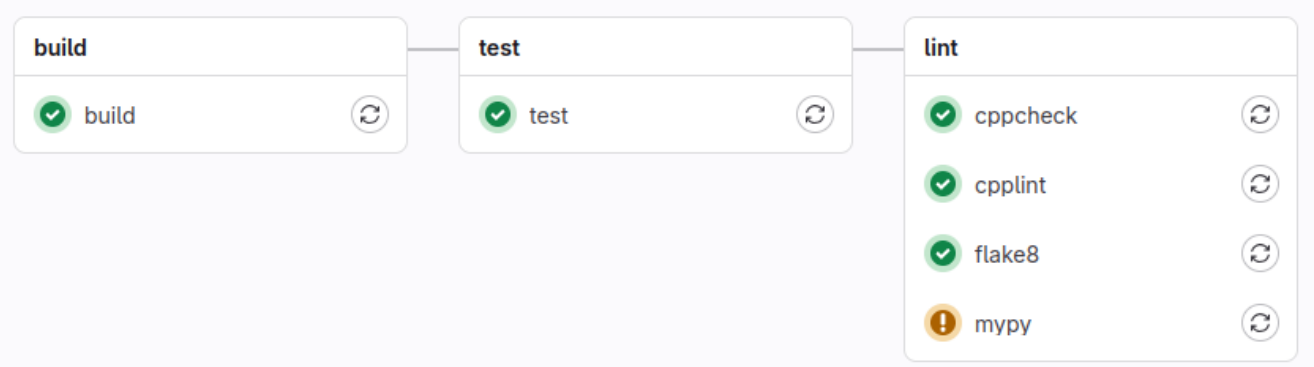
5.2 - Construct an appropriate CI pipeline for the software

Design Inspiration

From my experience in EUFS (Edinburgh University Formula Student), there are three crucial steps inside our pipeline: Build, Test, Lint.



Example of one of our gitlab pipelines

Building is crucial for compiled projects especially, and is the process of ensuring the pushed code is *runnable*.

Testing focuses on running the tests inside the pipeline, automating the process of checking crucial functionality hasn't been broken by a change.

Linting is especially useful for projects with larger teams - and is the practice of ensuring certain code quality standards are met throughout the project, for example enforcing that no line is longer than 100 characters, encouraging developers to improve readability.

My Design

Inspired by our pipelines at EUFS, I would follow almost exactly the same pipeline design. For my Java project rather than a C++ or Python one (as is common in EUFS), we would have a build with Maven step. Not only does this ensure that our java code is syntactically correct, but building with Maven *also runs any recognised tests*. If these tests fail, the build fails, which means the pipeline fails. This deals with the build and test stages at once.

This was easy enough to implement using GitHub's actions, and so I added this part of the design to my project.

If I had the time to improve this further, I would also add the linting stage. Linting is a crucial part of code quality, and having a force-fail linting step both ensures that merged code is up to scratch, and teaches developers to write better code (so as to do less linting!).