Continuous Integration Report

Team 15

Joe Wrieden

Benji Garment

Marcin Mleczko

Kingsley Edore

Abir Rizwanullah

Sal Ahmed

Continuous Integration Report v0.5

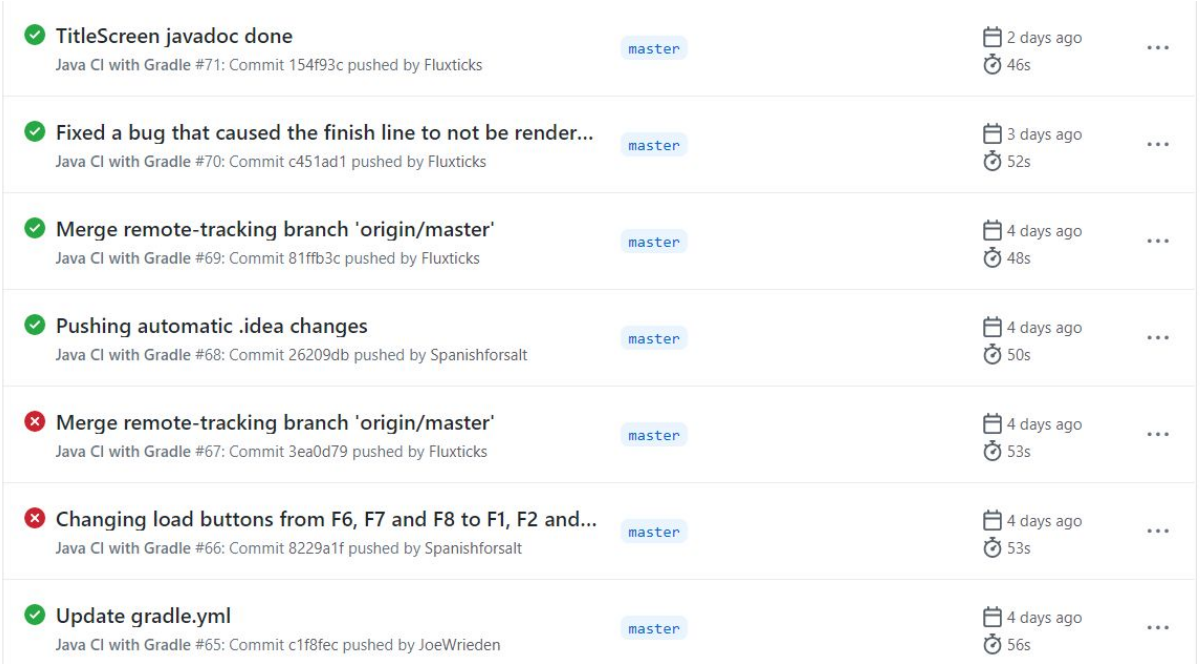
Preface

- v0.1 Creation of document, introduction to the continuous integration methods and its justification
- v0.2 Added brief report on the continuous integration infrastructure we set up.
- v0.3 Continuous integration methods and approaches completed, but the section needs reviewing
- v0.4 Brief report on the CI infrastructure is completed but the section needs reviewing
- v0.5 Completion of Continuous Integration document after final review

Continuous Integration Method and Approaches and their Justifications

Continuous Integration is a Software Development practice which consists of team members integrating their work into the main code base frequently, with the code being verified at each integration [2]. The code is verified by checking if every integration build passes. CI enables us to create cohesive software by avoiding different versions of the software from diverging from the code base to a significant extent, and so conflicts are reduced when merged and bugs can be detected earlier [4]. CI allows all members to easily see the current state of the system and the changes that have been made to it [3].

The build will only be executed if there are no errors in building or testing. The screenshot of the CI dashboard below shows some of the builds that have occurred in the project. The green ticks show when the result of the code pushed (consisting of both a build and full run of all unit tests) was successful, whereas the red crosses imply that something has failed:



The automation of these tests played a big part in our Software Verification, as it helped catch any bugs present in the program and with the frequent integration stated earlier, reduces the amount of bugs that are found in the program making it easier to highlight any problems as they should be easier to spot.

