SWEN 438 *Special Topic:DevOps*

Laboratory 2: GitLab

The purpose of this laboratory is to set up a working CI/CD pipeline using our local GitLab instance. In particular, this could be beneficial for Assignment 3.

Getting Started

To begin with, create a suitable repository on GitLab (e.g. entitled `Lab_2`). Next, download the sample code (linked from the SWEN438 homepage under LectureSchedule). *You should commit and push the Java source files to your repository.*

You will see that includes build files for both Maven and Gradle and you are free to use whichever you prefer. Eitherway, you should ensure that the build is working on your local machine. To do this, run "`mvn test`" or "`gradle test`" from the command-line and it should build all the files and run the tests. The output will be something like this:

```
> mvn test
[INFO] Scanning for projects...

...

-------------------------------------------------
TEST 70
-------------------------------------------------
[INFO] Tests run: 70, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.023s
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 70, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.470 s
[INFO] Finished at: 2021-08-03T15:14:31+12:00
[INFO] ------------------------------------------------------------------------
```

NOTE: To be absolutely sure things are working correctly, you should edit the file `src/test/java/chuckie/tests/ChuckieEggTests.java` and deliberately break one of the tests. Then, rerun the build command from above and check that it correctly identifies a failing test.

Part 1: Creating a GitLab CI/CD Pipeline

At this point, we are going to create a GitLab CI/CD Pipeline which will build and test our project whenever we push changes to our repository. This means we will be notified when changes we have pushed "break the build". To do this, we need to create a special file (`.gitlab-ci.yml`) in the repository root with the following contents:

```
image: maven:latest

default:
  tags:
   - docker

build:
  script:
    - mvn test
```

(if you prefer to use gradle then replace "`maven:latest`" with "`gradle:latest`" and "`mvn test`" with "`gradle test`")

You should commit and push this file to your repository. At this point, visit your repository with a webbrowser and find the `CI/CD` tab (its icon is a rocket). You should see that is a pipeline listed there, which may still be "in progress". If you click on the pipeline identifier (e.g. #53348) and then the build item at the bottom you should be able to see the physical output from the pipeline run. Hopefully everything passed OK!

At this stage, the goal is again to check that the pipeline is indeed working. To do this, you should try various edits which "break the build" in some way. For example, editing a source file so that it no longer compiles or editing the test file (as above) so that at least one test fails, etc. Push these changes up to the repository and you should see the build failing!

Part 2:

The next part of the lab is to add a second job to the pipeline. This job is going to use Maven again to run the Checkstyle tool on our source files. This is a linting tool) for Java which you may have encountered in other courses.

We'll schedule our checkstyle job to come *after* our build&test job above. To do this, we'll specify there are two *stages* (`test` and `lint`). Update your `.gitlab-ci.yml` as follows:

```
image: maven:latest

stages:
  - test
  - lint

default:
  tags:
   - docker

build:
  stage: test
  script:
    - mvn test

checkstyle:
  stage: lint
  script:
    - mvn install
```

This first defines the two stages, and then defines the two jobs `build` and `checkstyle` (one for each stage). In principle, we could add more jobs to the same stage and these could then be run concurrently.

The advantage of splitting our CI/CD Pipeline into multiple jobs is that this helps us determined more easily where the problem was. You should find that the `build` jobs succeeds, whilst the `checkstyle` job fails. If you look carefully out the output from this job you will see that there are a number of failing checks on the source code.

Submission and Marking

For this first lab, please submit a file `submission.txt` containing a link to your repository via the ECS Submission system https://apps.ecs.vuw.ac.nz/submit/SWEN438/Laboratory_2.

Full marks will be earned for meeting the following criteria:

1. Build&Test job is shown to run on GitLab
2. Checkstyle job is shown to run on GitLab