Cohort 3 - Group 22: The Wafflers

Eng1 Assessment 2

Change Report

Team Members:

Tunahan Sisman
Yousef Omar
Merrill Davis
Tom Comrie
Rohan Sandhu
Cameron Pounder
Daniel Redshaw

6a) On a commit to the main repository, the integration is verified via automatic build and tests. The integration pipeline is as follows:

The CI server sets up environments for windows, ubuntu and macos, the three platforms the project is designed to be built and run on. Each environment is designed to be as similar to the production environment as possible. This includes the same Java and gradle version.

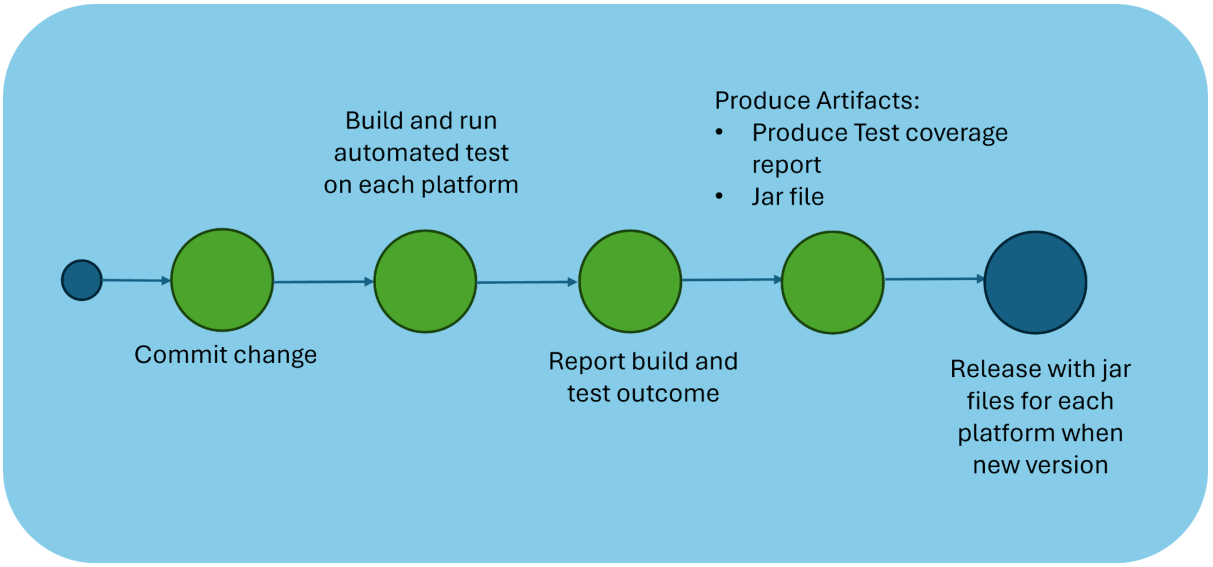
In each environment, the project is built, the automated tests are run and then the test coverage report generated along with the JAR, using the same single command used in the production environment, simplifying the process for the developer.

If either fails, then the pipeline returns the associated error messages, in the same way it is returned in the production environment.

If successful, the pipeline then outputs the artifacts, the automated testing report and the JAR file for each platform.

If the commit indicates it's a version to be released, then the pipeline takes as input the three JAR files for each platform and automatically releases them with the associated version name alongside the source code.

If the commit does not indicate it is a new version then this stage of the pipeline is skipped.



6b) The project's continuous integration pipeline was implemented using GitHub Actions, with the workflow set out in the YML file.

The Github Action contains two jobs: build and release, with release requiring build to have been completed successfully to be run.

In the build job, an OS matrix is used for the job to be run multiple times, one for each platform, on the latest version. (windows, ubuntu and macOS). These runs are executed in parallel to increase the speed of the integration.

The steps in the build job are as follows: Actions are first run to setup JDK 11 and gradle wrapper environment, so that they are the same as the production environment. Then a command is run to make the gradlew executable, which is required for the macOS and ubuntu environments to build.

In the next step, the single command used by developers in the production environment is then run, "./gradlew build jacocoTestReport dist". This builds, tests and then produces the JAR file for the project. (assuming the build compiles successfully and all automated tests pass).

The next two steps then use the upload artifact action to produce the output JAR file and coverage report. This is the end of the build job.

If the build job fails then the team is then emailed by Github.

The next job is the release job, this checks whether the integration has the appropriate tag, in the format vX.X.X

If an appropriate tag is found, the following steps are run, else the job is skipped:

First, the download artifact action is run for each of the platform's JAR file artifacts outputted by the build job. These files are renamed into names appropriate for release, being the name of the game along with the platform built for (windows, macos, ubuntu). A final github action is then run which makes a release, which includes these 3 JAR files, alongside the source code.

The workflow takes just under 2 minutes to complete, providing rapid feedback to the developers.

The image below represents the two jobs in the workflow described above.