab-ci.yaml* file [7]. This file defines the steps that constitute a continuous integration pipeline. It defines the environment in which to execute the pipeline, the commands necessary to execute it and what to do with the artefacts created by the pipeline, i.e., the analysis results.

The setup the integration, GitLab has to communicate with a REANA cluster to create the analysis on the cluster, upload the required files, start the analysis execution, follow its status, and eventually download its results. This process happens via the REANA-Client component, the command line interface tool installed and run by the GitLab CI/CD builder on a containerized environment.

Installing the REANA-Client component should be one of the first commands defined on the *.gitlab-ci.yaml* file. It enables the use of REANA's command-line interface on the GitLab CI container. After this installation, the pipeline is able to execute REANA-Client commands, which should also be instructions on the *.gitlab-ci.yaml* file, and communicate to a REANA cluster through its REST API.

After adding the *.gitlab-ci.yaml* file to the project's root and setting the REANA authentication parameters as environment variables on the GitLab web interface the GitLab users have a connection between GitLab and a REANA cluster. The GitLab CI/CD toolset will take care of triggering a REANA analysis execution on every new commit added to the project, and the users will have a guarantee of its reproducibility.

b. GitLab application

This solution requires less configuration than the previous one. It requires the GitLab users to authorize REANA to access their projects. After the authorization is granted, the application takes care of setting up a webhook on a GitLab project that will let REANA know whenever a new commit or new merge request is created.

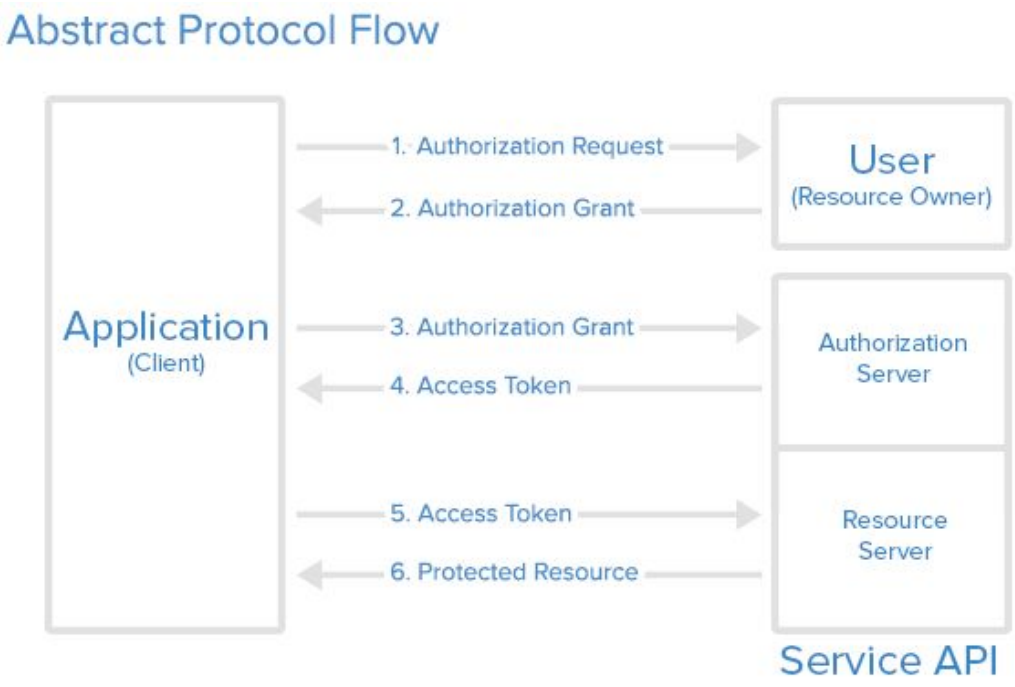

Figure 4: Authorization granting flow [8]

The webhook created by REANA inside GitLab will trigger new runs on every new commit to the master branch in the project repository and on every new merge request created. Since the user granted REANA permissions to access the project, it will update the status for the given commit in the GitLab UI whenever something changes on the REANA execution.

4. IMPLEMENTATION

a. GitLab CI/CD solution

This solution uses the GitLab CI/CD toolset as its foundation. The toolset handles the installation of the REANA-Client service and the subsequent creation and execution of the scientific analysis hosted on GitLab. It also manages the user feedback and the analysis' artefacts retrieval after the execution finishes.

