# Continuous Integration

Group 26 - Spice Traders (prior team 22)

James McNair
Dan Wade
Alice Cui
Charlie Crosley
Rob Murphy
Marc Perales Salomo

## **a) Continuous Integration Methods**

For our project we decided to automate as much of the build and testing process as possible, this is so we can spend as much time as possible working on the actual project instead of doing repetitive tasks around it. Automating the testing process also ensures that tests are continuously run so any changes in the code that cause a test to fail are detected as quickly as possible. In our project we decided to automate the following:

- Deployment of our website - We choose to do this so changes can be made quickly and it is easy to visually check the updated website at the same time
- Testing of our game - Automated testing is advantageous as it is easy to forget to run Unit Tests or to break another section of the program that is not currently being tested. Automating the process will both save time and improve reliability.
- Building our game - Our game is going to automatically be packaged into a jar file immediately after tests are run, so we can be sure the latest jar file has passed all tests and is up to date.

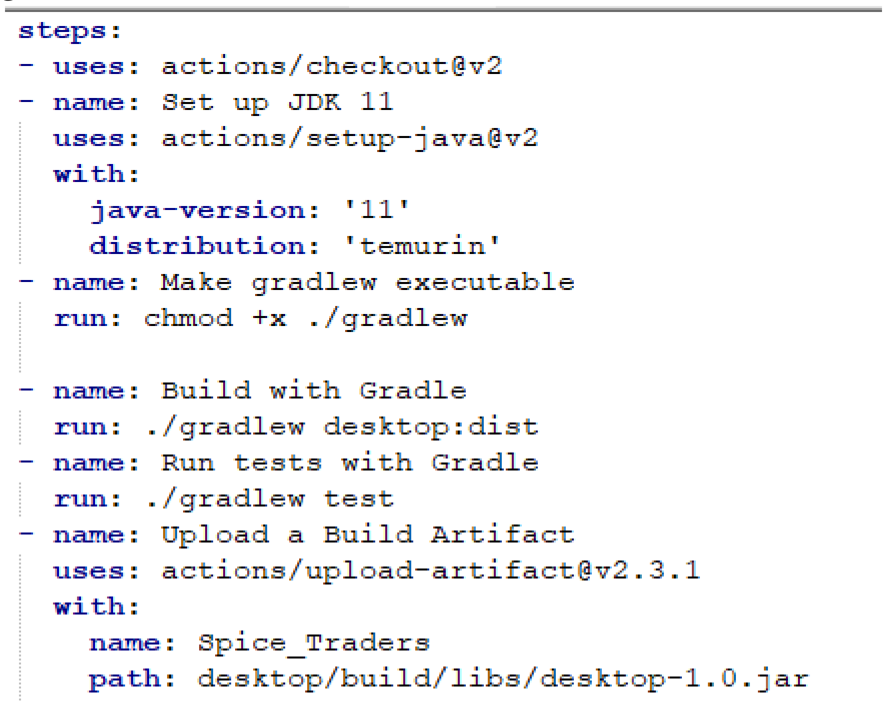
Along with automating a lot of the simple work, continuous integration involves a change in mindset for the programmers. We intend to merge any changes we have on our local machines whenever our versions are able to pass all tests. This will help with continuous integration as it will keep the shared copy of our game up to date, regularly merging our local copies into the shared copy will also reduce the risk of merge conflicts and duplicate code as we will be able to see all code to date without waiting for sections of code from another person.

**b) Continuous Integration Infrastructure**

For all our infrastructure we are using GitHub, this has the benefit of being the same place that our code is stored, and so it is easily accessible and makes it harder for us to forget about in the future.

For our website deployment, we are using GitHub Pages, this is because with only a few setting changes GitHub already has support to do the rest for us, so we don't have to worry about testing our continuous integration solution as.

Building and testing of the game is done using GitHub Actions, this allows us to write a simple YAML file which will describe how a server hosted by GitHub will be configured to test and build our game. Pictured below is a part of the script used to test and build our game.



```yaml
steps:
- uses: actions/checkout@v2
- name: Set up JDK 11
  uses: actions/setup-java@v2
  with:
    java-version: '11'
    distribution: 'temurin'
- name: Make gradlew executable
  run: chmod +x ./gradlew

- name: Build with Gradle
  run: ./gradlew desktop:dist
- name: Run tests with Gradle
  run: ./gradlew test
- name: Upload a Build Artifact
  uses: actions/upload-artifact@v2.3.1
  with:
    name: Spice_Traders
    path: desktop/build/libs/desktop-1.0.jar
```

Using GitHub actions has the advantage of direct integration with Pull requests and will not allow us to merge code into the main branch if the above steps find that tests are failing or the jar file is unable to be built. This means that at any time we can be certain that the version of our game stored on GitHub will be passing all the tests we have written along with being able to be built into a single jar file.