5. Continuous Integration

Team 10 | YorKorsairs

Abdullah
Tom Burniston
Omer Gelli
Adam Kirby-Stewart
Sam Mabbott
Benjamin Stevenson

A. Methods

The primary purpose of continuous integration (CI) is to automate the integration of code changes from multiple contributors into a single software repository. Continuous integration entails two components:

Building / testing new code

We have written our tests in conjunction with implementation, so that the team members writing the tests can communicate with those implementing the code and have an understanding of how the code works and the individual units involved in newly implemented classes. Despite this being a slower approach, it allows for more accurate and well implemented tests. Once a class has been tested, the test files can be added to the CI process, so that it can then be run automatically by the merging process.

Merging it into a central repository

We use GitHub to store our central codebase and CircleCi to automate testing and merging new code into the GitHub repository. These methods allow us to significantly reduce breaking changes and failure rate of our production code. By using a centralised repository, it allows multiple contributors to work on the project and push changes to a central place - this means that we can easily synchronise the code / tests between everyone and ensure consistency.

In addition to this, our implementation of CircleCi allows for automatic email sending caused by test failures which is composed with a detailed report of the errors and issues within the build process. This is beneficial because this engenders a collaborative approach to solving erroneous bugs and problems which prevent work progression. Furthermore, CircleCi provides an encompassing test report that details not only failures but also successes across all of the previous commits, to allow for a broader view and understanding of the project's progression – this advantageous system improves our outlook and helps prioritisation within task breakdowns which enable a smooth and efficient work output.

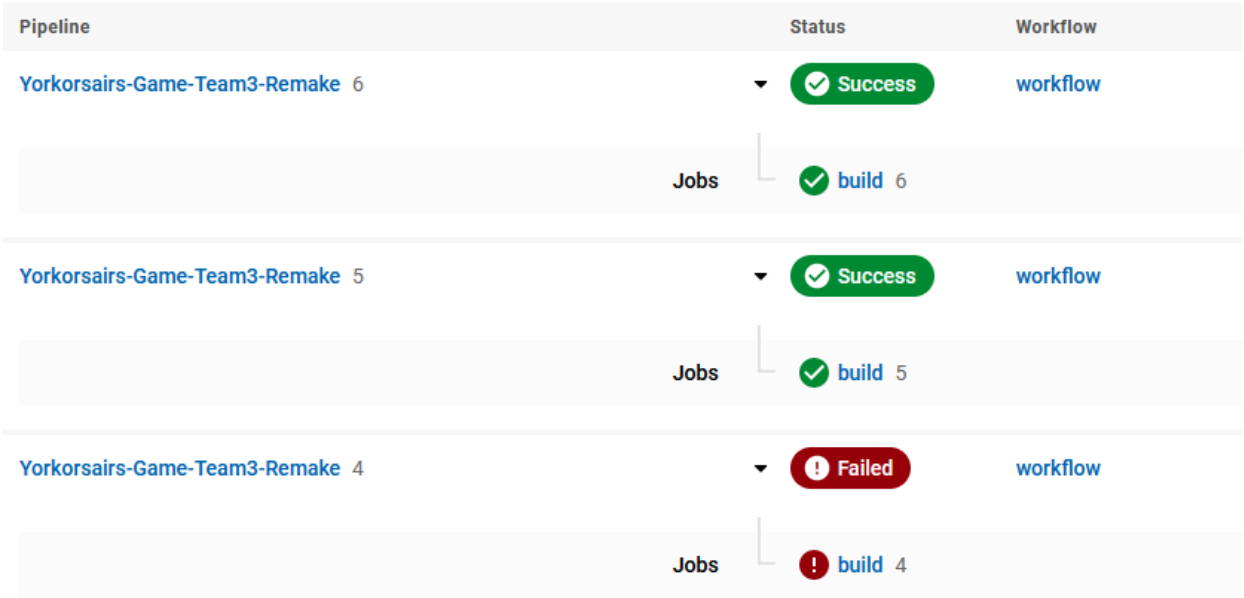
B. Infrastructure

Technologies

As part of our continuous integration (CI), we use CircleCi, GitHub, JUnit5, and Mockito to run tests and automate the merge process. CircleCi is connected to the central repository which is hosted on GitHub.

Methodology

Each time a contributor pushes new code to the repository, a new docker Java container is spun on CircleCi which runs the "gradle build" command and builds the entire project, runs the tests, and outputs a status code of either pass or failure. If the status code is passed, CircleCi will merge the new code into the GitHub repository, otherwise, an error message is displayed on the dashboard with the tests that have been failed as well as an email stating failure of workflow is sent to the admin.



Within the code there is a configuration (config.yml) file which dictates how CircleCi manages the project which allows us to specify a docker image to ensure consistency between testing instances. This file includes Java environment variables (JVM options) which specifies RAM limitations, as well as the precise steps that Gradle will run through. Having this config in an accessible directory allows for rapid development changes. Moreover, the use of Docker and JVM options imparts the specificity which is a requirement for us to implement the tests.

https://aj141299.github.io/Kroojel.github.io/ci.html