The integration also relies on the REST API endpoints used by the REANA-Client service. Its flow begins when the users commit a new change to the project repository. The commit prompts the GitLab CI/CD toolset to initialize a new CI pipeline, which will execute the REANA-Client commands to connect with a REANA cluster and create and start an analysis execution.

While the analysis is running the REANA-Client service keeps pooling the REANA cluster for updates, following the execution and preventing the CI pipeline to stop before the execution finishes. Once the analysis run successfully ends the CI pipeline executes a command to list the analysis' output files' URLs. The users can then follow the URLs and access the analysis' outputs via the REANA UI service, which will ask them to login using CERN Single Sign-On service [9]. After the users authenticate, they are redirected to the output endpoint and can download the analysis' output files. This whole interaction is documented as a sequence diagram in Figure 5.

Those are two important features that were added to the REANA-Client service to allow this integration. The first one, following the analysis execution until it finishes, is the *--follow* flag, this new flag defines whether the analysis execution should be tracked, that is if the terminal should keep waiting for execution updates after starting the analysis execution. This flag was essential to the integration because otherwise, the CI pipeline would not be able to retrieve the execution status correctly.

The second feature is the option to list the URLs of the analysis' results. This feature allows GitLab users to access and download the analysis' results via the web browser instead of the terminal. It provides a link to the results in the logs after a successful execution. Both features were added to the REANA-Client component, as it is the tool used by the CI pipeline to communicate with a REANA cluster.

