Continuous integration with Travis-CI

Getting Started

This guide briefly explains the idea of continuous integration (CI) and gives a step-by-step guide to implement a simple CI setup using Travis CI and git (via GitHub).

Continuous integration

The idea of continuous integration is simply to merge developing effort into a central, or integrated, environment[1]. The main reason is to avoid heterogeneous environments. Merging code from many developers can be hard because of local dependencies or settings which 'works on my machine' (also known as integration hell). In continuous integration, the code will be pushed to a different clean server and tested, before being deployed to production.

In practices like eXtreme Programming (XP) where tests are put in place before code, automated testing (with for instance JUnit) is exploited to act as a 'guard' to the production environment. If the tests are green the code can be released. If they are not green, someone needs to cough up a good excuse for ruining production code (and not release the code).

This becomes especially useful if testing can be integrated into the normal development flow. A common method is to use versioning tools like git. You are using this anyway, so in a sense, you can get CI for free. Git is especially good at this because it features commit hooks which can be run whenever someone is committing. By using these hooks, tests can be set up to run automatically when someone pushes to a shared repository.

This guide will show you how you can use CI with git (on GitHub) and Travis-CI.

Setting up a GitHub repository

For this exercise, you should fork the following repository and create a copy under your own GitHub account: https://github.com/cphdat3sem2019spring/travisGettingStarted
The first thing you should do is, remove the code in the README.md file that renders the Travis status icon. At the end of this tutorial you will insert it again, but so it provides a status for YOUR project.

This is a very simple Java project which contains a function which returns the integer 10, along with a single test that verifies that the method actually returns 10. The project uses Maven to run the test and fetch dependencies. After forking the project to your own GitHub account, try to pull it and run the test. It fails, but do not fix it yet.

What is Travis?

Travis is a service which simply builds code and runs the test. The good thing is that it is free for open-source projects, but do not get too attached to it. There are many other CI services out there. Another popular alternative is Jenkins which can be installed and run with private projects. For instance, Jenkins could be installed on a Digital Ocean server. However, that takes time to install so for the purpose of this exercise, and this semester, we just use Travis.

[1] https://en.wikipedia.org/wiki/Continuous_integration

Using Travis

The goal of this exercise is to tell our CI server (Travis) to monitor and watch code being pushed to your repository. Whenever Travis figures that out, it should test it to check whether some smug developer did write bad code. To figure out which language and code to run, Travis uses a configuration file which must be placed in the project root. Take a look at the file, it is called .travis.yml (notice the . in the beginning). As you can see it is very simple. First of all, we tell Travis what language we are using. Secondly, we have to tell Travis which JDK to use, we will choose JDK 8 (for future use, you, and your group should decide for which Java Version to use, on your laptops and on Digital Ocean).

Note that we actually do not write anything about execution in the *.yml* file. Since Maven is such a popular Java build tool, Travis actually guesses that it is a Maven project (because of the *pom.xml* file) and simply defaults to mvn test. Naturally, this can be changed to do anything you want[2]. If you want to explore this have a look at their documentation on using Java: https://docs.travis-ci.com/user/languages/java/ . In this example, we will only be using the default command.

Setting up Travis-CI

Before running Travis you must connect your GitHub project to the CI service. The easiest way to do this is to create an account on Travis using your GitHub account and giving Travis access to all your repositories. This way Travis can set up all the commit hooks for you. Of course, this will give Travis access to your code so if this annoys you can trigger builds manually using this guide.

Now, create a user via https://travis-ci.org/

If you integrated your service with GitHub you should be faced with a list of your GitHub projects after creating a user. Each project has a switch which tells Travis-CI to listen for commits on that project. Find your forked project and turn on Travis integration. If you get this step to work you should see a page with a log file like this:

Notice line #653 where our test fails. It expects 10 but gets 5. You should see a similar log when your project runs. If you did not get it to work, check out Travis for beginners. But please be critical. Use your own project names and languages (protip: you are not writing in PHP).

Completing the guide

That was it. Really. Your project is now using CI. Every time you push to your repository, Travis will rebuild and re-test. If you go to Travis you should see your project listed and in the process of being built for the first time. To the right of the project name is a grey icon and a status icon. When the build is over it will tell you that your build failed (d'uh).
If you click on the icon, you can actually get a status image which can be posted anywhere on the web.

[2] Literally. You can execute bash code here if you want to.

Do this, but instead of getting an image URL, select Markdown and copy paste that line into the README.md file inside your GitHub project. Commit and push it. The GitHub page for your project should now show a Build failed-image so people can see how bad you are at testing.

Let us fix that. Try to make the tests succeed and push it to GitHub. If everything goes well, Travis should build another (successful) version and your status image should go green.