As the system is not very large, we can frequently build and test our system several times per day ensuring that the system runs successfully despite the numerous changes. The benefit of frequently integrating the system is that it increases the possibility of finding problems stemming from the component interactions early in the process. It also helps support well structured unit testing of the components.

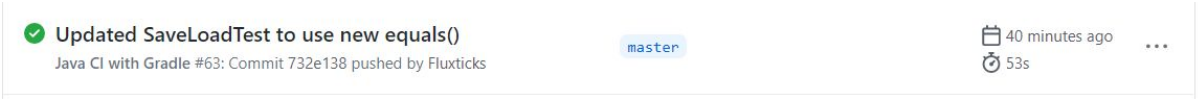
Brief report on CI infrastructure

Our CI solution consisted of GitHub as our CI Server and GitHub Actions's Java with Gradle workflow as our CI Agent. We configured our CI Server via a text configuration file in YAML. This configuration file went into our source control management system: GitHub. The Continuous Integration workflow that we created allowed us to automate the running of Unit tests that we created, specifically JUnit 4 tests, on the master branch as well as on the testing branch of the GitHub Repository. This is an alternative to triggering the build manually, which would require more time and effort.

As stated above we used Gradle for automation build , as it was the build system we felt suited our needs best. Gradle allowed for build issues to be logged for further investigation, without catastrophic build fails. It also allowed us to build different test suites [1]. Gradle also identifies the program's dependencies, and automatically generates a build script from the program. We were able to include automated tests - both unit and integration - into the automated build process using test automation tools. Through this and the CI dashboard interface we are clearly notified of the result of the build, as in if the build has been broken by any of the changes that have been made in the code recently. By being notified of the unsuccessful team, we as a team can focus on fixing these issues and reduce the number of unsuccessful tests.

Since the Version Control System we were using to store the code and all additional artefacts required for building was GitHub, we decided to use GitHub Actions. GitHub Actions worked in conjunction with Gradle to build the executable whenever we pushed the code to the GitHub repository - allowing for seamless building [5]. GitHub's functionality also extends to keeping track of code changes and merging conflicts automatically for the most part was also helpful in diverging versions of the software. This made it easier to automate our workflow and implement the Continuous Integration tools with ease [6].

The infrastructure of the Continuous Integration workflow is as follows:



Above shows the commit where the SaveLoadTest was implemented. It has been pushed to the master branch by one of our members at which point the Java Continuous Integration with Gradle starts work on running the test. The framework creates an Ubuntu instance which runs the commands specified in the YAML file which in our case is a full build and run of all Unit Tests. In this case, it was successful and further information can be obtained by clicking on the specific action and looking within the different steps run and logged during the test.

To ensure that an unsuccessful build is not overlooked, an immediate notification in the form of an email is sent to the team members listed as collaborators. Included in the email will be a pointer to where the error is, for easier correction of the code.

References

[1] "Building and testing Java with Gradle - GitHub Docs", *Docs.github.com*, 2021. [Online]. Available: https://docs.github.com/en/actions/guides/building-and-testing-java-with-gradle [Accessed: 02-Feb-2021].

[2] "About continuous integration - GitHub Docs", *Docs.github.com*, 2021. [Online]. Available: https://docs.github.com/en/actions/guides/about-continuous-integration [Accessed: 04-Feb-2021].

[3] "Continuous Integration" Available: https://martinfowler.com/articles/continuousIntegration.html [Accessed: 04-Feb-2021].

[4] “Continuous Integration”, *Science Direct.* [Online]. Available: https://www.sciencedirect.com/topics/computer-science/continuous-integration [Accessed: 04-Feb-2021].

[5] “GitHub Actions Docs”, [Online]. Available: https://docs.github.com/en/actions [Accessed: 04-Feb-2021].

[6] “What Are GitHub Actions and How to Use Them”, [Online]. Available: https://blog.bitsrc.io/what-are-github-actions-and-how-to-use-them-e89904201a41 [Accessed: 04-Feb-2021].