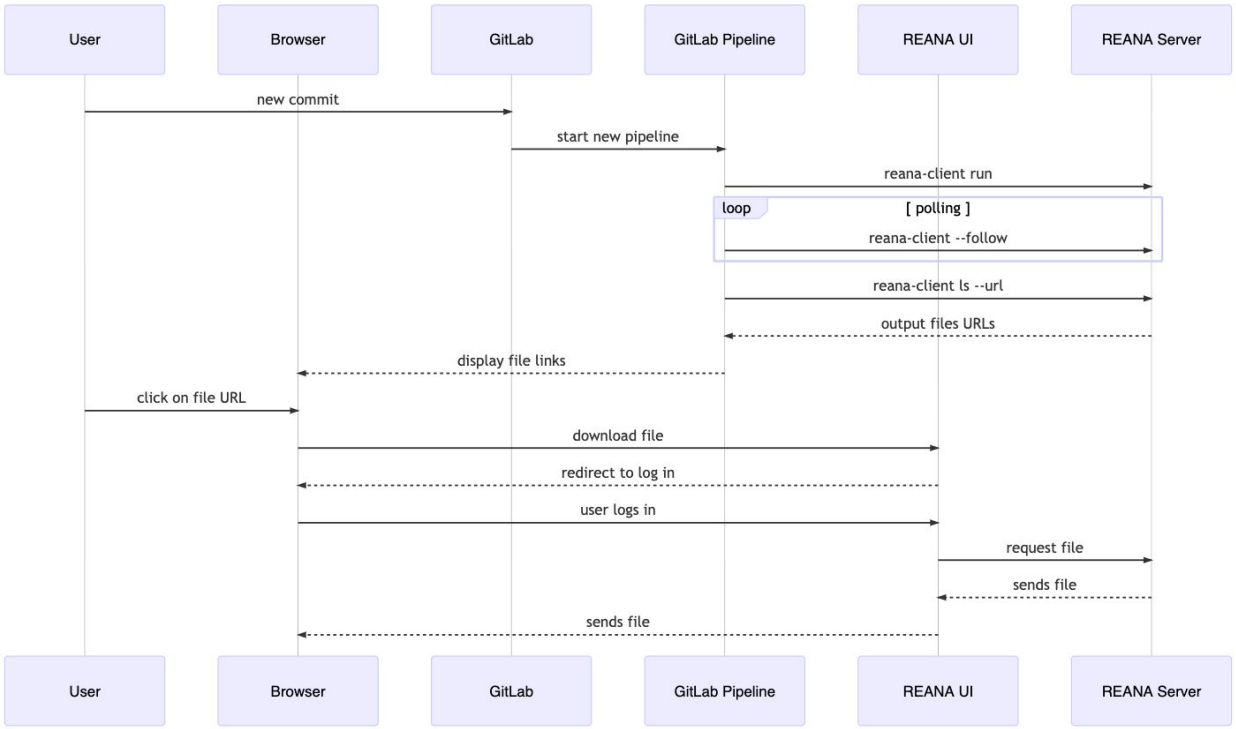

Figure 5: GitLab CI/CD solution sequence diagram

As mentioned before, to set the GitLab CI/CD properly the users need to define a *.gitlab-ci.yaml* file containing a set of steps that download and install the REANA-Client, create an analysis execution, start the analysis, and finally download the results from it. The code snippet below shows an example of a *.gitlab-ci.yaml* file for the REANA GitLab integration.

```
reana:

  image: "python:3"
  script:
      # install reana-client
      - pip install reana-client

      # create the workflow
      - reana-client create -n $CI_PROJECT_NAME
```

```
# export workflow name
- export REANA_WORKON=$CI_PROJECT_NAME

# upload analysis' code and input files
- reana-client upload <code_path> <data_path>

# execute the workflow
- reana-client start --follow

# download output files
- reana-client download <output_files_path>

artifacts:
    paths:
        - <output_files_path>
    expire_in: <expiration_period>
      when: on_success
```

As a result of this implementation, the users only need to interact with the GitLab UI to use REANA as a continuous integration service. After setting the GitLab CI/CD environment variables and creating a *.gitlab-ci.yaml* file similar to the example above, every new commit to the project repository will trigger an analysis execution and the users will have real-time feedback about their analysis reproducibility. The end result of this solution is shown in Figure 6.

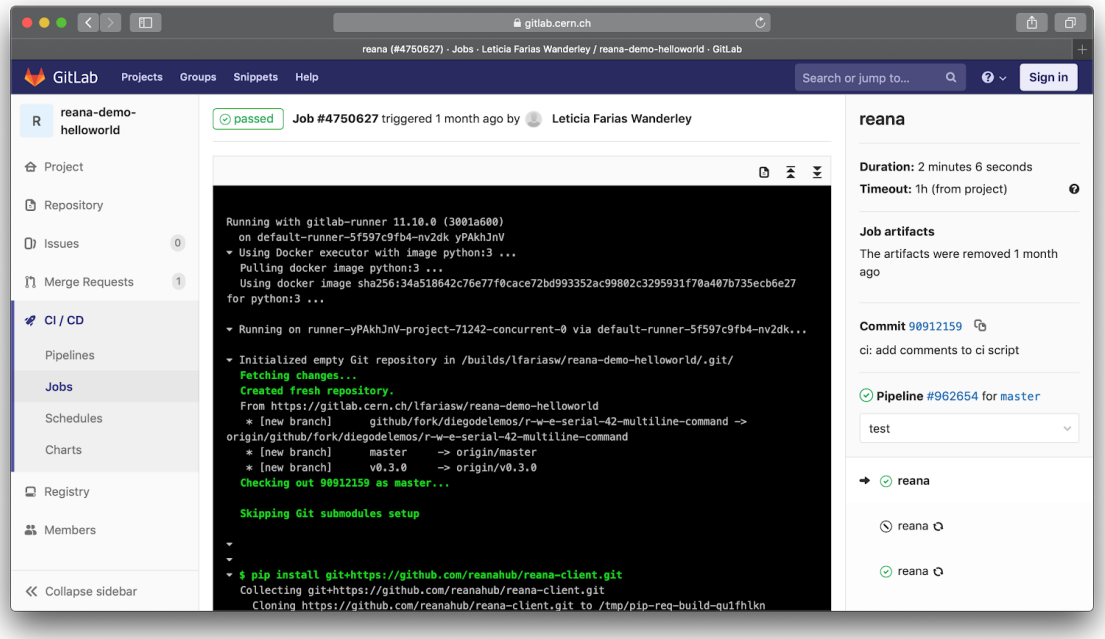

Figure 6: GitLab CI/CD solution's CI pipeline logs

b. GitLab application solution

The implementation of the REANA GitLab application involved more REANA services. One of the pressing points of this solution's development was granting GitLab's authorization to the REANA application. To do that the REANA platform also needs to ensure that the users are properly authenticated and secure. This solution's workflow begins with the users authenticating themselves via the REANA UI service. This service redirects them to the CERN SSO login page which checks if there is a REANA OAuth 2.0 [10] client registered at CERN and, then, if the users who are trying to authenticate themselves have entered the correct credentials.

After those two checks are completed and successful, CERN SSO redirects the user back to the REANA-Server service that finishes the OAuth 2.0 flow by requesting the authorization token to CERN SSO and redirecting the user back to the REANA UI while also setting a session cookie on the users' browsers. The workflow described above was included in the REANA platform via yet another integration. The package Invenio-OAuth-Client [11] was incorporated as one of the REANA-Server service's dependencies. This package provides the OAuth client endpoint structure necessary to connect to, among other services, CERN SSO.

Once authenticated at CERN, the users can interact with the REANA UI and initialize the GitLab authorization process [12]. Similarly to what happens with CERN SSO, users are redirected to GitLab's authentication page, after authenticating themselves on GitLab they are prompted to authorize REANA to access GitLab's API as the user (Figure 7). Once the users agree to this, the users are redirected to the REANA-Server service, that stores the GitLab's authorization token and redirects the users back to the REANA UI.

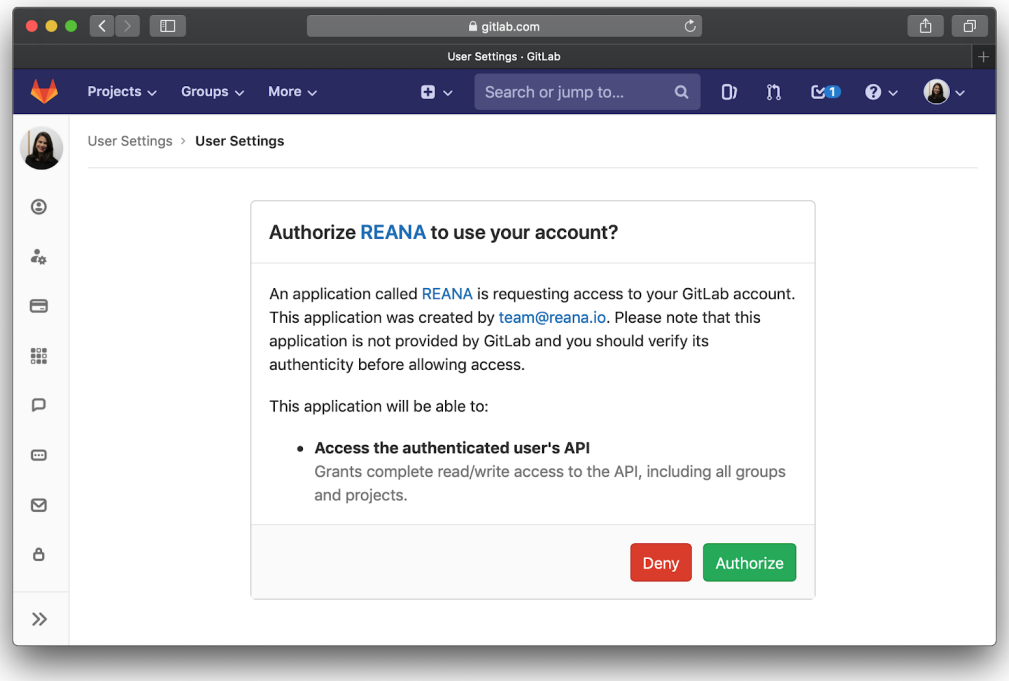

Figure 7: Authorizing the REANA application on GitLab

At this point, the REANA-Server service is able to read the users' project and so it does as the UI prompts the users to select which GitLab projects should be continuously integrated by REANA. As the project is selected, the REANA-Server service creates a webhook on GitLab, again using the users' authorization token, so that on every commit to the master branch and on every new merge request the webhook is triggered. This means that on every change, GitLab makes a REST call to the REANA-Server service to create and execute that project's analysis. The whole interaction flow is depicted as a sequence diagram in Figure 8.

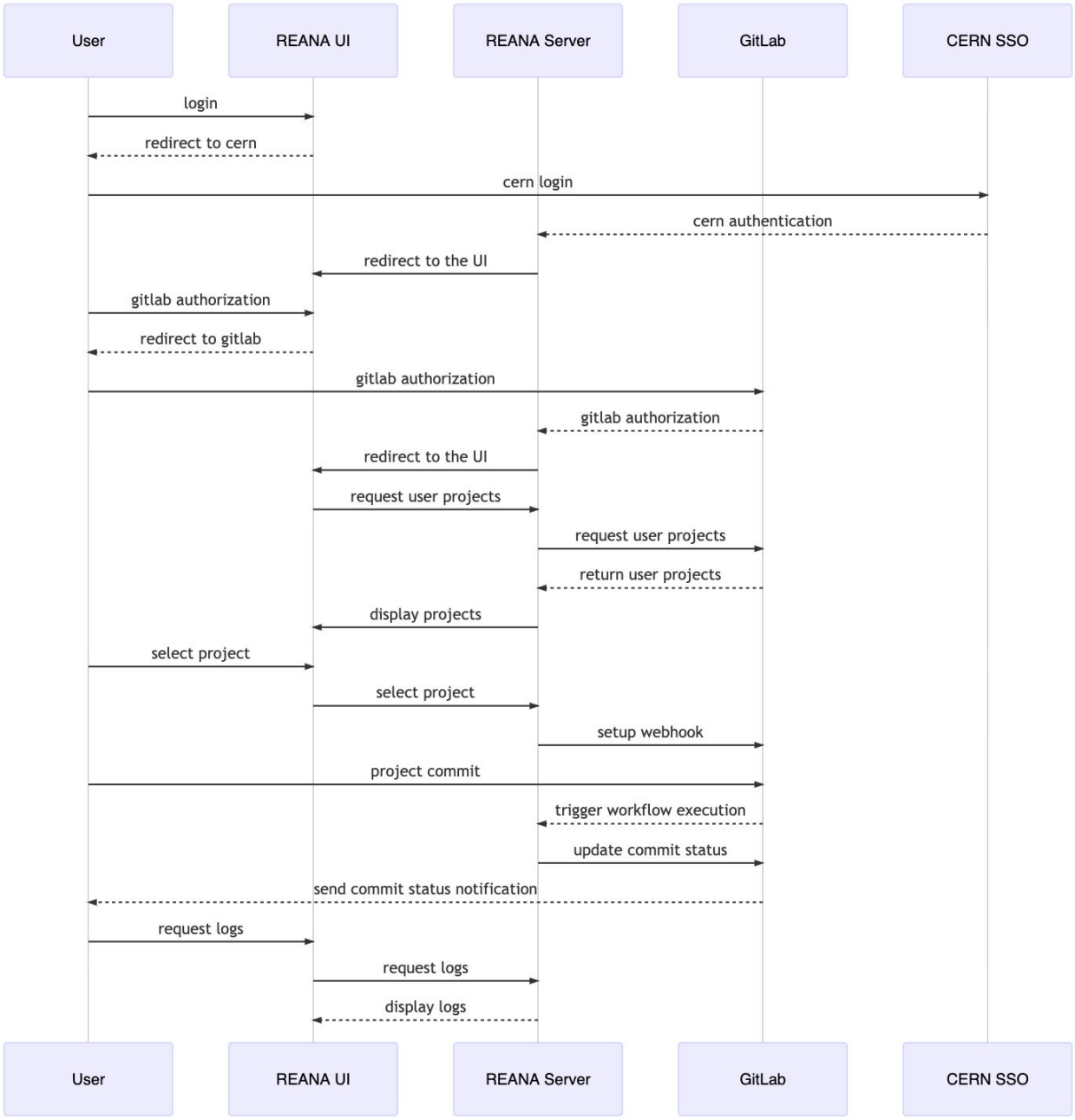

Figure 7: GitLab application solution sequence diagram

As soon as the execution begins, the commit that triggered it gets updated with the “running” status [13]. This update is made by another REST call from REANA to GitLab using the user’s authorization and it means that the continuous integration build has started executing the code from that commit. Whenever the execution finishes, be it successful or not, the commit status is updated again, this time with the final outcome (Figure 9). By clicking on the commit status badge on the GitLab interface the users get redirected to the execution logs page on REANA.

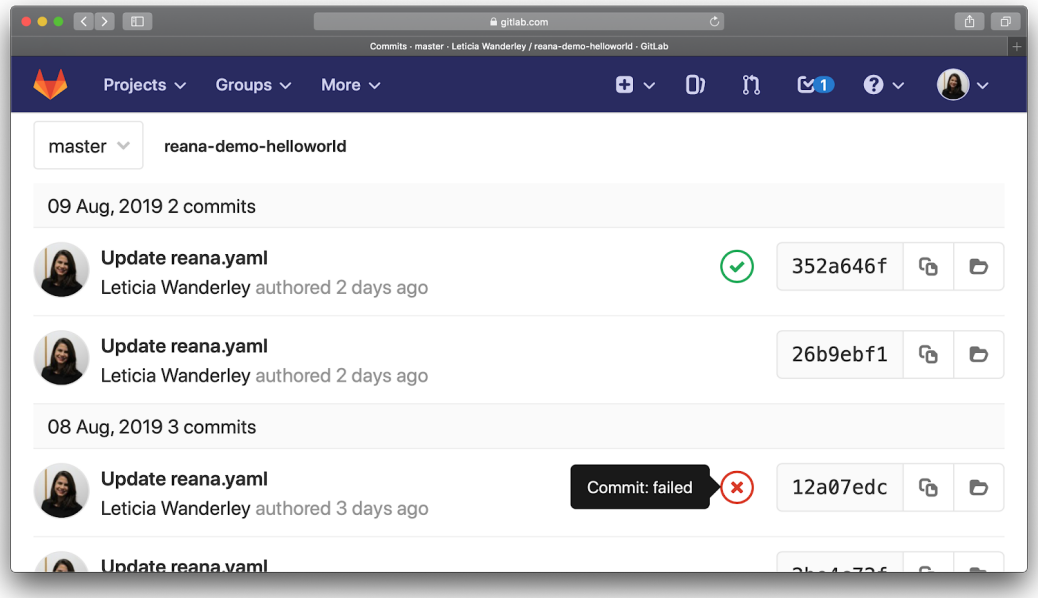

Figure 9: Commit status on the GitLab web interface

5. CONCLUSION

This report presents the results of the design and implementation of two fully functional solutions for the integration between REANA and GitLab. Both solutions are integrated into the REANA platform codebase as they accomplish the requirements set at the beginning of the project.

They allow GitLab users to have REANA as a continuous integration service and cater to the needs of different users. The GitLab CI/CD solution aims at flexibility and it requires more technical knowledge of its user. The GitLab application solution is meant to be an easy way to plugin GitLab and attends users with fewer specific requirements. Table 1 presents the main differences between the two implemented solutions regarding configurability and Ease of Use.

	GitLab CI/CD solution	GitLab Application Solution
Configurability	High	Low
Ease of Use	Low	High

Table 1: Configurability vs Configuration: Gitlab integration solutions

There is still work to be done to improve the user experience with the REANA GitLab integration, mostly user interface related:

- Improve and setup the navigation flow between REANA and GitLab so users do not get confused while moving between applications;
- Create a richer page to visualise the workflow logs. Currently, only plain text is available;
- Enhance the page that shows which projects have been integrated with GitLab;
- Make GitLab Application solution configuration easy to change for a given REANA deployment.

6. REFERENCES

[1] https://about.gitlab.com/

[2] Baker, M. (2016). 1,500 scientists lift the lid on reproducibility. *Nature News*, *533*(7604), 452.

[3] https://reana-client.readthedocs.io/en/latest/

[4] https://martinfowler.com/articles/continuousIntegration.html

[5] https://docs.gitlab.com/ee/ci/

[6] https://forge.etsi.org/rep/help/ci/README.md

[7] https://docs.gitlab.com/ee/ci/yaml/README.html

[8] https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

[9] https://sso-management.web.cern.ch/

[10] https://oauth.net/2/

[11] https://invenio-oauthclient.readthedocs.io/en/latest/

[12] https://docs.gitlab.com/ee/api/oauth2.html

[13] https://docs.gitlab.com/ee/api/commits.html#post-the-build-status-to-a-commit